

POSTER PAPERS

MULTIAGENTS

Network Meta-Reasoning for Information Assurance in Mobile Agent Systems

Donovan Artz

Max Peysakhov

William Regli

Department of Computer Science

Drexel University

Philadelphia, PA 19104-2875

1 Introduction

This paper develops a practical means of measuring information assurance for mobile agent systems operating on wireless, ad hoc networks based on meta-reasoning [Dix *et al.*, 2000; Xuan *et al.*, 2001] to improve the security of communication. Figure 1 shows an agent system and its two distinct layers of communication: host-to-host and agent-to-agent. Given the plethora of new techniques for identifying network intruders, we study the *compromised host* problem: determining the appropriate response to an identified intruder. In the context of a mobile, multi-agent system operating on an ad hoc network [Forman & Zahorjan, 1994], it is not merely a simple matter of removing the compromised hosts and its agents. While keeping the compromised host can result in *information disclosure*, removal of the host can degrade or even sever the network. We develop a state description for an agent system and introduce a measure of *information assurance* for the system in terms of the integrity of the messages delivered to the agents in a given network state. Agents have three responses to a compromised host: *ignore* the compromised host; *reroute* around the compromised host using network route redundancies; or *remove* the compromised host, by having the agents instruct their hosts to eliminate it from the network. These responses are shown in Figure 2.

2 Technical Formulation

A state description for a *mobile agent network* is defined in terms of sets of hosts (H) and agents (A). Given H and A , we can define:

- *Agent topology*, T_A , is a graph of the connections between agents in the agent layer: $T_A = (A, L_A)$ where $L_A \subseteq A \times A$.
- *Host topology*, T_H , is a graph of the direct connections between hosts in the network layer: $T_H = (H, L_H)$ where $L_H \subseteq H \times H$.
- *Host routing*, $\mathcal{R}$, is a set of sequences of hosts which describes the network routes used between hosts in the network layer: $\mathcal{R} = \{R_{i,j} : h_i, h_j \in H\}$ where $R_{i,j}$ is the current physical network route from host h_i to host h_j . If no route exists from h_i to h_j , then $R_{i,j} = \emptyset$.
- *Agent location*, η , is a function mapping an agent to the host on which it is physically located in a given state:

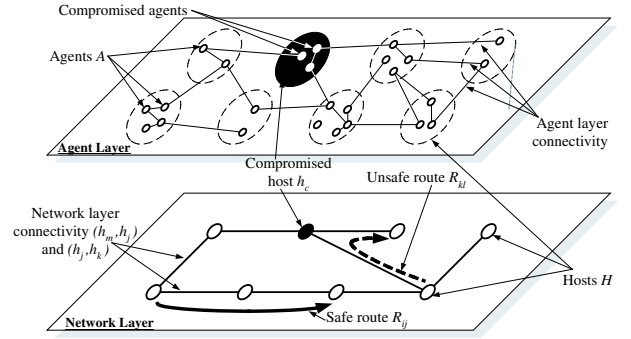


Figure 1: Nodes are either physical hosts or agents; edges are network host or agent-to-agent connections. The shaded node is a *compromised host*. The bold arrow is a *safe network route*; the dashed arrow is a *compromised network route*.

$\eta : A \rightarrow H$. Inversely, the function η^{-1} maps a host to the set of agents located on that host.

The state of a mobile agent network is defined as a tuple: $N = \langle T_A, T_H, \mathcal{R}, \eta \rangle$

2.1 Evaluation of an Agent System Network

We model the information assurance level in an agent network by analyzing how agents communicate. Observe that agents must send messages to other agents in order to collaborate in any decision procedure. In a decision procedure, typically certain agents are *authorities* that collate voting messages from all other agents involved in the decision. In the context of T_H , any host housing an agent involved in at least one decision procedure authority is a *sink* in the network topology graph into which messages flow. For a simple decision, there may be only one sink; in the most complex case, all hosts are sinks. This paper will assume the most complex case, in which all hosts are housing decision authority agents.

For any decision procedure, messages sent to the decision authority can be classified into *successful messages* and *failed messages*. A *successful message* is delivered without using the compromised host. A *failed message* either: (1) originates or ends at the compromised host; (2) is routed through the compromised host; or (3) cannot be routed because no network route exists.

For a given N , a change in T_H can cause a change

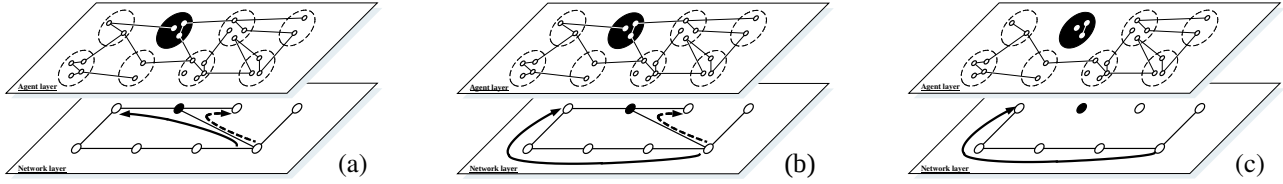


Figure 2: Responses to compromised hosts: (a) ignore; (b) reroute; (c) remove.

in routing. If a route changes, the time taken to transmit a message over that route may also change. A change in message delivery time can negatively impact a decision procedure. Moreover, a compromised host contains agents that may violate their expected behavior in a decision procedure. Both factors must be considered when evaluating a state N . We define two values that can be used to evaluate N with respect to each factor:

1. A *message integrity rating*, which relates successful messages received to failed messages.
2. A *time rating*, which is an estimate of optimality for the current network routes.

Measuring Message Assurance. If each agent is in communication with all of the other agents, $n_i = (|A| - |\eta^{-1}(h_i)|)|\eta^{-1}(h_i)|$ is the total number of messages that the agents on host h_i expect to receive from agents on other hosts per unit of time given state N . Based on the current routes $\mathcal{R}$, one can calculate the number of successful messages received on host h_i , $\text{succ}(h_i)$. The *message integrity rating* for host h_i is computed as: $m_i = \frac{\text{succ}(h_i)}{n_i}$. Note, as m_i increases, the integrity of messages sent to host h_i also increases. When all messages sent to h_i are successful, $m_i = 1.0$. When all messages sent to h_i fail, $m_i = 0.0$. The mean *message integrity rating* over the entire mobile agent network in state N is: $\bar{m} = \frac{\sum_{i=1}^{|H|} m_i}{|H|}$.

Measuring Network Routing Efficiency. The trade off is between message integrity and the timeliness of message delivery. Network routing algorithms find sets of routes that minimize some value for all routes in a network. In general, routing algorithms use a weight function on each route and find the shortest, single source, paths to all vertices in T_H . In this context, the function ρ represents the network routing algorithm which returns a set of shortest path routes for the set of hosts, given their current physical network topology: $\mathcal{R} = \rho(H, L_H, w)$, where w is the edge weight function. There are several schemes that can be used to weight routes in wireless, ad hoc networks [Royer & Toh, 1999], all of which can be approximated or bounded using a w that returns a value proportional to the time required to transmit a message through the network. The units of time returned by w are used for relative comparison of network routes, which we normalize w to simplify computations: $w : \mathcal{R} \rightarrow [1, |H|]$, where $w(R_{i,j}) = |H|$ signifies that the route from host h_i to h_j is non-existent and weight of the longest possible route is $|H| - 1$. Now we can define a *time rating*, t_i , for the network

routes used by host h_i as: $t_i = \frac{\sum_{j=1, j \neq i}^{|H|} w(R_{j,i})}{|H|(|H|-1)}$. Note, as t_i increases, the routing efficiency to host h_i decreases. When the routing efficiency is minimized (i.e., no connections exist) for a host h_i , $t_i = 1.0$. As the routing efficiency increases (i.e., shorter routes are used) for host h_i , t_i approaches 0.0. Hence, the mean *time rating* for the entire mobile agent network in state N is: $\bar{t} = \frac{\sum_{i=1}^{|H|} t_i}{|H|}$.

Assurance for Whole Mobile Agent Network. A linear combination of *message integrity rating* and *time rating* defines a utility function assessing a mobile agent network in state N in terms of both assurance and routing efficiency:

$$V(N) = \frac{\alpha \bar{m} - (1 - \alpha) \bar{t} + 1}{2} \quad (1)$$

α is a coefficient between 0 and 1 that determines the balance between assurance and network performance. If $\alpha = 1.0$, only message integrity is considered; if $\alpha = 0.0$, only time efficiency is considered. Note that $V : N \rightarrow [0.0, 1.0]$, where $V(N) = 1.0$ is the best possible result.

2.2 Operators on an Agent System Network

Using Equation 1, agents can decide how to operate on their network. Naturally, the *ignore* operator does nothing. The *reroute* operator generates a new set of network routes, using only safe routes wherever possible. If $\mathcal{R}'$ is the set of *safest possible routes*, the resulting mobile agent network $N' = \langle T_A, T_H, \mathcal{R}', \eta \rangle$ is generated using the reroute operator on N (Figure 2(b)). Given the routing algorithm ρ and route weight function w , the following algorithm can be used to compute N' :

Algorithm 1 *reroute*(N, h_c)

```

 $\mathcal{R}' \leftarrow \rho(H - \{h_c\}, L_H, w)$ 
 $\mathcal{R}' = \mathcal{R}' \cup \{R_{i,j} : R_{i,j} \in \mathcal{R} \wedge R'_{i,j} \in \mathcal{R}' \wedge R'_{i,j} = \emptyset\}$ 
return( $\langle T_A, T_H, \mathcal{R}', \eta \rangle$ )

```

The *remove* operator results in the complete removal of the compromised host from participation in the agent system's underlying network. The new mobile agent network, $N'' = \langle T'_A, T'_H, \mathcal{R}'', \eta \rangle$, is the result of applying the removal operator on N and is generated by the following algorithm:

Algorithm 2 *remove*(N, h_c)

```

 $A' \leftarrow \{a : a \in A \wedge \eta(a) \neq h_c\}$ 
 $L'_A \leftarrow \{(a, b) : (a, b) \in L_A \wedge a, b \in A'\}$ 
 $T'_A \leftarrow (A', L'_A)$ 
 $H' \leftarrow H - \{h_c\}$ 
 $L'_H \leftarrow \{(h_i, h_j) : (h_i, h_j) \in L_H \wedge h_i, h_j \in H'\}$ 

```

$$\begin{aligned}
T'_H &\leftarrow (H', L'_H) \\
\mathcal{R}'' &\leftarrow \rho(H', L'_H, w) \\
\text{return} &(\langle T'_A, T'_H, \mathcal{R}'', \eta \rangle)
\end{aligned}$$

In order to select the operator resulting in the highest valued agent system, consider the values $V(N)$, $V(N')$, and $V(N'')$. The highest of these values represents the best action for the agent system.

3 Application: A Compromised Auction

The disclosure of bids to a compromised host can affect the intended, timely, outcome of an auction. A Vickrey auction is a sealed bid auction where the second highest bid is paid by the highest bidder [Vickrey, 1961]. All bidders maximize their payoff if they employ a truthful bidding strategy. Agents often use Vickrey auctions to acquire resources. Each agent submits its bid to an auctioneer host h_1 (the sink)—but unless the physical host of a bidding agent is directly connected to h_1 , the message containing the bid must pass through other hosts in the agent system. All agents operating under normal conditions have neither the intent nor the capability of reading bids that are routed through their physical hosts.

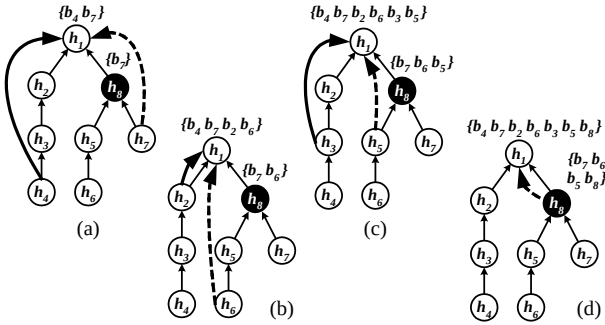


Figure 3: Bid propagation to host h_1 in a Vickrey auction: (a) h_4 and h_7 bid (b) h_2 and h_6 bid (c) h_3 and h_5 bid (d) h_8 bids

First, a compromised host, h_c can read all bids that are sent directly to or routed by means of h_c and potentially corrupt the auction. In this case, instead of maximizing absolute payoff, the bidding agents on h_c maximize their payoff *relative* to other bidding agents. In this type of “antisocial bidding” [Brandt & Weiss, 2001], assuming there are n bidding agents in the agent system, h_c most successful if it knows all bids $b_1, b_2, \dots, b_n$ placed by *all* of the other bidding agents. The worst case is when all physical hosts use routes that contain h_c . In general this is not the case. Hence, if there are n' hosts that use routes containing h_c , the probability that the highest (or any) bid is disclosed to h_c is equal to n'/n .

Secondly, time is an issue: auctioneers are not willing to wait indefinitely for all bidders to respond. In any given decision problem there is some threshold, τ , such that, if a bidder is more than τ hops from the auctioneer, its bid will not reach the auctioneer in time. Let U_i be the set of all hosts that communicate with host h_i via a route longer than τ hops: $U_i = \{h_j : h_j \in H \wedge |R_{j,i}| > \tau\}$ U_i may contain hosts that use a route containing the compromised host. To adjust for this overlap, compute the set C_i of hosts affected by the

operator	$\bar{m}$	$\bar{t}$	$V(N)$	C_1/n	U_1/n
ignore	0.286	0.250	0.509	0.5	0.0
reroute	0.714	0.321	0.598	0.25	0.25
remove	0.833	0.524	0.577	0.0	0.42857

Table 1: The terms and result of $V(N)$ using $\alpha = 0.5$, the probability of compromised messages $\frac{C_1}{n}$, and the probability of unreceived messages $\frac{U_1}{n}$ for the result of each operator.

compromised host, but not by the required message delivery time: $C_i = \{h_j : h_j \in H \wedge h_c \in R_{j,i} \wedge |R_{j,i}| \leq \tau\}$

The disclosure of bids to a compromised host during this decision procedure is illustrated in Figure 3. Assume for timing that $\tau = 3$. In this example, the probability $|C_1|/n$ is representative of the effect of compromised messages, and the probability $|U_1|/n$ is representative of the effect of time. As either probability increases, the value of the underlying mobile agent system should decrease.

Table 1 demonstrates how $V(N)$ can be used to minimize the effect of a compromised host in a Vickrey auction. As the probability of compromised messages increases, the *message integrity rating* decreases. As the probability of unreceived messages increases, the time rating also increases. The operator yielding the highest value in this example is *reroute*.

4 Conclusions

This paper developed a utility-based model for agents to balance information assurance and network routing efficiency. We have discovered that there exists a natural tradeoff between information assurance and network routing efficiency for ad hoc mobile agent networks. Further, by empowering agents to decide for themselves how they communicate at the network level, one can increase the overall level of message integrity in an agent system. Our approach involves a novel exploitation of properties of ad hoc networks, enabling mobile agents to automatically adapt to changes that affect the security of their communication and migration. The capability to dynamically reason about the state of their network will provides new possibilities for secure computing.

References

- [Brandt & Weiss, 2001] F. Brandt & G. Weiss. Antisocial agents and Vickrey auctions. *Wkshp on ATAL*, V 2333, pp 335–347, 2001.
- [Dix et al., 2000] J. Dix, V. S. Subrahmanian, & G. Pick. Meta-Agent Programs. *J. Logic Prog.*, 46(1):1–60, 2000.
- [Forman & Zahorjan, 1994] G. H. Forman & J. Zahorjan. The challenges of mobile computing. *IEEE Comp.*, 27(4):38–47, Apr 1994.
- [Royer & Toh, 1999] E. M. Royer & C. K. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Pers. Comm.*, 6(2):46–55, Apr 1999.
- [Vickrey, 1961] W. Vickrey. Counter speculation, auctions, and competitive sealed tenders. *J. Fin.*, 16(1):8–37, 1961.
- [Xuan et al., 2001] P. Xuan, V. Lesser, & S. Zilberstein. Communication decisions in multi-agent cooperation: model and experiments. *AAAS*, pp 616–623, 2001.

Towards Cooperative Negotiation for Decentralized Resource Allocation in Autonomic Computing Systems

Craig Boutilier

Department of Computer Science
University of Toronto
Toronto, ON, M5S 3H5, Canada
cebly@cs.toronto.edu

Rajarshi Das Jeffrey O. Kephart William E. Walsh

IBM T.J. Watson Research Center
19 Skyline Dr.
Hawthorne, NY 10532, USA
{rajarshi, kephart, wwals1}@us.ibm.com

Abstract

Resource allocation is a key problem in autonomic computing. In this paper we use a data center scenario to motivate the need for decentralization and cooperative negotiation, and describe a promising approach that employs preference elicitation.

1 Resource Allocation in an Autonomic Computing System

An autonomic computing system is designed to drastically reduce the role of human administrators by automating most of the managerial decision making [Kephart and Chess, 2003]. Automated resource allocation is necessary for an autonomic computing system to optimize its performance. In the large, distributed autonomic computing systems we would expect in big businesses, resource allocation will occur at multiple scales. Local allocation decisions will be made within individual elements (servers, databases, storage units, etc.) and small clusters of elements. Local clusters will contend for pools of resources in the larger domain, or across administrative domains. Although we can generally assume that elements in an autonomic computing system of a single corporation will be cooperative, sharing the common goal of optimizing total business value, the complexity of local information will often preclude centralized allocation across the entire system. Cooperative negotiation, using preference elicitation techniques, can be an effective approach to decentralization.

To motivate the problem, consider the problem of resource allocation within a *data center*. The data center provides information technology resources to multiple organizations, each in a separate domain governed by a *workload manager* (WLM). Each WLM decides how to allocate resources in its domain to maintain quality of service (QoS) for each of a set of n classes of transactions.

The QoS specification for a transaction class c includes terms describing monetary payments or penalties as a function of the measured attributes of the service provided to that class by the data center. For simplicity, we shall consider here just a single attribute, the response time t_c for class c . We denote the revenue function for class c (including all rewards and penalties) by $r_c(t_c)$.

In a real system, a WLM can adjust various parameters $\vec{f}$ for multiple resources. Here, we assume a single resource, with quantity L . Given fixed L , $\vec{f}$, and class demand vector $\vec{d}$, and assuming that the resources have no incremental costs, the expected total revenue is then:

$$R(\vec{f}, L, \vec{d}) = \sum_c \int_{t_c} r_t(t_c) \Pr(t_c | \vec{f}, L, \vec{d}) \quad (1)$$

A WLM's internal resource optimization problem is to find a feasible $\vec{f}$ that maximizes Eq. (1).

To handle fluctuations in client demand, a single provisioner at the data center periodically reallocates resources among the WLMs. Denoting an individual WLM by i , the resource allocation problem for the provisioner is to compute:

$$\arg \max_{\vec{L}_i} \sum_i U_i(L_i) \text{ where} \quad (2)$$

$$U_i(L_i) = \max_{\vec{f}} R_i(\vec{f}, L_i, \vec{d}) \quad (3)$$

The provisioner can compute (2) centrally if it has a good model of the internal operation of each WLM and can obtain all relevant state information. In a real system however, the model and data necessary to compute $\Pr(t_c)$ may be complex and large. Moreover, in a system with transient, heterogeneous components (e.g., differently configured WLMs or different components altogether), the internal models of the components may simply not be available to the provisioner. In these cases, it is necessary to decentralize the resource allocation problem. That is, WLMs would perform local computations and send summary information to the provisioner.

A natural division of labor is to have each WLM send its entire U_i curve, so that the provisioner can compute Eq. (2). Figure 1 shows an example of curves from two WLMs, each with two transaction classes.¹ The provisioner wishes to find the peaks of the aggregate curve, also shown.

2 Cooperative Negotiation

To compute its full U_i curve, a WLM must solve Eq. (3) for each feasible L_i . If each r_c is in a computationally manageable form (e.g., piecewise linear or quadratic) and we have a

¹We computed the utility curves assuming a simple M/M/1 queue model for each t_c . L is the total service rate available, and the f_c component of $\vec{f}$ indicates the fraction of L given to class c .

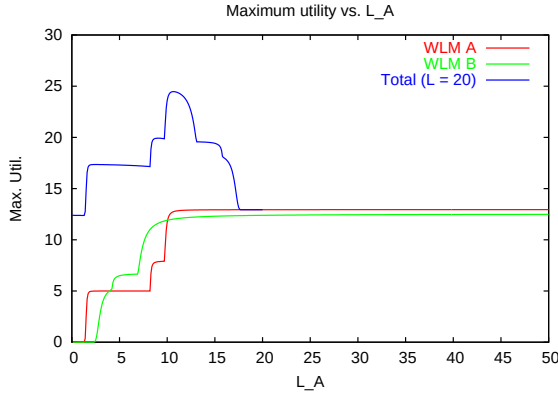


Figure 1: Maximum system utility as a function of allocation, with total resources $L = 20$. Curve “WLM A” indicates maximum utility to A as a function of L_A . Curve “WLM B” indicates maximum utility to B as a function of L_B . Curve “Total” indicates total utility as a function of L_A provided to A (with $L - L_A$ provided to B).

simple model for t_c (e.g., M/M/1 queue), then computing each $U_i(L_i)$ point may be tractable. However, in typical real systems the dependency of the service attributes upon resources and demand is likely to be sufficiently complex as to require a combination of optimization and simulation to compute each $U_i(L_i)$ point. Moreover, WLMs will often have substitutable and complementary preferences over different quantities of *multiple* goods, giving rise to large, expensive-to-compute, multidimensional U_i curves. Such complexities would make it infeasible for a WLM to send its entire U_i to the provisioner.

Instead, we propose to model the resource allocation problem as cooperative negotiation. In the context of autonomic computing, cooperative negotiation is not simply non-cooperative negotiation with the simplifying assumption that agents are non-strategic. Rather, the objective is to achieve the right balance between global optimization and negotiation time. We are developing preference elicitation techniques [Boutilier, 2002] to address this problem. We can view the negotiation for resources between the WLMs and the provisioner as involving computation or “elicitation” of relevant parts of the U_i curves.

Partial elicitation may be feasible in the case of negotiation among WLMs for different resource levels. It will often be possible to identify the region of $\vec{L}$ -space in which the optimal allocation lies without complete knowledge of these utility functions. The provisioner could be given a small number of samples of the utility functions $U_i(L_i)$ for WLMs i . Making simple monotonicity assumptions, the provisioner could determine the region of allocation space in which the optimal allocation lies.² For instance, having samples of the two (lower) U_i curves in Figure 1 at points $L_i = \{10, 15, 20\}$ for $i = A, B$, is sufficient to determine that the optimal allocation lies somewhere in the region $L_A \in [10, 15]$.

We are currently developing incremental utility elicitation

²Monotonicity of U_i seems natural, corresponding to a “free disposal” assumption.

procedures in which the provisioner gradually narrows down the region in which the optimal allocation lies until a decision which is guaranteed to be ϵ -optimal is found. A rough sketch of one such procedure is presented here. Assume that each WLM has provided evaluations of its utility function U_i at a set of m sample points $\tau_i^1 < \tau_i^2 < \dots < \tau_i^m$; we assume that τ_i^m is the maximum value of L_i and bounds U_i under any allocation. With this small set of samples, the provisioner is assured that $U_i(\tau_i^k) \leq U_i(L) \leq U_i(\tau_i^{k+1})$ for any $\tau_i^k \leq L \leq \tau_i^{k+1}$. Armed with this information, it is reasonably straightforward to determine the *maximum regret* of any allocation $\vec{L}$. Furthermore, we can also determine the allocation that has *minimax regret* given this incomplete knowledge of the utility function:

$$\vec{L}^* = \arg \min_{\vec{L}} \max_{\vec{u}} \max_{\vec{L}'} \sum_i [u_i(L'_i) - u_i(L_i)]$$

Here $\vec{u}_i$ ranges over the set of utility functions U_i consistent with the bounds above. Minimax regret is a reasonable error criterion, and the minimax-optimal allocation $\vec{L}^*$ can be determined using a tractable series of linear programs.

Incremental elicitation arises when the minimax-optimal allocation has a max-regret level that is too high, say, greater than ϵ . In this case, the provisioner does not have enough information about the U_i curves to determine an allocation whose worst-case error is less than ϵ ; this necessitates additional samples of the U_i curves. We have developed intelligent query strategies that efficiently determine points L_i for which knowledge of $U_i(L_i)$ will “quickly” reduce minimax regret. Once sufficiently many points have been evaluated, minimax regret will reach an acceptable level and an allocation whose error is bounded by ϵ can be offered. We do recognize that the procedure becomes more complicated with multidimensional U_i curves.

This procedure obviates the need for WLMs to compute their entire utility curves. The complex optimization required to determine $U_i(L)$ need only be applied by WLM i at a small collection of points; and these points are determined by the provisioner based on information about the utility functions of *other* WLMs. In this sense, such incremental elicitation techniques can truly be viewed as a form of collective negotiation between WLMs and the provisioner.

We are exploring error criteria other than minimax regret in our general model. Optimization w.r.t. specific utility constraints, and determining queries that reduce error, would both require tailoring to the specific criterion adopted; but the same incremental framework still applies. For instance, if distributional information over the U_i curves is available, Bayesian decision criteria could be used (e.g., an allocation that maximizes expected utility w.r.t. $\Pr(\vec{u})$).

References

- [Boutilier, 2002] Craig Boutilier. A POMDP formulation of preference elicitation problems. In *Eighteenth National Conference on Artificial Intelligence*, pages 239–246, 2002.
- [Kephart and Chess, 2003] Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–52, 2003.

A Formalization of Equilibria for Multiagent Planning

Michael Bowling, Rune Jensen, and Manuela Veloso

Computer Science Department
Carnegie Mellon University
Pittsburgh PA, 15213-3891

1 Introduction

Traditionally, planning involves a single agent for which a planner needs to find a sequence of actions that can transform some initial state into some state where a given goal statement is satisfied. But in general, “planning” can be viewed as being concerned with a general action selection problem. The planning framework has been extended from the classical deterministic plan generation problem along many other dimensions, in particular nondeterministic actions. With the introduction of nondeterministic actions, the presence of an environment and other agents can become a consideration. In fact, actions may have nondeterministic effects not only because of the uncertainty of their own execution, but also due to the uncertainty of the actions of other agents. The possible presence of other agents as executors is the challenge of “multiagent planning.” The interest in this area has been steadily increasing and many issues remain open.

Despite the existence of planning languages with explicit models of uncontrollable agents [Jensen and Veloso, 2000], we believe that there has not been a formal discussion of the space of multiagent plans or *solutions*. In this work, we do not focus on the problem of plan generation for multiagent planning. Instead, we focus on the interesting question of analyzing and comparing solutions for multiagent planning. Our motivation comes from making an analogy with game theory and the notion of equilibria [Owen, 1995].

Inspired by game theory and extending previous formal definitions of single-agent planning [Cimatti *et al.*, 2000], in this paper, we introduce a formal definition of *equilibria* for multiagent planning.

2 Planning Equilibria

In this section, we formalize the concept of a multiagent planning equilibrium. In order to help make these concepts clear we will first describe an example, which is small enough to easily enumerate all of the states.

2.1 A Simple Example — The Narrow Doorway

Consider a two agent robot domain where both agents are in a hallway and want to move into the same room through a single doorway. The agents have an operator to go through the door (G) that only succeeds if the other agent is not also trying to go through the door. They also have the choice of waiting (W). Each agent’s goal is simply to be in the room.

2.2 The Formalization

We first begin by formalizing some planning related concepts. The definitions parallel closely with Cimatti and colleagues’ single-agent formalization [2000]. We extend their definitions of planning domains, problems, and solutions to encompass multiple agents. We then follow this formal framework with a definition of multiagent planning equilibrium. Notice that the definitions and concepts presented are not bound to any particular planning algorithm or language.

We start by defining a multiagent planning domain.

Definition 1 (Multiagent Planning Domain)

A multiagent planning domain D is a tuple $\langle \mathcal{P}, \mathcal{S}, n, \mathcal{A}_{i=1\dots n}, \mathcal{R} \rangle$ where,

- $\mathcal{P}$ is the finite set of propositions,
- $\mathcal{S} \subseteq 2^{\mathcal{P}}$ is the set of valid states,
- n is the number of agents,
- $\mathcal{A}_i$ is agent i ’s finite set of actions, and
- $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ is a nondeterministic transition relation where $\mathcal{A} = \mathcal{A}_1 \times \dots \times \mathcal{A}_n$ and must satisfy the following condition. If $\langle s, a, s' \rangle \in \mathcal{R}$ and $\langle s, b, s'' \rangle \in \mathcal{R}$ then, $\forall i$ there exists $s''' \in \mathcal{S}$,

$$\langle s, \langle a_1, \dots, a_{i-1}, b_i, a_{i+1}, \dots, a_n \rangle, s''' \rangle \in \mathcal{R}.$$

I.e., each agent’s set of actions that can be executed from a state are independent.

In addition, let $\text{ACT}_i(s) \subseteq \mathcal{A}_i$ be the set of applicable or executable actions in state s . Formally,

$$\text{ACT}_i(s) = \{a_i \in \mathcal{A}_i \mid \exists \langle s, \langle \dots, a_i, \dots \rangle, \cdot \rangle \in \mathcal{R}\}.$$

The additional condition in the planning domain definition on $\mathcal{R}$ requires that each agent be capable of selecting actions independently. Formally this amounts to the following. For all states s and executable actions for the agents $a_i \in \text{ACT}_i(s)$ there exists some transition $\langle s, \langle a_{i=1\dots n} \rangle, s' \rangle$ that is in $\mathcal{R}$.

In the doorway domain, $\mathcal{P}$ contains two propositions, *A-in-room* and *B-in-room*. The set of states $\mathcal{S}$ corresponds to all four possible subsets of $\mathcal{P}$, since all combinations of propositions are valid in this domain. n is two and $\mathcal{A}_{A,B}$ is the set of actions $\{G, W\}$. The transition relation $\mathcal{R}$ is defined by the rules described above. The complete enumeration of states and transitions is shown in Figure 1. The figure also numbers the states so they can be referred to in an abbreviated

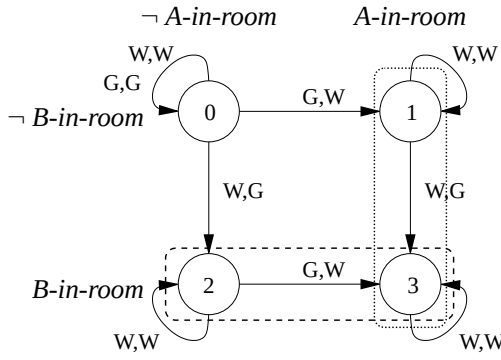


Figure 1: Doorway domain. $\bigcirc$ and $\bigcirc$ represent A's and B's goal states, $\mathcal{G}_A$, and $\mathcal{G}_B$, respectively.

form. Note that this domain satisfies the independent action condition on $\mathcal{R}$.

Definition 2 (Multiagent Planning Problem)

Let $\mathcal{D} = \langle \mathcal{P}, \mathcal{S}, n, \mathcal{A}_{i=1\dots n}, \mathcal{R} \rangle$ be a multiagent planning domain. A multiagent planning problem P for $\mathcal{D}$ is a tuple $\langle \mathcal{D}, \mathcal{I}, \mathcal{G}_{i=1\dots n} \rangle$, where $\mathcal{I} \subseteq \mathcal{S}$ is the set of possible initial states and $\mathcal{G}_i \subseteq \mathcal{S}$ is the set of goal states for agent i .

In the doorway example, the goal states for agent A are $\{1, 3\}$ and for agent B are $\{2, 3\}$. The initial state set is the singular set $\{0\}$. With this definition of domain and problem, we can now formalize a notion of a plan.

Definition 3 (State-Action Table)

A state-action table π_i for agent i in domain $\mathcal{D}$ is a set of pairs $\{ \langle s, a_i \rangle \mid s \in \mathcal{S}, a_i \in \text{ACT}_i(s) \}$. A joint state-action table π constructed from state-action tables for each agent $\pi_{i=1\dots n}$ is the set of pairs

$$\{ \langle s, \langle a_1, \dots, a_n \rangle \rangle \mid s \in \mathcal{S}, \langle s, a_i \rangle \in \pi_i \}$$

A (joint) state-action table is complete if and only if for any $s \in \mathcal{S}$ there exists some pair $\langle s, \cdot \rangle$ in the state-action table.

For the doorway domain, a state-action table (or plan) for each agent might be,

$$\begin{aligned} \pi_A &= \{ \langle 0, G \rangle, \langle 1, W \rangle, \langle 2, G \rangle, \langle 2, W \rangle, \langle 3, W \rangle \}, \\ \pi_B &= \{ \langle 0, G \rangle, \langle 0, W \rangle, \langle 1, G \rangle, \langle 2, W \rangle, \langle 3, W \rangle \}. \end{aligned}$$

These are also *complete* state-action tables since they specify at least one action for each state. We can combine these tables into a complete *joint* state-action table. In general, a joint state-action table together with a multiagent planning domain determines the entire execution of the system. In order to define what it means for a plan to be a solution to a planning problem we need to formalize the notion of reachability and paths of execution. We will do this by first defining the execution structure of the multiagent system.

Definition 4 (Induced Execution Structure)

Let π be a joint state-action table of a multiagent planning domain $\mathcal{D} = \langle \mathcal{P}, \mathcal{S}, n, \mathcal{A}_i, \mathcal{R} \rangle$. The execution structure induced by π from the set of initial states $\mathcal{I} \subseteq \mathcal{S}$ is a tuple $K = \langle Q, T \rangle$ with $Q \subseteq \mathcal{S}$ and $T \subseteq \mathcal{S} \times \mathcal{S}$ inductively defined as follows:

- if $s \in \mathcal{I}$, then $s \in Q$, and
- if $s \in Q$ and there exists a state-action pair $\langle s, a \rangle \in \pi$ and transition $\langle s, a, s' \rangle \in \mathcal{R}$, then $s' \in Q$ and $\langle s, s' \rangle \in T$.

A state $s \in Q$ is a terminal state of K if and only if there is no $s' \in Q$ such that $\langle s, s' \rangle \in T$.

Intuitively, Q is the set of states that the system could reach during execution of the plan π , and T is the set of transitions that the system could cross during execution. For our doorway domain the execution structure induced by our example joint state-action table is,

$$\begin{aligned} Q &= \{0, 1, 3\}, \\ T &= \{ \langle 0, 1 \rangle, \langle 0, 0 \rangle, \langle 1, 3 \rangle, \langle 1, 1 \rangle, \langle 3, 3 \rangle \}. \end{aligned}$$

We can now formalize an execution path.

Definition 5 (Execution Path)

Let $K = \langle Q, T \rangle$ be the execution structure induced by a state-action table π from $\mathcal{I}$. An execution path of K from $s_0 \in \mathcal{I}$ is a possibly infinite sequence $s_0, s_1, s_2, \dots$ of states in Q such that, for all states s_i in the sequence:

- either s_i is the last state of the sequence, in which case s_i is a terminal state of K , or
- $\langle s_i, s_{i+1} \rangle \in T$.

A state s' is reachable from a state s if and only if there is an execution path with $s_0 = s$ and $s_i = s'$.

For our doorway domain and example joint state-action table one execution path from the initial state is,

$$0, 0, 0, 0, 1, 1, \dots$$

We are now in a position to define what it means for our plan to solve a planning problem. We actually define multiple concepts increasing in strength. These concepts formalize some of the intuitive discussion from the previous section about whether a plan has one or more of the following properties:

- the possibility of reaching the goal,
- a guarantee of reaching the goal, and
- a guarantee of reaching the goal in a finite number of steps.

These concepts and their formalization are inspired and highly related to Cimatti and colleagues' single-agent solution concepts [Cimatti *et al.*, 2000].

Definition 6 (Multiagent Planning Solutions)

Let $\mathcal{D}$ be a multiagent planning domain and $P = \langle \mathcal{D}, \mathcal{I}, \mathcal{G}_{i=1\dots n} \rangle$ be a multiagent planning problem. Let π be a complete joint state-action table for $\mathcal{D}$. Let $K = \langle Q, T \rangle$ be the execution structure induced by π from $\mathcal{I}$. The following is an ordered list of solution concepts increasing in strength.

1. π is a weak solution for agent i if and only if for any state in $\mathcal{I}$ some state in $\mathcal{G}_i$ is reachable.
2. π is a strong cyclic solution for agent i if and only if from any state in Q some state in $\mathcal{G}_i$ is reachable.
3. π is a strong solution for agent i if and only if all execution paths, including infinite length paths, from a state in Q contain a state in $\mathcal{G}_i$.

4. π is a perfect solution for agent i if and only if for all execution paths $s_0, s_1, s_2 \dots$ from a state in Q there exists some $n \geq 0$ such that $\forall i \geq n, s_i \in \mathcal{G}_i$.

A state-action table's strength $\text{STRENGTH}(\mathcal{D}, \mathcal{P}, i, \pi)$ is the largest number whose condition above applies for agent i . If no conditions apply then $\text{STRENGTH}(\mathcal{D}, \mathcal{P}, i, \pi) = 0$.

For our doorway domain, the joint state-action table is a strong cyclic solution for both agents but not strong (i.e., it has a strength of 2 for both agents). This means that there is a path to the goal from any reachable state. But there are also paths that do not include either agents' goal states, and so it is not a strong solution for either agent.

These solutions define what it means for one agent to be successful given a joint state-action table. The goal of planning from one agent's perspective is to find a plan that has the highest strength given the plans of the other agents. But the other agents' selection of a plan is equally contingent upon the first agent's plan. This recursive dependency leads to our main contribution of the paper: planning equilibria.

Definition 7 (Multiagent Planning Equilibria)

Let $\mathcal{D}$ be a multiagent planning domain and $\mathcal{P} = \langle \mathcal{D}, \mathcal{I}, \mathcal{G}_{i=1 \dots n} \rangle$ be a multiagent planning problem. Let π be a complete joint state-action table for $\mathcal{D}$. Let $K = \langle Q, T \rangle$ be the execution structure induced by π from $\mathcal{I}$. π is an equilibrium solution to $\mathcal{P}$ if and only if for all agents i and for any complete joint state-action table π' such that $\pi'_{j \neq i} = \pi_j$,

$$\text{STRENGTH}(\mathcal{D}, \mathcal{P}, i, \pi) \geq \text{STRENGTH}(\mathcal{D}, \mathcal{P}, i, \pi').$$

I.e., each agent's state-action table attains the strongest solution concept possible given the state-action tables of the other agents.

Note that our example joint state-action table for the doorway domain is *not* an equilibrium. Both agents A and B currently have strength 2, but B can achieve a strength of 4 by choosing a different state-action table. Specifically, B should select the wait (W) action from the initial state and the go (G) action in state 1.

To make the concept of planning equilibria clearer, we illustrate it in the doorway domain. We gave above an example joint state-action table that is not a multiagent planning equilibrium for this domain. An equilibrium is the following state-action tables:

$$\begin{aligned} \pi_A &= \{ \langle 0, G \rangle, \langle 1, W \rangle, \langle 2, G \rangle, \langle 3, W \rangle \}, \\ \pi_B &= \{ \langle 0, W \rangle, \langle 1, G \rangle, \langle 2, W \rangle, \langle 3, W \rangle \}. \end{aligned}$$

In this case agent A goes through the door while agent B waits and then follows through the door. This is a perfect plan for both agents and so obviously no agent can achieve a higher strength with a different state-action table. Similarly, the symmetric tables where agent B goes through the door while agent A waits is also an equilibrium. There is an additional equilibrium,

$$\begin{aligned} \pi_A &= \{ \langle 0, G \rangle, \langle 0, W \rangle, \langle 1, W \rangle, \langle 2, G \rangle, \langle 3, W \rangle \}, \\ \pi_B &= \{ \langle 0, G \rangle, \langle 0, W \rangle, \langle 1, G \rangle, \langle 2, W \rangle, \langle 3, W \rangle \}. \end{aligned}$$

Here both agents nondeterministically decide between going through the door and waiting. This results in a strong cyclic

solution for both agents, but given this state-action table for the other agent no strong or perfect plan exists for either agent. So this is also an equilibrium although obviously inferior to the other equilibria where both agents have higher strength plans. In game theory, such a joint strategy is called Pareto dominated.

Collision variation. Consider a variation on this domain where collisions (when both agents choose G) result in the robots becoming damaged and unable to move. In this case, the first two state-action tables above remain equilibria, but the third inferior table no longer is an equilibrium. This joint plan is now only a weak solution for both agents since there is a possibility of never achieving the goal. Each agent can also change to a different plan where it waits for the other agent to get through the door thus achieving a strong cyclic plan and a higher strength.

Door closing variation. Finally, consider that one agent entering the room sometimes causes the door to close behind it. Once the door is closed it cannot be opened and the doorway cannot be used. In this case, the same two joint plans are again an equilibrium but now they have different strengths for the different agents. The first joint state-action table is a strong plan for agent A, but only a weak plan for agent B, though it can do no better. The second is just a symmetry of this.

3 Conclusion

We presented a formalization of multiagent planning and introduced the concept of a multiagent planning equilibrium. This is the first known solution concept that explicitly accounts for the goals of all the agents. This work provides a unifying framework for considering planning in multiagent domains with identical, competing, or overlapping goals. It also opens up many exciting questions related to practical algorithms for finding equilibria, the existence of equilibria, and the coordination of equilibria selection.

Acknowledgments

This research was sponsored by grant No. F30602-00-2-0549. The content of this publication does not necessarily reflect the position of the funding agencies and no official endorsement should be inferred.

References

- [Cimatti *et al.*, 2000] Alessandro Cimatti, Marco Pistore, Marco Roveri, , and Paolo Traverso. Weak, strong, and strong cyclic planning via symbolic model checking. IRST Technical report 0104-11, Istituto Trentino di Cultura, April 2000.
- [Jensen and Veloso, 2000] Rune Jensen and Manuela Veloso. OBDD-based universal planning for synchronized agents in non-deterministic domains. *Journal of Artificial Intelligence Research*, 13:189–226, 2000.
- [Owen, 1995] Guillermo Owen. *Game Theory*. Academic Press, 1995.

Bidding Marginal Utility in Simultaneous Auctions

Amy Greenwald

Department of Computer Science
Brown University, Box 1910
Providence, RI 02912
amy@cs.brown.edu

One of the key challenges autonomous bidding agents face is to determine how to bid on complementary and substitutable goods—i.e., goods with combinatorial valuations—in simultaneous, or parallel, auctions (SAs). Complementary goods are goods with super-additive valuations: $v(AB) + v(\bar{A}\bar{B}) < v(AB)$. Substitutable goods are goods with subadditive valuations: $v(\bar{A}\bar{B}) + v(AB) > v(AB)$. The simple bidding strategy “for each good x in auction x , bid its (independent) valuation” is inapplicable in this framework.

Rather than attempt to reason about the independent valuations of goods, bidding agents that operate in this setting can reason about *marginal* valuations, or the valuation of a good x relative to a set of goods X . If an agent holds the goods in X , it can ask questions such as: “what is the marginal benefit of buying x ?” or, “what is the marginal cost of selling x ?” To do so, the agent must reason about the *set* of goods $X \cup \{x\}$ or $X \setminus \{x\}$, relative to the set X —the valuations of which are well-defined.

In reasoning about sets of goods bidding agents may pose and solve questions such as the following [Greenwald and Boyan, 2001]:

1. “Given only the set of goods I already hold, what is the maximum valuation I can attain, by arranging my individual goods into sets of goods?”
2. “Given the set of goods I already hold, and given market prices and *supply*, on what set of additional goods should I place *bids* so as to maximize my utility: i.e., valuation less purchase costs?”
3. “Given the set of goods I already hold, and given market prices, supply, and *demand*, on what set of goods should I place bids or *asks* so as to maximize my utility: i.e., valuation plus proceeds less costs?”

The third, and most general, of these problems, which we call *completion*, provides a foundation for bidding strategies in simultaneous single- and double-sided auctions. The second problem, which we term *acquisition*, provides a foundation for bidding strategies in single-sided auctions. The first problem, *allocation*, is a special case of the others in which all goods cost either 0 or ∞ , and all goods are worth either 0 or $-\infty$. These so-called bid determination problems are formally defined in [Greenwald and Boyan, 2001], where it is argued that completion can be reduced to acquisition.

1 Bidding Principles

This paper advocates the following bidding principle for agents participating in simultaneous auctions for complementary and substitutable goods: “coherent” marginal utility bidding: i.e., bidding marginal utilities on a coherent subset of goods. To validate this principle, we compare two combinatorial bidding strategies inspired by ATTAC [Stone *et al.*, 2002] and ROXYBOT [Greenwald and Boyan, 2001], two agents in the Trading Agent Competition (TAC) (see [Wellman *et al.*, 2002]):

1. one strategy computes the marginal utility of each good independently, much like ATTAC
2. one strategy computes coherent marginal utilities—it solves the acquisition problem before bidding marginal utilities—much like ROXYBOT

It is established that approach 2 outperforms approach 1 in certain environments. In a longer version of this paper, we experiment with stochastic versions of these algorithms, and establish that approach 2 also outperforms approach 1 in uncertain environments. The insights gained from this study about the design of autonomous bidding agents are applicable beyond the scope of TAC: e.g., eBay is home to numerous SAs for complementary and substitutable goods.

2 Bidding Under Certainty

In constructing bidding policies—mappings from goods to prices—for multi-unit SAs, such as those that characterize TAC, we propose the following natural breakdown of an agent’s bidding decisions:

1. how many copies of each good do I want?
2. for the goods I want, how much am I willing to pay?

One straightforward approach to answering these questions, which was employed by ATTAC [Stone *et al.*, 2002], is for the agent to skip question 1, and simply compute how much it is willing to pay for each copy of each good. An alternative approach, employed by ROXYBOT, is to explicitly answer question 1 before question 2; in this way, the agent is certain to bid on a “coherent” set of goods: i.e., a set of goods which together comprise an optimal solution to the acquisition problem.

In this short paper, we compare these two approaches under two assumptions: (i) prices are exogenously determined: *i.e.*, we ignore the impact of agent behaviors on prices; and (ii) prices are known with certainty. These assumptions are clearly applicable when prices are posted. In an auction setting, we interpret these assumptions as follows: the payment rule is “pay the known price;” the winner determination rule is “win by bidding equal to or more than the price.” It is as if an agent is capable of perfectly predicting all other agents’ bids, as well as the impact of its own bids on clearing prices.

We begin by investigating ATTAC’s approach. One formula for determining willingness to pay is to compute marginal utilities (MUs). This direct MU approach to answering the aforementioned bidding questions explicitly answers only question 2, but implicitly answers question 1: a willingness to pay 0 suggests that an agent wants 0 additional copies of that good.

As alluded to earlier, computing MUs depends on solving the acquisition problem. Let us introduce some notation, and formally define acquisition and MU. Let X denote a set of goods; let $v(X)$ denote the (combinatorial) valuation of X ; let $p(x)$ denote the price of $x \in X$ and $p(X) = \sum_{x \in X} p(x)$; finally, let $u(X) = v(X) - p(X)$ denote the utility of X .

Definition 2.1 Given a set of goods X , a combinatorial valuation function $v : 2^X \rightarrow \mathbb{R}^+$, and an exogenous pricing mechanism $p : X \rightarrow \mathbb{R}^+$, an optimal solution to the acquisition problem is a subset $A^* \subseteq X$ *s.t.*

$$A^* \in \arg \max_{Y \subseteq X} v(Y) - p(Y) \quad (1)$$

Definition 2.2 Given a set of goods X , the marginal utility of good x is defined as follows: $\mu(x|X) = u(X^*) - u(Y^*)$, where

$$X^* \in \arg \max_{Y \subseteq X \cup \{x\}} v(Y) - p(Y \setminus \{x\}) \quad (2)$$

and

$$Y^* \in \arg \max_{Y \subseteq X \setminus \{x\}} v(Y) - p(Y) \quad (3)$$

In words, the marginal utility of good x , relative to the set X , is simply the difference between the utility of X , assuming x costs 0, $u(X|p(x) = 0)$, and the utility of X , assuming x costs ∞ , $u(X|p(x) = \infty)$.

As is evident from the definition of MU, computing marginal utilities requires two calls to a combinatorial optimization solver. If an agent explicitly answers question 1 before question 2, it might determine that it is not necessary to compute marginal utilities for all copies of all goods (although in the worst case it solves one additional optimization problem). ROXYBOT explicitly answers question 1, before question 2, and usually achieves computational savings over direct MU calculation. But the real motivation behind ROXYBOT’s approach is that reasoning about individual goods, rather than coherent sets of goods, when valuations are combinatorial, is suboptimal. In the following example, ROXYBOT’s approach is optimal, but the direct MU approach is not.

Example 2.3 Given goods x, y, z , with combinatorial valuations as follows: $v(xyz) = v(xy) = v(yz) = 500$ and $v(x) = v(y) = v(z) = v(xz) = 0$. Each good in isolation is of no value, but together with y , either x or z is worth 500. Assume all goods are priced equivalently at 100. The optimal sets of goods in this example are xy and yz , each of which yields utility of 300.

Now let us compute marginal utilities: $\mu(x|xyz) = u(xyz|p(x) = 0) - u(xyz|p(x) = \infty) = 400 - 300 = 100$; $\mu(y|xyz) = u(xyz|p(y) = 0) - u(xyz|p(y) = \infty) = 400 - 0 = 400$; and $\mu(z|xyz) = u(xyz|p(z) = 0) - u(xyz|p(z) = \infty) = 400 - 300 = 100$. Thus, a MU bidder bids on all three goods, and wins all three goods, obtaining utility $v(xyz) - (p(x) + p(y) + p(z)) = 200$.

A bidder that first determines the set of goods it wants bids on either x and y , or y and z , but not both. Computing marginal utilities for (say) x and y yields: $\mu(x|xy) = u(xy|p(x) = 0) - u(xy|p(x) = \infty) = 400 - 0 = 400$ and $\mu(y|xy) = u(xy|p(y) = 0) - u(xy|p(y) = \infty) = 400 - 0 = 400$. This bidder earns the optimal utility $v(xy) - (p(x) + p(y)) = 300$. $\square$

The main observations reported in this paper are the following: (i) for all goods x in an optimal acquisition, the price of x does not exceed the marginal utility of x ; (ii) for all goods x *not* in an optimal acquisition, the marginal utility of x does not exceed the price of x . It follows that bidding marginal utilities on precisely those goods in an optimal acquisition is an optimal bidding policy, if prices are exogenous (*e.g.*, an agent’s individual bidding behavior does not impact prices), and prices are known with certainty.

Proposition 2.4 If $A^* \subseteq X$ is an optimal solution to the acquisition problem, then $\mu(x|A^*) \geq p(x)$, $\forall x \in A^*$, and $\mu(x|A^*) \leq p(x)$, $\forall x \notin A^*$.

ROXYBOT’s strategy answers questions 1 and 2 in turn, and bids marginal utilities on all the goods it wants. This strategy is optimal, under our assumptions, by the previous observation. In particular, ROXYBOT wins all the goods it wants by bidding marginal utilities. Of course, bidding $p(x)$, or ∞ , $\forall x \in A^*$, are also optimal policies in this setting. But bidding the marginal utility of good x is a reasonable heuristic, since it is in fact an optimal bidding policy if the prices of all goods other than x are exogenous and certain.

Acknowledgments

This research was supported by NSF Career Grant #IIS-0133689. The author also acknowledges Maureen Hurtgen and the funding of CRA’s DMP.

References

- [Greenwald and Boyan, 2001] Amy Greenwald and Justin Boyan. Bidding algorithms for simultaneous auctions: A case study. In *Proceedings of Third ACM Conference on Electronic Commerce*, 115-124 2001.
- [Stone et al., 2002] Peter Stone, Robert E. Schapire, János A. Csirik, Michael L. Littman, and David McAllester. ATTAC-01: A learning, autonomous bidding agent. In *Workshop on Agent Mediated Electronic CommerceIV: Designing Mechanisms and Systems*, 2002.
- [Wellman et al., 2002] Michael P. Wellman, Amy Greenwald, Peter Stone, and Peter R. Wurman. The 2001 Trading Agent Competition. In *Proceedings of the Fourteenth Innovative Applications of Artificial Intelligence Conference*, pages 935-941, July 2002.

NoA – A Normative Agent Architecture

Martin J. Kollingbaum and Timothy J. Norman

Department of Computing Science
University of Aberdeen
Aberdeen, AB24 3UE, UK
mkolling@csd.abdn.ac.uk
tnorman@csd.abdn.ac.uk

Abstract

NoA is an agent architecture that supports the development of agents motivated by norms: obligations, permissions and prohibitions. Obligations motivate a normative agent to act: a motive to achieve a state of affairs or to perform some action. Prohibitions restrict an agent's behaviour, whereas permissions allow an agent to pursue certain activities. To test the architecture, NoA agents are applied to automated business transaction scenarios where the correct execution of contracts is paramount to create a situation of trust.

1 Introduction

The NoA normative agent architecture is designed to support the development of agent societies where agents are motivated by norms: obligations, permissions and prohibitions. The development of the NoA agent architecture is driven by the particular needs of automated business transactions within electronic commerce environments. However, NoA is designed as an architecture that is generally applicable for agent system development. The specific concern of the NoA architecture is to provide means for the development of norm-motivated practical reasoning agents. This architecture is determined by two main elements: The NoA language for the specification of plans and norms and the NoA interpreter, which is capable of interpreting and executing plan and norm specifications formulated in the NoA language. The NoA architecture and the NoA specification language for norms and plans is influenced by systems such as Agentspeak(L) [Rao, 1996]. There are three principal distinctions, however, between these systems and NoA:

- *Multiple effects.* In common with planning domain specification languages such as PDDL [PDDL], but in contrast to languages such as JAM [Huber, 1999], the NoA plan specification language allows *all* the effects of a plan to be declared. Any of these effects can be the reason for the agent to select a plan. This provides greater flexibility in the specification of agent

capabilities, and enables a NoA agent to reason about the side-effects of executing a plan.

- *Motivated by norms.* NoA agents are motivated by norms rather than desires and intentions. These norms capture states of affairs or actions the agent is obliged or permitted to achieve or perform, or prohibited from achieving or performing.
- *Distinction between states and actions.* The norms governing the behaviour of a NoA agent refer to either actions that are obligatory, permitted, forbidden, or states of affairs that are obligatory, permitted or forbidden [Norman and Reed, 2001].

2 The NoA Language

The main elements that influence the behaviour of a NoA agent are (a) a set of beliefs, (b) a set of pre-specified plans and (c) a set of norms. These concepts are reflected in the NoA specification language. The general form of a NoA plan carries information about when it would be allowed to select it for execution (preconditions), what kind of states of affairs it can achieve (effects) and the actual behaviour specification as the body of the plan.

```
plan stack ( X, Y )  
  preconditions ( ontable ( X ), clear ( Y ) )  
  effects ( on ( X, Y ), not ontable ( X ),  
           not clear ( Y ) )  
{  
  achieve clear ( X ) ;  
  primitive doMove ( X, Y ) ;  
}
```

Figure 1. A NoA Plan

Figure 1 shows an example plan, usually applicable in a blocks world scenario for stacking blocks. NoA plans are chosen for achieving a state of affairs according to their effects (see the effects list in figure 1). For the performance of an action, plans are chosen according to their signature (name and parameter list). The plan body in this example shows language constructs to establish a subgoal (achieve clear(X)) and to perform primitive actions.

```

obligation ( BlockMover,
              achieve ontable( a ),
              on ( a, b ),
              clear ( b ) and ontable( a )
            )

```

Figure 2. A NoA norm specification

Figure 2 shows a norm specification: an obligation for agent BlockMover to achieve the placement of a block *a* on the table (again in an imagined blocks world). Each normative statements carries information under what circumstances such a norm is “active” and therefore relevant to an agent. Normative statements include activation and expiration conditions that determine when a norm is active and when it expires.

3 The NoA Interpreter

The behaviour of a NoA agent is determined by norms and plans. Norms are relevant to an agent only if they are “active”. Norms become active when the agent’s beliefs reflect their specified conditions of activation. Activated norms influence the behaviour of an agent in two ways, (a) by motivating the generation of a goal that must be achieved or an action that has to be performed, these are externally motivated activities, and (b) by being used in a special “norm filtering” step within the execution cycle of the NoA architecture by restricting the options of the agent in fulfilling its responsibilities.

Plans are instantiated when their preconditions hold. From this set of plan instances, a set of plan options has to be selected. This selection takes place according to one of their effects, whereas all other effects represent side-effects of this plan, which occur during its execution. All the effects of a plan must be taken into account when it is considered for execution, e.g. a plan selected for satisfaction of one obligation may have a side-effect that is contrasting other norms.

```

while state / action requested
  if achievement of state is requested
    then plan_options = choose_plan_instances (state)
    else plan_options = retrieve_plan (action)
  endif
  filtered_plans = norm_filter (plan_options)
  if filtered_plans not empty then
    plan = deliberate (filtered_plans)
    execute (plan)
  endif
endwhile

```

Figure 3. Execution cycle of the NoA interpreter

The execution cycle of the NoA interpreter is shown in figure 3. It is assumed that a change in the agent’s beliefs activates / de-activates (a) obligations that motivate the achievement of a state of affairs or the performance of an action, (b) prohibitions and permissions relevant for the norm filtering step, and (c) instantiates plans. In case that

a state of affairs must be achieved, a set of plan instances is chosen according to one of the effects they would produce if they would be executed (see figure 3). From the set of currently instantiated plans (because their preconditions hold) the set of plan options is chosen. In case that the direct performance of an action is required, one specific plan instance is chosen. Active norms such as prohibitions and permissions are applied to this set of plans to filter out those that would produce effects that are forbidden or are directly acts that are forbidden. One of the plans must be chosen for execution. The execution of a plan can have various outcomes: performing a primitive action affecting the world, updating beliefs or establishing a subgoal or subsidiary actions.

5 Conclusion

In this paper, the NoA normative agent architecture has been presented as a framework that supports the development of agent societies where agents are motivated by norms. This architecture introduces norms as external motivators for an agent to achieve states of affairs or perform actions. It provides agents with capabilities of decision-making driven by such norms expressing the obligations, permissions and prohibitions of an agent. The NoA normative agent architecture is based on three key concepts: (a) a clear distinction between an agent taking responsibility for the achievement of a state of affairs and the agent taking responsibility for the performance of an action, (b) agents understand normative statements within contracts and how to correctly execute them and (c) goal / action generation and plan selection is based on norms.

References

- [Kollingbaum and Norman, 2002] Martin J. Kollingbaum and Timothy J. Norman. Supervised Interaction – creating a Web of Trust for Contracting Agents in Electronic Environments. In C. Castelfranchi & W. Johnson (eds), AAMAS 2002, ACM Press, New York, pages 272-279, 2002.
- [Georgeff and Lansky, 1987] Michael P. Georgeff and Amy L. Lansky. Reactive Reasoning and Planning. In *Proceedings of AAAI-87*, pp. 677-682, 1987.
- [Huber, 1999] Marcus Huber. JAM: A BDI-theoretic mobile agent architecture. *Agents'99*, pp. 236--243, (May 1999).
- [Rao, 1996] Anand S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. *LNAI 1038*, pp. 42--55. Springer, (1996).
- [Norman and Reed, 2001] Timothy J. Norman and Chris Reed. Delegation and Responsibility. In C. Castelfranchi, Y. Lesperance (eds), *Intelligent Agents VII: Proceedings ATAL 2001*, LNAI 1986, pp. 136-149, 2001.
- [PDDL] <http://www.dur.ac.uk/d.p.long/IPC/pddl.html>

A heuristic model for concurrent bi-lateral negotiations in incomplete information settings

Thuc Duong Nguyen and Nicholas R. Jennings*

Abstract

Bi-lateral negotiations represent an important class of encounter in agent-based systems. To this end, this paper develops and evaluates a heuristic model that enables an agent to participate in multiple, concurrent bi-lateral encounters in competitive situations in which there is information uncertainty and deadlines.

1 Introduction

Automated negotiation is a key form of interaction in agent-based systems. Such negotiations exist in many different forms including one-to-one, one-to-many and many-to-many. Generally speaking, however, the latter two cases are dealt with using some form of auction protocol (be it single-sided or double-sided, respectively), while the former are often tackled using some form of heuristic method. Here we focus on the one-to-one case, in which one agent wants to purchase a service¹ from another. Moreover, we consider competitive situations in which the agents have *no a priori knowledge about the preferences of their opponents*. In such cases, the agents exchange proposals, representing acceptable solutions, until either an agreement is reached or the negotiation terminates with a failure.

To date, one of the inherent drawbacks of bi-lateral negotiation models is that the agent has to a priori identify a single partner to interact with. However, this is inefficient in an uncertain setting where there are multiple providers of the service that each have different characteristics. In this case, there are two alternatives: (1) negotiate sequentially with all the providers or (2) negotiate concurrently with them. The former has the disadvantage that it may result in lengthy negotiation encounters, but has the advantage that it is comparatively easy to use the outcome of one negotiation to dictate behavior in subsequent ones. The latter case has the advantage of taking less time, but the disadvantage that coordinating behaviors among the various negotiation threads is more difficult.

*Department of Electronics and Computer Science, University of Southampton, UK. Email: {tdn01r, nrj}@ecs.soton.ac.uk

¹A service is here viewed as an abstract representation of the capability of an agent.

Since we are interested in situations in time-constrained domains (such as e-commerce and grid computing), we concentrate on the concurrent case and develop a coordinated bidding model in which the various negotiation threads *mutually influence* one another. By mutually influence, we mean that the progress and agreement in one negotiation thread is used to alter the behavior of the agent in another thread for the same service. For example, having obtained a good deal in one thread, the agent may adopt a tougher stance in its other threads, to see if it can get an even better deal than the one it already has².

2 The concurrent negotiation model

The agent that wishes to purchase the service is called the *buyer* and the agents that offer the service are called the *sellers*. Service agreements (contracts) are assumed to be multi-dimensional (covering issues such as *price, quality, quantity*, etc.). The buyer has a deadline t_{max} by when it must conclude its negotiations for the service. Similarly, each seller i has its own negotiation deadline $t_{i,max}$. All the agents have their own preferences about the service and this information is private (as are the strategies the individual agents follow). The agents follow an alternating sequential protocol, in which the illocutions are *offer* (a proposal made by one agent to the other), *counter-offer* (a counter proposal from an agent in response to a proposal it received), *accept* (accept a proposed offer), *finalize* (secure a deal with the chosen seller), *decline* (reject the previously accepted offer) and *withdraw* (terminate the negotiation thread). The difference between an *accept* and *finalize* is necessary in this work to deal with the problem of concurrent encounters. If the buyer accepts an offer from a seller then this is viewed as binding on the seller (for a specified period of time that is assumed to be longer than t_{max}). However, it is not binding on the buyer. Thus, the buyer may accept several offers from multiple sellers in any one negotiation episode. However, when it has completed all the negotiations, the buyer will *finalize* one of the accepted deals with one of the sellers and *decline* the others (thus freeing them

²This model differs from a one-to-many auction in that it allows *direct* interaction between the agent requiring the service and the providers offering it. This ability to exchange unmediated counter-offers enables the participants to indicate their preferences and constraints directly to one another.

from their commitment to the proposal). This two phase process is necessary so that the buyer can use accepted deals as a base line for the subsequent concurrent negotiations.³

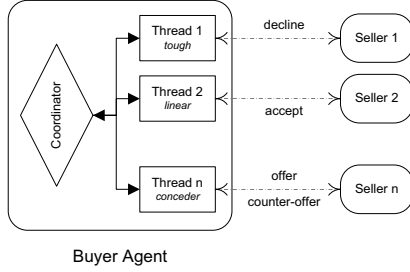


Figure 1: System architecture

In more detail, the model for the buyer agent consists of two main components: a *coordinator* and a number of *negotiation threads* (see figure 1). The negotiation threads deal directly with the various sellers (one per seller) and are responsible for deciding what counter-offers to send to them and what proposals to accept. For maximum flexibility, we assume that the buyer agent may adopt different strategies in each of its threads. We adopt separate semi-independent threads for reasons of modularity and coherence. The alternative of having every single negotiation move centrally coordinated and intercepting all the bids received from all the sellers, is viewed as a computational bottleneck for the time-constrained environments we are targeting.

Each negotiation thread inherits the preferences from the main buyer agent, including the acceptable ranges of values for each negotiation issue, the deadline of the negotiation and the current reservation value (the lowest utility value of an offer that the agent considers acceptable). The coordinator decides the negotiation strategies for each thread. After each negotiation round, the threads report back their status to the coordinator. If a thread reaches a deal with a particular seller, it terminates its negotiation. Based on the coordination schema it is using (see section 2.1 for more details), the coordinator will then notify all other negotiation threads of the new reservation value and it may change the negotiation strategy of some of them. The detailed working of the two components are described below.

2.1 The coordinator

The coordinator is responsible for coordinating all the negotiation threads and choosing an appropriate negotiation strategy for each thread.

As a first step, the coordinator acts like a blackboard for shared information about the ongoing encounters. It receives the current status from the various negotiation threads (including the proposal's values), keeps a list of agreements reached and notifies the threads about any changes in their reservation values. Second, and more importantly, the coor-

dinator decides the negotiation strategy for each thread initially and whether this should change over time to reflect any agreements that have been made to date.

To ground our model, at this time we consider the set of strategies S to be composed of the class of time dependent strategies advocated in [Faratin, 2001] for bi-lateral negotiations in uncertain environments with time constraints. These strategies fall into three categories, namely: *conceder* (S_c), *linear* (S_l) and *tough* (S_t) where $S = S_c \cup S_l \cup S_t$. All of the strategies start with the same initial value that is generated in relation to the deadline and the reservation value. The *conceder* strategy quickly lowers its value until it reaches its reservation value. The *linear* strategy drops to its reservation value in a steady fashion. Finally, the *tough* strategy keeps its value until the deadline approaches, then it rapidly drops to its reservation value.

In his empirical analysis of the behavior of negotiating agents that adopted these strategies, Faratin showed that if it is possible to approximate the type of the opponent then the agent can alter its strategy to be more effective. Given this observation, the coordinator attempts such a classification. Specifically, at time t : $2 < t \leq t_{max}$, called the *analysis time*, the coordinator tries to determine if a given seller is a *conceder* or a *non-conceder*. In particular, assume O_j^i is the value of the offer that seller agent i made at time j : $0 \leq j \leq t$. Then seller i is considered a *conceder* if $\forall k \in [2, t] : \frac{O_k^i - O_{k-1}^i}{O_{k-1}^i - O_{k-2}^i} > \alpha$ where α is the threshold value set on concessionary behavior. There is a similar characterization of non-conceders and if the agent falls into neither category, it is judged not classified.

Let the set of *conceder* and *non-conceder* agents be represented by A^c and A^n , respectively. Now, given the set of strategies $S = \{S_c, S_l, S_t\}$ and the set of agents $A = \{A^c, A^n\}$, the coordinator changes the strategy for each negotiation thread based on the type of the agent it is negotiating with, in order to try and obtain better outcomes. Agents belonging to the set A^c are willing to concede in order to end up with agreements. Therefore, if the agent negotiates toughly with some of them (keeping its offer consistent), it may obtain a deal that has higher value than if it continues negotiating in its present manner. However, if the agent negotiates in this way with all the agents, it may not obtain any deals at all. Therefore, for reasons of balance, the agent will negotiate in a tough manner with a subset of the agents in A^c , specifically with a percentage ($P_t^c\%$) of them. For the remainder of the agents in A^c , the strategy remains unchanged. Similarly, if the agent believes a particular agent is in the set A^n then in order to make sure it obtains a deal with some of them, it makes some of its own strategy more conciliatory. Specifically, for the agents belonging to the set A^n , a fixed percentage ($P_t^n\%$) of them will have their behavior made conciliatory, while the remainder have their strategies unchanged. There is no change to agents whose behavior cannot be classified.

2.2 The negotiation threads

An individual negotiation thread is responsible for dealing with an individual seller agent on behalf of the buyer. Each

³This model is obviously biased in favor of the buyer and future work will look at relaxing this constraint so that sellers can also renegotiate on deals.

such thread inherits its preferences from the buyer agent and has its negotiation strategy specified by the coordinator.

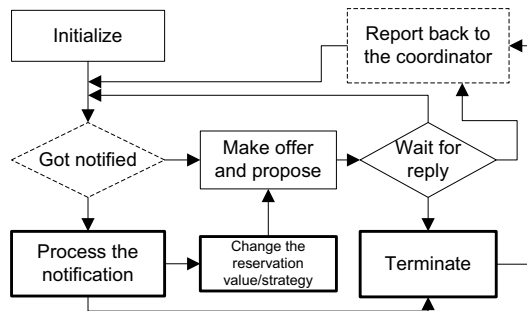


Figure 2: A single negotiation thread

In each thread (see figure 2), there are three main subcomponents; namely *communication* (represented by the dotted lines), *process* (represented by the bold lines) and *strategy*. The *communication* subcomponent is responsible for communicating with the coordinator. Before each round, it checks for incoming messages from the coordinator and if there are any, it passes them to the *process* subcomponent. After each round, it reports the status of the thread (including proposed proposals and the deal's value if an agreement is reached) back to the coordinator. The *process* subcomponent processes messages from the *communication* subcomponent. This can either be changing the reservation value or changing the strategy. The *strategy* subcomponent is responsible for making offers/counter-offers, as well as deciding whether or not to accept the offer made by the seller agent. It uses the reservation value as the basis for deciding whether to accept the seller's offer; in this case any offer with a value greater than this is accepted, otherwise a counter-proposal is made (unless the deadline has passed in which case a decline is sent).

3 Empirical evaluation

Having defined the model, the next step is to evaluate it. Given the aims of our work, we are interested in operational performance and so we decided to evaluate it empirically. In particular, we posed a number of hypotheses and evaluated them in different types of environment.

Our concurrent model is compared against a sequential negotiation model based on [Faratin, 2001]. In this model, all the agents' preferences, as well as the allocation of the strategies, are drawn from the same distributions as the concurrent ones. The only difference with [Faratin, 2001] is that if the buyer agent reaches an agreement of value p in negotiation i , then in all subsequent negotiations, p will be its new reservation value.

We now turn to specific hypotheses. Due to the limitation of space, we cannot show the corresponding graphs to support our claims (see [Nguyen and Jennings, 2003] for more details), but, nevertheless we can summarize our main findings:

Result 1 *The time to complete the negotiation is less for the concurrent model than for the sequential one.*

Result 2 *The number of proposals that are made in the concurrent model is less than the number made in the sequential model.*

Result 3 *To realize the benefits of concurrent negotiations, the buyer agent's deadline must not be too short.*

Result 4 *The final agreements reached by the concurrent model have, on average, higher or equal utility for the buyer than those of the sequential model (assuming the deadline is not too short).*

Result 5 *Changing the strategy in response to the agent's assessment of the ongoing negotiation is equal or better than not doing so.*

Result 6 *To improve the performance of the concurrent model, the analysis time should be moderately early (to have time to have some effect) but not too early (so it is reasonably accurate).*

Result 7 *When dealing with sellers in A^c , the tougher the buyer negotiates the better the overall outcome it obtains.*

4 Conclusions and future work

This paper has developed a heuristic model for managing concurrent negotiations in time-constrained settings where agents have no prior knowledge of their opponents or their types. Through empirical evaluation, we showed how the model leads to better deals, more quickly than its sequential counterpart. We also highlighted the importance of the time when the opponents' negotiation strategies are classified and on the response to this assessment in terms of the degree of toughness adopted.

There are, however, a number of areas that still require further work. First, the means by which negotiation opponents are classified as being conceder or non-conceder needs to be refined so that this monitoring can be made on an ongoing basis (rather than as a one-off decision). Second, we need to allow for the possibility of sellers decommitting and then having these commitment changes feedback into the buyer's negotiation behavior. Third, we wish to extend the implementation so that the coordinator and the negotiation threads also embody fundamentally different models of bi-lateral negotiation (e.g. based on constraint-satisfaction, game-theory or any other method that is likely to be effective). In this case, the key challenge is in designing the coordinator so that it can select, monitor and modify these strategies in line with the agent's overarching negotiation objectives.

References

- [Faratin, 2001] P. Faratin. *Automated Service Negotiation Between Autonomous Computational Agents*, PhD Thesis. Queen Mary College, London, England, 2001.
- [Nguyen and Jennings, 2003] T. D. Nguyen and N. R. Jennings. Concurrent bi-lateral negotiation in agent systems. *Proceedings of the 4th DEXA Workshop on e-Negotiations*, Prague, Czech Republic, 2003.

Imitation Learning of Team-play in Multiagent System based on Hidden Markov Modeling

Itsuki Noda

Cyber Assist Research Center, AIST and PRESTO, JST, Japan
i.noda@aist.go.jp

Abstract

This paper addresses agents' intentions as building blocks of imitation learning that abstract local situations of the agent, and proposes a hierarchical hidden Markov model (HMM) to represent cooperative behaviors of teamworks. The key of the proposed model is introduction of gate probabilities that restrict transition among agents' intentions according to others' intentions. Using these probabilities, the framework can control transitions flexibly among basic behaviors in a cooperative behavior.

1 Introduction

Imitation learning is considered to be a method to acquire complex human and agent behaviors and as a way to provide seeds for further learning [KI93; Sch99; MK98]. While those studies have focused on imitating behaviors of single agents, few works address imitation for teamwork among multiple agents because the complexity of the world state increases drastically in multi-agent systems. On the other hand, stochastic models like hidden Markov models (HMM) have been studied as tools to model and to represent multi-agent/human interactions [ORP00; IB99]. It is, however, hard to apply these stochastic models to imitate teamworks by observation because of the complexity of the model of multiple agents. This study focuses upon *intentions* of agents as building blocks of an abstract state of the local world for the agent in order to overcome the problem. Using *intention*, I formalize teamwork and propose a hierarchical hidden Markov model for imitation learning of teamwork.

2 Teamwork and Imitation

2.1 Intention and Play

We suppose that an *intention* is a short-term idea to achieve a certain condition from another condition. For example, in soccer, the intention 'to guide a ball in a certain direction' is an idea to move to a certain direction with the ball. We assume that an *intention* is an individual idea; therefore, an agent does not pay attention to others' efforts to achieve their intention.

A *play* is postulated as a sequence of atomic actions to achieve a single *intention*. The *play* is a basic building block

of overall behavior of agents. For example, in soccer, a 'dribble' is a *play* to achieve the *intention* 'guide a ball in a certain direction', which consists of atomic actions like 'turn', 'dash', 'kick', and so on. A play for the intention is also an individual behavior without collaboration with other agents because an intention is an individual idea. As shown below, an intention and the corresponding play are used as a main trigger to synchronize team-plays among multiple agents. This means that the intention is treated as a kind of partial condition of the world.

2.2 Team-play

We suppose that *team-play* is a collection of plays performed by multiple agents to achieve a certain purpose. As mentioned in the previous section, an intention is an individual idea. This means that multiple agents who do not change their intentions can not perform a *team-play* because they have no way to synchronize their plays. Instead, we assume that they can synchronize their plays by changing their intentions according to situations of environments and intentions of other agents. For example, in soccer, when two players (passer and receiver) guide a ball by dribble and pass, players will change their intentions as follows:

passer	receiver
dribbling	:
↓	supporting
seek-receiver	↓
↓	free-run
pass	↓
:	receive

In this example, the passer and the receiver initially have intentions 'dribbling' and 'supporting', respectively. Then, the passer changes the intention to 'seek-receiver', followed by the receiver's change to 'free-run', the passer's change to 'pass', and so on. Play synchronization is represented as conditions when agents can change the intention. In the example, the passer changes its intention from 'seek-receiver' to 'pass' when the teammate's intention is 'free-run'.

2.3 Imitation Learning of Team-play

The imitation learning of a teamplay is formalized as follows:

- (1) **Observation:** to observe behaviors of mentor agents and estimate what intention each agent has at each time step.
- (2) **Extraction:** to extract conditions prevailing when each agent

changes intentions. A condition is represented as a conjunction of others' intentions. (3) **Generation**: to generate a sequence of intentions according to changes of environment and others' intentions. In the second step of this process, the *intention* plays an important role: that is, conditions of changes of intentions. As described in Section 2.1, we consider that *intention* can represent world conditions. In addition to it, we use only *intentions* to construct rules for agents to change their *intentions*.

3 Hierarchical Hidden Markov Model for Agents

3.1 Single Behavior Model

We formalize behaviors of a basic *play* m performed by a single agent as a Moore-type HMM as follows:

$$\text{HMM}^s_m = \langle S_m, V_m, P_m, Q_m, R_m \rangle,$$

where $S_m = \{s_{mi} | i \in S_m\}$ is a set of states, $V_m = \{v_k\}$ is a set of a pair of sensor value and action commands which are used as outputs from the state, $P_m = \{p_{mij} | i, j \in S_m\}$, $Q_m = \{q_{mi}(v) | i \in S_m, v \in V_m\}$, and $R_m = \{r_{mi} | i \in S_m\}$ are probability matrixes of state transition, state-output, and initial state, respectively.

3.2 Cooperative Behavior Model

As discussed in the previous section, we consider that team-play consists of a sequence of intentions of multiple agents. This means that cooperative behavior of a single agent in a team of agents is considered as transitions among several basic plays (HMM^s). Therefore, we formalize cooperative behavior as the following modified Mealy-type HMM,

$$\text{HMM}^c = \langle M, U, E, F, G, H \rangle,$$

where $M = \{\text{HMM}^s_m\}$ is a set of basic plays and $U = \{u_k\}$ is a set of output from the model (normally, same as M); $E = \{e_m | m \in M\}$ is a set of initial play probabilities, $F = \{f_{min} | m \in M, i \in S_m, n \in M\}$ is a set of exiting probabilities from plays, and $G = \{g_{mnj} | m \in M, n \in M, j \in S_n\}$ is a set of entering probabilities to plays. Also, $H = \{h_{mn}(u) | m \in M, n \in M, u \in U\}$ is a set of gate probabilities between plays. Formally, these probabilities are defined as: $e_m = \Pr(R_m^{(0)})$, $f_{min} = \Pr(R_n^{(t)} | s_{mi}^{(t)})$, $g_{mnj} = \Pr(s_{nj}^{(t)} | R_m^{(t-1)})$, and $h_{mn}(u) = \Pr(u^{(t-1)} | R_m^{(t-1)}, R_n^{(t)})$.

Using these probabilities, an actual probability from state i in play m to state j in play n is calculated as

$$\Pr(s_{nj}^{(t)} | s_{mi}^{(t-1)}) = \begin{cases} f_{mim} p_{mij} & m = n \\ f_{min} g_{mnj} & m \neq n \end{cases}.$$

3.3 Joint-Behavior Model

Finally, we coupled multiple HMM^c s, each of which represents the behavior of an agent. Coupling is represented by gate probabilities H . For example, when agent X and agent Y are collaborating with each other, the gate probability $h_{mn}(u)$ in HMM^c for agent X indicates the probability that agent Y is performing play u at time t when agent X changes the play from m to n during time $t \rightarrow t + 1$. Using the gate probability,

the agent calculate a likelihood of the state s_{nj} at a certain time $t + 1$ according to the following equation:

$$L(s_{nj}^{(t+1)}) = \begin{cases} f_{mim} p_{mij} q_{mj}(\hat{v}^{(t+1)}) & m = n \\ f_{min} g_{mnj} h_{mn}(u^{(t)}) q_{nj}(\hat{v}^{(t+1)}) & m \neq n \end{cases} \quad (1)$$

, where $\hat{v}^{(t+1)}$ is a partially observed output value in v at time $t + 1$.

4 Experiments

I applied the framework to collaborative play of soccer. The demonstration by mentors is dribble and pass play as shown in Fig. 1: A player starts to dribble from the center of the left half field and brings the ball to the right half. At the same time, another player runs parallel along the upper (or lower) side of the field supporting the dribbling player. Then, the first player slows to look-up the second player; it then passes the ball to that player. Simultaneously, the second player starts to dash to the ball and dribbles after receiving the ball. After the pass, the first player exchanges roles with the teammate so that it becomes a supporting player for the second player.

To imitate this demonstration, I trained six HMM^s to model 'dribble', 'slow-down and look-up', 'pass', 'free-run', 'chase-ball', and 'support'. Each of HMM^s has 5 states. The output of these HMM^s s consists of local situations (the relative position and the velocity to the ball) and agent's actions ('turn', 'dash', 'small-kick', 'long-kick', 'trap', and 'look'). Note that there is no information about others' situations for output of HMM^s s. As described in Section 2.2, others' situations are taken into account during the **Extraction** phase in learning.

Two HMM^c s for agent X (the first player) and Y (the second player) are constructed after the training of the HMM^s s. Then, the learner observes behaviors of the mentor and adjusts probabilities of the HMM^c s.

Figure 2 shows result of observation and estimation. This figure shows the relative likelihood of each play state for each agent at each timestep estimated by **Observation** phase. In this figure, there are 12 rows of small squares: upper 6 rows correspond 6 plays of the first player (agent X), and the rest 6 are plays for the second player (agent Y). Each row corresponds to a play D, K, P, F, C, and S, each of which means 'dribble (D)', 'slow-down and look-up (K)', 'pass (P)', 'free-run (F)', 'chase-ball (C)', and 'support (S)', respectively. In each row, a column consists of 5 small squares each of which corresponds a state of HMM^s for one of the 6 plays at a certain timestep. The ratio of black area in the square indicates the relative likelihood with which the state of the HMM^s is active at the timestep. Columns are aligned along with time. So, a horizontal line of squares means changes of likelihood of a state of HMM^s . From this figure, we can see that the learner estimates that the agent X starts the play with the intention of 'dribble', followed by 'slow-down', 'pass' and 'support', while the player Y starts 'support' play, followed by 'chase-ball' and 'dribble' plays.

After the training by the **Observation**, the learner can generate behaviors similar to the demonstration by using the acquired probabilities of the HMM^c s as shown in Fig. 3. This

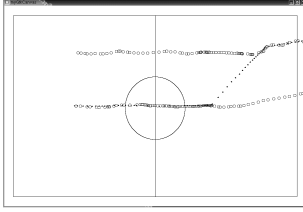


Figure 1: Exp. 3: Dribble and Pass Play by Mentor

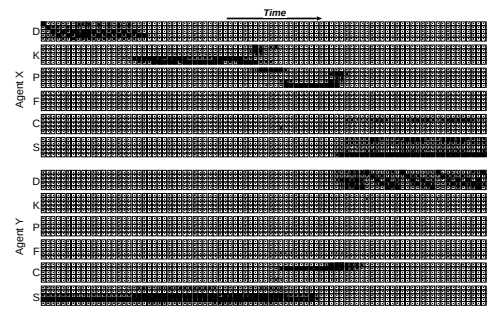


Figure 2: Exp. 3: Result of Recognition of Mentor's Behaviors

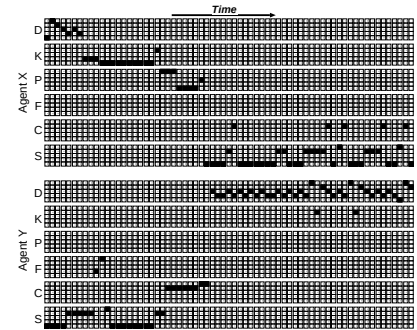


Figure 3: Exp.3: State Transitions Generated by Learned HMM

figure is constructed in the same way as Fig. 2, but only one square is filled in a timestep because the learner decides one of the possible states according to the likelihood shown in Eq. 1. In this example, although the learner sometimes generates wrong state transitions, for example a transition to states to the 'free-run' play in agent Y during agent X is doing 'slow-down', it recovers to the suitable transitions and continues to imitate the demonstrator. This shows robustness of the model against accidents. Because the model is coupled loosely with world and other's states by output probabilities of HMM, it can permit variation and misunderstanding of world and others' states.

5 Discussion

There are several works on coupling HMMs that can represent combinational probabilistic phenomena like multi-agent collaboration [JGS97; GJ97; JGJS99]. In these works, probabilistic relation among several HMMs (agents) are represented as state-transition probabilities, such that the amount of memory complexity increases exponentially. This is a serious problem for imitation learning because we assume that the number of examples for imitation is small. In our model, the relation among agents is represented by gate probabilities H , in which others' states are treated as outputs instead of as conditions of state transition. Using them, the likelihoods of state-transitions are simplified as products of several probabilities (Eq. 1). In addition, detailed states of other agents are abstracted by play (intention). As a result, the number of parameters is reduced drastically, so that learning requires very small number of examples as shown in above examples. Although such simplification may decrease flexibility of representation as a probabilistic model, experiments show that the proposed model has enough power to represent team-play among agents.

Intention in the model brings another aspect to communication among agents. We assume that there are no mutual communication in the proposed model. However, we can introduce communication as a bypass of observation and estimation of other's intention (play). The proposed model will be able to provide criteria for when an agent should inform their intention to others by comparing agents' actual intentions and estimated intention of the agent itself by simulating its own **HMM**.

One important issue is the design of the intention. In the proposed model, intentions play various important roles like chunking of the actions and conditions of world state. Therefore, we must design intentions carefully so that team-plays can be represented flexibly.

References

- [GJ97] Zoubin Ghahramani and Michael I. Jordan. Factorial hidden markov models. *Machine Learning*, 29:245–275, 1997.
- [IB99] Y. A. Ivanov and A. F. Bobick. Recognition of multi-agent interaction in video surveillance. In *International Conference on Computer Vision (Volume 1)*, pages 169–176. IEEE, Sep. 1999.
- [JGJS99] Michael I. Jordan, Zoubin Ghahramani, Tommi Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.
- [JGS97] Michael I. Jordan, Zoubin Ghahramani, and Lawrence K. Saul. Hidden markov decision trees. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 501. The MIT Press, 1997.
- [KI93] Y. Kuniyoshi and H. Inoue. Qualitative recognition of ongoing human action sequences. In *Proc. IJCAI93*, pages 1600–1609, 1993.
- [MK98] Hiroyuki Miyamoto and Mitsuo Kawato. A tennis serve and upswing learning robot based on bi-directional theory. *Neural Networks*, 11:1331–1344, 1998.
- [ORP00] Nuria M. Oliver, Barbara Rosario, and Alex Pentland. A bayesian computer vision system for modeling human interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):831–843, 2000.
- [Sch99] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 3(6):233–242, Jun. 1999.

Virtual World as Interface for Human-Robot Interaction

Eric Normand and Sheila Tejada

University of New Orleans

Lakefront Campus

Department of Computer Science

311 Mathematics Bldg

New Orleans, LA 70148

enormand@cs.uno.edu and sheila@cs.uno.edu

Abstract

This article describes preliminary work on a research environment called Virtual Synergy to represent a shared virtual map of an area for multiple autonomous robots by modifying the GameBots 3D multi-agent system. The use of GameBots will allow multiple users to interact with robots and agents at different levels of adjustable or dynamic autonomy. By interacting with the robots as another team member, the users take on different roles to suit the situation: supervisor, peer, mechanic, teleoperator, and spectator.

1 Introduction

GameBots [Adobbati *et al.*, 2001] is a modification of the commercial 3D multi-player video game *Unreal Tournament*. We are creating an adaptation of GameBots for robotic collaboration and intelligent agent research called Virtual Synergy. Our system allows the robots to share two environments at once: the physical environment and a virtual environment hosted on a GameBots server, which can be accessed remotely via the Internet; we'll deal with that aspect in section 2. The server also allows a user to be present in the virtual environment and interact with the robot through various mechanisms, which will be discussed in section 3. Section 4 discusses the use of software agents. Finally, in section 5, we will address how the current system can be expanded to include intelligent agents to facilitate human-robot interaction.

2 Inhabiting Multiple Worlds

We are developing a robot team to participate in the AAAI Urban Search and Rescue (USAR) competition. One of the tasks for the robots is to explore a simulated disaster area and map the environment, locating walls, fallen objects, areas to avoid (for structural reasons), and victims.

The Virtual Synergy system provides a means for agents in the virtual world to create a three-dimensional map by placing objects that are detected in the real world in their corresponding locations in the virtual environment. Each robot on the team will exist in both the physical world and the virtual world of Virtual Synergy. The robot will receive information from its sensors in the real environment and the Virtual Synergy server. The server will act as a virtual sensor, but providing map information instead of data. Because multiple agents will process the information gathered from multiple sensors, we must deal with the issues of sensor fusion.

A similar system called Player/Stage has been developed for simulating sensors and robots in a virtual two-dimensional world [Gerkey *et al.*, 2001]. Both Player/Stage and GameBots are designed to allow heterogeneous systems to communicate with the server over an IP network. Player/Stage sends simulated sensor readings to the client and the client sends actuator commands back to Player/Stage. The Virtual Synergy system, however, takes a higher-level approach; it provides mapping and path information; it also has the advantage of a three-dimensional view and the ability for humans to take on a persona in the environment.

3 Human Interactions

The nature of disaster areas often prohibits humans from being physically present; therefore, it is necessary to monitor and direct the robots from a distance, occluded by walls and debris. Research in teleoperation has focused on multiple operators controlling one robot. Murphy *et al.* suggest that teleoperation can lead to "cognitive fatigue" of the operators during long periods of operation and that the information received by the user often exceeds the user's capacity to process it [Murphy, 2000]. Our system attempts to resolve these issues with an intuitive interface that mimics physical interaction with the robots. Research dealing with multiple users interacting with multiple robots has shown the need for "adjustable autonomy," in which a

robot can be controlled to varying degrees by user intervention [Kortenkamp *et al.*, 2002]. Multiple levels of adjustable autonomy will be possible with the Virtual Synergy system.

A user in the environment of a Virtual Synergy server has various modes of navigation; he/she can fly unimpeded by gravity, walk through walls, or use a more realistic physics model. The interface for navigation is intuitive; it was chosen because it was designed for navigation of a three-dimensional environment by multiple human players.

We are researching the use of user roles to facilitate user interactions with robots. Current research [Scholtz, 2002] defines three levels of interaction with a robot: supervisor, peer, and mechanic interactions. The different roles define the information displayed to the user and the set of possible interactions available to the user. In addition to the three roles described in [Scholtz, 2002], we add two other roles: spectators can view the environment but not interact with the robots and teleoperators take complete control of the robot by temporarily inhabiting its representation in order to direct it through areas of difficulty.

Perzanowski *et al.* have developed a system for controlling teams of robots using voice and gesture commands [Perzanowski *et al.*, 2002]. Our robotic interactions are currently not based on speech or gestures, but on direct manipulation of the robot's world model. While navigating in the virtual environment, the user will instruct the robots through various commands such as "come here." The commands are English-like only for purposes of user interface (natural language and gesture commands are a foreseen addition). The user can also directly modify the environment by leaving a path to follow or indicating "hot spots" which suggest areas to explore. The system also provides the user the ability to manipulate the environment in order to rectify any sensor anomalies.

4 Software Agents

Autonomous software agents will also inhabit the virtual environment as teammates. Agents can facilitate communication between the users and the robots, effectively providing a more natural, more human interface to dealing with information from the robots. Since all agents (robots, software agents, and people) in the virtual environment are collaborators on the same team, they can all take on the different roles with each other; in addition to carrying out user-specified tasks, the software agents could also autonomously lead a group of robots with minimal human supervision.

5 Conclusion and Future Work

The ultimate goal of our project is to facilitate human interaction with a team of robots and agents. We are researching ways to increase the efficiency of robot collaborators; our aim is to decrease the number of humans required to monitor and collaborate with a larger team of

robots without overloading the users with information from an increasingly complex team of autonomous agents.

The two problems we are immediately facing are sensor fusion and localization. Localization is important for creating an accurate map of the environment; sensor fusion becomes a problem once two robots are involved in building the map: they could potentially be looking at the same object. We would not want the same object to appear twice.

The use of two environments, one virtual, one actual, allows for complex human-robot interaction and useful inter-robot collaboration. Using the 3D environment of Virtual Synergy will give the user an intuitive interface to interact with the robots as another member of the team. User roles will act to define the possible interactions with the robots. Intelligent agents will assist the user by responding to queries, gathering information, and automating tasks.

References

- [Adobbati *et al.*, 2001] Rogelio Adobbati, Andrew N. Marshall, Andrew Scholer, Sheila Tejada, Gal Kaminka, Steven Schaffer, and Chris Solitto. Gamebots: A 3D Virtual World Test-Bed For Multi-Agent Research, *Proceedings of the International Conference on Autonomous Agents (Agents-2001) Workshop on Infrastructure for Agents, MAS, and Scalable MAS*, Montreal, Canada, 2001.
- [Gerkey *et al.*, 2001] Brian P. Gerkey, Richard T. Vaughan, Kasper Støy, Andrew Howard, Gaurav S. Sukhatme, and Maja J. Mataric. Most Valuable Player: A Robot Device Server for Distributed Control, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001)*, pages 1226-1231, Wailea, Hawaii, October 29 - November 3, 2001.
- [Kortenkamp *et al.*, 2002] David Kortenkamp, Debra Schreckenghost, and Cheryl Martin. User Interaction with Multi-Robot Systems in (Eds.) A.C. Schultz and L. E. Parker. *Multi-Robot Systems: From Swarms to Intelligent Automata*, pages 177-184, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.
- [Murphy, 2000] Robin R. Murphy. Introduction to AI Robotics. MIT Press, Cambridge, Massachusetts, 2000.
- [Perzanowski *et al.*, 2002] D. Perzanowski, A. C. Shultz, W. Adams, M. Bugajska, E. Marsh, J. G. Trafton, D. Brock, M. Skubic, M. Abramson. Communicating with Teams of Cooperative Robots in (Eds.) A.C. Schultz and L. E. Parker. *Multi-Robot Systems: From Swarms to Intelligent Automata*, pages 177-184, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.
- [Scholtz, 2002] Jean C. Scholtz. Human-Robot Interactions: Creating Synergistic Cyber Forces in (Eds.) A.C. Schultz and L. E. Parker. *Multi-Robot Systems: From Swarms to Intelligent Automata*, pages 177-184, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.

Learning Algorithms for Software Agents in Uncertain and Untrusted Market Environments

Thomas Tran and Robin Cohen

School of Computer Science, University of Waterloo

Waterloo, ON, Canada N2L 3G1

{tt5tran, rcohen}@math.uwaterloo.ca

The problem of how to develop algorithms that guide the behaviour of personal, intelligent software agents participating in electronic marketplaces is a subject of increasing interest from both the academic and industrial research communities. Since a multi-agent electronic market environment is, by its very nature, open, dynamic, uncertain, and untrusted, it is very important that participant agents are equipped with effective and feasible learning algorithms in order to achieve their goals. In this paper, we propose algorithms for buying and selling agents in electronic marketplaces, based on reputation modelling and reinforcement learning (RL).

We model the agent environment as an open marketplace which is populated with economic agents (*buyers and sellers*), freely entering or leaving the market. The process of buying and selling goods is realized via a 3-phase mechanism: (i) A buyer announces its request for a good. (ii) Sellers submit bids for delivering such goods. (iii) The buyer evaluates the submitted bids and selects a suitable seller. Thus, the buying and selling process can be viewed as an *auction* where a seller is said to be *winning the auction* if it is able to sell its good to the buyer. We assume that the quality of a good offered by different sellers may not be the same, a seller may alter the quality of its goods, and there may be dishonest sellers in the market. We also assume that a buyer can examine the quality of the good it purchases only after it receives that good from the selected seller.

In our approach, buying agents learn to avoid the risk of purchasing low quality goods and to maximize their expected value of goods by dynamically maintaining sets of reputable and disreputable sellers. Selling agents learn to maximize their expected profits by adjusting product prices and optionally altering the quality of their goods.

In our buying algorithm, a buyer b uses an expected value function f^b , where $f^b(g, p, s)$ represents buyer b 's expected value of buying good g at price p from seller s . Buyer b maintains reputation ratings for all sellers, and chooses among its set of reputable sellers S_r^b a seller $\hat{s}$ that offers good g at price p with maximum expected value. In addition, with a small probability ρ , buyer b chooses to explore (rather than exploit) the marketplace to discover new reputable sellers by randomly selecting a seller $\hat{s}$, provided that $\hat{s}$ is not from the set of disreputable sellers S_{dr}^b . After paying seller $\hat{s}$ and receiving good g , buyer b can examine the quality q of g and calculate the true value $v^b(g, p, q)$ of g . The expected value

function f^b is incrementally learned in an RL framework:

$$f^b(g, p, \hat{s}) \leftarrow f^b(g, p, \hat{s}) + \alpha(v^b(g, p, q) - f^b(g, p, \hat{s}))$$

where α is called the *learning rate* ($0 \leq \alpha \leq 1$). The reputation rating of $\hat{s}$ is then updated based on whether or not the true value of good g is greater than or equal to the value demanded by b . Our reputation updating scheme implements the traditional ideas that reputation should be difficult to build up but easy to tear down, and that a transaction with higher value should be more appreciated than a lower one. The set of reputable and disreputable sellers (S_r^b and S_{dr}^b) are accordingly re-calculated based on the updated reputation rating of $\hat{s}$.

In our selling algorithm, a seller s makes use of an expected profit function h^s , where $h^s(g, p, b)$ represents seller s expected profit if it sells good g at price p to buyer b . Seller s chooses a price $\hat{p}$ to sell good g to buyer b such that its expected profit is maximized. After the transaction, the expected profit function h^s is learned incrementally using RL:

$$h^s(g, \hat{p}, b) \leftarrow h^s(g, \hat{p}, b) + \alpha(\phi^s(g, \hat{p}, b) - h^s(g, \hat{p}, b))$$

where $\phi^s(g, \hat{p}, b)$ is the actual profit of seller s when it sells good g at price $\hat{p}$ to buyer b , and is defined as follows:

$$\phi^s(g, \hat{p}, b) = \begin{cases} \hat{p} - c^s(g, b) & \text{if } s \text{ wins the auction} \\ 0 & \text{otherwise.} \end{cases}$$

where $c^s(g, b)$ is the cost of seller s to produce good g for buyer b . Our selling algorithm also allows sellers to alter the quality of their goods in order to meet the buyers' needs and to further increase their future profit, depending on the success of their previous sales with the buyers.

We believe that our approach should lead to improved satisfaction for buyers and sellers, since buyers should be less at risk of purchasing low quality goods when maintaining sets of reputable and disreputable sellers, and since sellers are allowed to adjust both price and quality to meet the buyers' demands.

We have performed experimentation to measure the value of our model on both microscopic and macroscopic levels. On the micro level, we were interested in examining the individual benefit of agents, particularly their level of satisfaction. Our experimental results show that in both modest and large sized marketplaces, buyers following the proposed buying algorithm (i.e., using the combination of RL and reputation modelling) will achieve better satisfaction than buyers using

RL alone, and that sellers following the proposed selling algorithm (i.e., using RL and adjusting product quality) will make better profit than sellers using only RL. On the macro level, we studied how a market populated with our buyers and sellers would behave as a whole. Our results demonstrate that such a market can reach an equilibrium state where the agent population remains stable (as some sellers who repeatedly fail to sell their goods will decide to leave the market), and this equilibrium is optimal for the participant agents.

We report some experimental results confirming the satisfaction of buyers following the proposed algorithm in a large sized marketplace. The simulated market is populated with 160 sellers and 120 buyers where each buyer participates in 5000 auctions. The seller population is equally divided into four groups: Group A offers goods with quality chosen randomly from the quality interval [32, 42]. Group B consists of dishonest sellers, who try to attract buyers with high quality goods ($q = 45$) and then cheat them with really low quality ones ($q = 1$). Sellers in group C use RL but do not consider adjusting product quality; they offer goods with fixed quality 39. Sellers in group D follow the proposed selling algorithm and offer goods with initial quality 39. The buyer population is divided equally into two groups: Group I uses RL alone and group II follows the proposed buying algorithm. We set quality equal to cost to support the common idea that it costs more to produce high quality goods. Also, we set the true product value $v^b = 3.5q - p$, the demanded product value $\vartheta^b = 100$, the learning rate $\alpha = 1$ and exploration rate $\rho = 1$ initially with both decreased over time by factor 0.9997 down to $\alpha_{min} = \rho_{min} = 0.1$. The results reported here are based on the average taken over the respective populations of the two groups of buyers.

Figure 1(a) and (b) present the histograms of true product values obtained by a buyer using RL alone, and by a buyer following the proposed buying algorithm, respectively. Clearly, the buyer following the proposed algorithm receives more goods with higher value ($v^b = 110$) and fewer goods with lower values ($v^b = 65...105$), and is therefore better satisfied. In particular, the buyer following the proposed algorithm makes about 2350 more purchases with high product value of 110 (or 15.67 times greater) than those purchases made by the buyer using RL alone.

We are also interested in seeing how much better the buyer following the proposed algorithm is able to avoid interaction with the dishonest sellers. Figure 2(a) and (b) show the profits made by the dishonest sellers from the buyer using RL alone, and from the buyer following the proposed algorithm, respectively. We notice that graph (a) is higher than graph (b), indicating that the dishonest sellers are able to make more profit from those buyers that only use RL but do not model sellers' reputation. Moreover, the profit in graph (b) is reduced to zero after about 2700 auctions, implying that in the long run the dishonest sellers are not able to make any profit from the buyer following the proposed algorithm, because they are rated as disreputable sellers and therefore no longer chosen by the buyer.

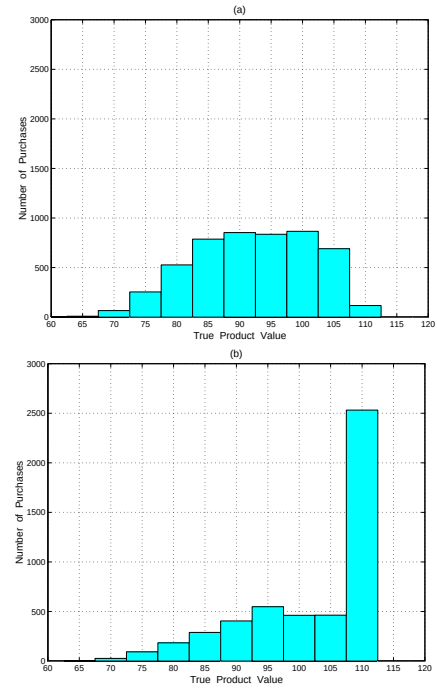


Figure 1: Histograms of true product values obtained by a buyer using RL alone (a), and a buyer using the proposed algorithm (b).

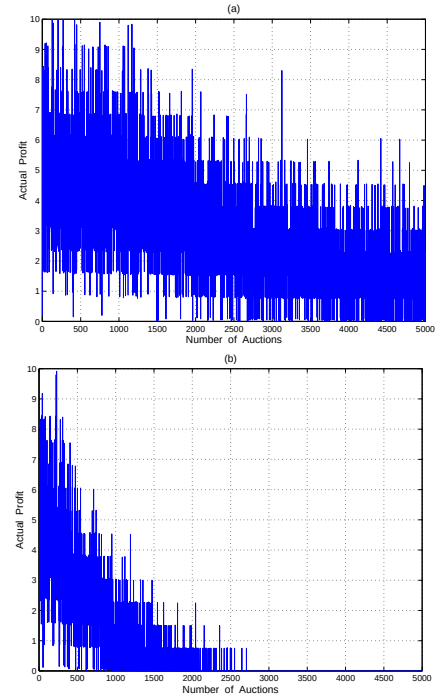


Figure 2: Profits made by the dishonest sellers from the buyer using RL alone (a), and the buyer using the proposed algorithm (b).

In general, our work demonstrates that reputation modelling can be used in combination with reinforcement learning to design intelligent learning agents that participate in electronic marketplaces. This research also aims to provide a principled framework for building effective economic agents and desirable electronic market environments.

A Multi-Agent Computational Linguistic Approach to Speech Recognition

Michael Walsh, Robert Kelly, Gregory M.P. O'Hare,
Julie Carson-Berndsen, Tarek Abu-Amer

University College Dublin, Ireland

{michael.j.walsh, robert.kelly, gregory.ohare, julie.berndsen, tarek.abuamer}@ucd.ie

Abstract

This paper illustrates how a multi-agent system implements and governs a computational linguistic model of phonology for syllable recognition. We describe how the Time Map model can be recast as a multi-agent architecture and discuss how constraint relaxation, output extrapolation, parse-tree pruning, clever task allocation, and distributed processing are all achieved in this new architecture.

1 Introduction and Motivation

This paper investigates the deployment of multi-agent design techniques in a computational linguistic model of speech recognition, the Time Map model [Carson-Berndsen, 1998]. The architecture of the model is illustrated in Figure 1. In brief, phonological features are extracted from the speech signal using Hidden Markov Models resulting in a multi-tiered representation of the utterance. A phonotactic automaton (network representation of the permissible combinations of sounds in a language) and axioms of event logic are used to interpret this multilinear representation, outputting syllable hypotheses [Carson-Berndsen and Walsh, 2000]. In what follows, we firstly introduce the multi-agent paradigm and discuss the use of agents equipped with mental models. Secondly we illustrate how the model can be recast within a multi-agent architecture. The paper concludes by highlighting the benefits of such an approach.

2 The Multi-Agent paradigm

The Multi-Agent paradigm is one which promotes the interaction and cooperation of intelligent autonomous agents in order to deal with complex tasks (see [Ferber, 1999] for an overview of the area). The agents used to recast the Time Map model are deliberative reasoning agents which use mental models. The particular architecture chosen to recast the Time Map model is known as the Belief-Desire-Intention (BDI) architecture [Rao and Georgeff, 1991]. According to the BDI paradigm an agent is equipped with a set of beliefs about its environment and itself, a set of computational states which it seeks

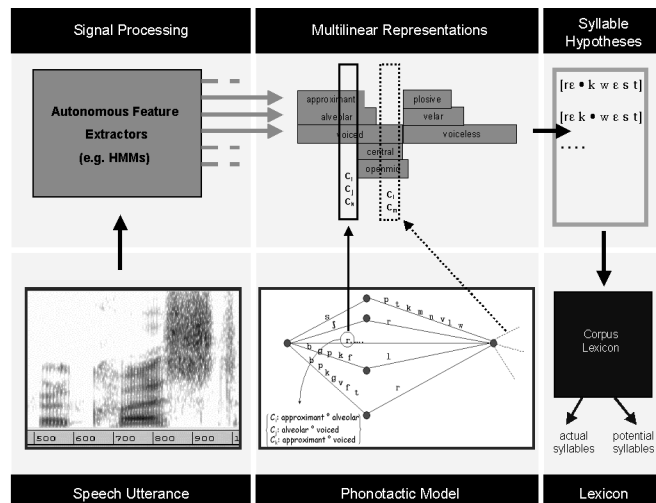


Figure 1: The Time Map model architecture

to maintain, essentially desires and a set of computational states which the agent seeks to achieve, that is intentions.

The agents are created through Agent Factory [O'Hare *et al.*, 1999], a rapid prototyping environment for the construction of agent based systems. Agents created using Agent Factory are equipped with a Mental State model and a set of methods (the actuators). Agents used in speech technology in the past [Erman *et al.*, 1996] exhibited only weak agenthood. However, the agents delivered through Agent Factory are intentional agents with rich mental states governing their deductive behaviour; they exhibit strong notions of agenthood. The model presented below is both pioneering and in stark contrast to prior research in that it represents the first attempt to explicitly commission a multi-agent approach in sub-word processing. The benefits of this model are discussed in section 4.

3 The Time Map Model as a Multi-Agent System

A number of agents are required in order to recast the Time Map model accurately, namely the Feature Extrac-

tion Agents, the Windowing Agent, the Segment Agents, and the Chart Agents. These agents are detailed below. The architecture is illustrated in Figure 2.

The Feature Extraction Agents

Numerous Feature Extraction Agents, operating in parallel, output autonomous temporally annotated features (i.e. events). These events are extracted from the utterance using Hidden Markov Model techniques, for more details see [Abu-Amer and Carson-Berndsen, 2003]. The feature output is delivered to the Windowing Agent.

The Chart Agent

Chart Agents have a number of roles to play in the syllable recognition process. One role is to inform the Windowing Agent of all phonotactically anticipated phoneme segments for the current window. This information is extracted from all transitions traversable from the current position in the phonotactic automaton. For example in Figure 2 an [s] segment has already been recognised at the onset of a syllable. Based on this the Chart Agent can predict a number of subsequent segments, including a [p] as illustrated. Thus, it communicates its segment predictions, along with their constraint, rank, threshold and corpus-based distributional information, to the Windowing Agent.

Chart Agents also monitor progress through the phonotactic automaton by maintaining records of the contiguous transitions that have been traversed as a result of recognising phonemic segments in the input. The Chart Agent is informed of which segments have been recognised by Segment Agents. If the segment is one which was anticipated (i.e. a phonotactically legal segment) by the Chart Agent then the associated transition is traversed. Chart Agents begin tracking the recognition process from the initial state of the phonotactic automaton. If a Chart Agent reaches a final state in the phonotactic automaton then a well-formed syllable is logged. It is also possible that the segment recognised is not one anticipated by the Chart Agent. In this case an ill-formed structure is logged and the Chart Agent returns to the initial state of the automaton in anticipation of a new syllable. In certain cases the Chart Agent receives recognition results from Segment Agents based on underspecified input. In these cases the Chart Agent can augment the results by adding anticipated feature information provided that there is no conflicting feature information in the input. This is known as output extrapolation.

The Windowing Agent

As previously mentioned the Windowing Agent receives segment predictions from Chart Agents for the current window. The Windowing Agent also takes the current output produced by the Feature Extraction Agents, constructs a multilinear representation of it, and proceeds to window through this representation. For each window examined it identifies potential segments that may be present based on partial examination of the feature

content of the window. Potential segments can be identified by using a resource which maps phonemes to their respective features. Priority is placed on attempting to recognise predicted segments first. Therefore, the Windowing Agent spawns a Segment Agent for each potential segment that is also predicted by a Chart Agent, before spawning a Segment Agent for potential segments not predicted by a Chart Agent. The incremental spawning of Segment Agents for potential segments is dependent on the progress made by Segment Agents already activated. In Figure 2 the Windowing Agent has received predictions from the Chart Agent. Having placed an initial window over the feature-extracted multilinear representation of the utterance, the Windowing Agent identifies a number of potential segments in the window. Segment Agents are spawned, two of which are illustrated, one for the predicted potential segment [p], and one for the unpredicted potential segment [b]. The Segment Agent is discussed below.

The Segment Agent

Each Segment Agent spawned by the Windowing Agent has a specific phonemic segment which it seeks to recognise. Segment Agents for segments which were not predicted have default information from the resource which maps phonemes to their respective features. For example, according to Figure 2 a Segment Agent attempting to recognise a [b] requires that voiced (voi+), stop and labial features all overlap in time. However, Segment Agents for segments which were predicted may have altered rankings, provided by the Chart Agent and founded on corpus-based distributional information or cognitive factors. A Segment Agent attempts to satisfy each of its overlap relation constraints by examining the current window. Each time a constraint is satisfied its rank value is added to a running total known as the Segment Agent's degree of presence. If the degree of presence reaches the threshold then the Segment Agent is satisfied that its segment has been successfully recognised. For predicted segments certain constraints may be relaxed, i.e. not all constraints need to be satisfied in order to reach the threshold. Rather than all Segment Agents reporting back to the Windowing Agent, each Segment Agent communicates its degree of presence to the other Segment Agents in the environment. Only if a Segment Agent identifies that it has the greatest degree of presence and has reached its threshold will it ask the Windowing Agent to proceed to the next window. It will also inform Chart Agents that its segment has been successfully recognised. When a Segment Agent finds that one of its constraints is satisfied it can share this result with other Segment Agents. Thus a Segment Agent attempting to satisfy the same constraint may not have to do so.

4 Benefits of the Multi-Agent Approach

The recasting of the Time Map model results in an innovative multi-agent system for speech recognition which is significantly different from more traditional approaches.

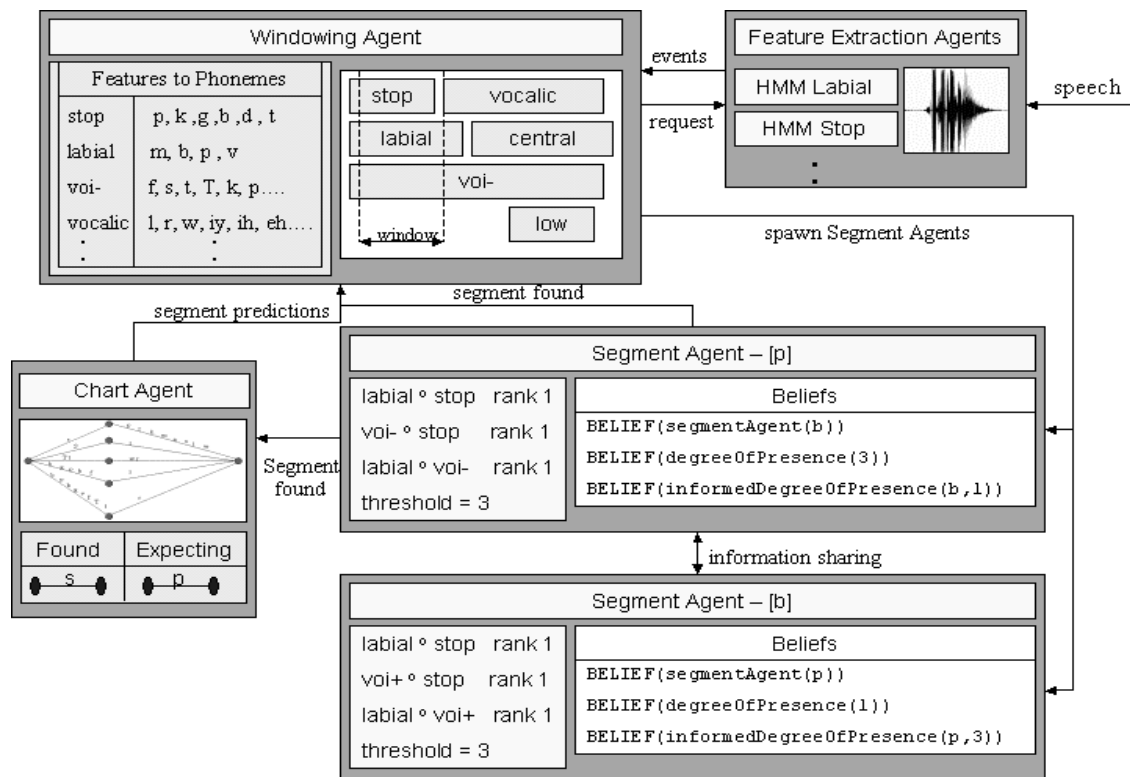


Figure 2: Multi-Agent Time Map Architecture

This new architecture provides a principled means of distributing a computationally heavy work load among several task-specific agents, operating in parallel, and collaboratively, thus alleviating the computational strain. The use of agents facilitates information sharing, search space pruning, constraint relaxation and output extrapolation. From a computational linguistic viewpoint this architecture allows an explicit separation of the declarative and procedural aspects of the model. In this way the knowledge sources can be maintained independently of the application which is particularly important in speech recognition applications to ensure scalability to new task domains and migration to other languages.

Acknowledgments

This research is part-funded by Enterprise Ireland under Grant No. IF/2001/021 and part-funded by the Science Foundation Ireland under Grant No. 02/IN1/I10. The opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of either granting body.

References

- [Abu-Amer and Carson-Berndsen, 2003] T. Abu-Amer and J. Carson-Berndsen. Multi-linear HMM Based System for Articulatory Feature Extraction. In *Proceedings of ICASSP 2003*, Hong Kong, April 2003.
- [Carson-Berndsen, 1998] J. Carson-Berndsen. Time Map Phonology: Finite State Models and Event Logics in Speech Recognition. Kluwer Academic Publishers, Dordrecht, 1998.
- [Carson-Berndsen and Walsh, 2000] J. Carson-Berndsen and M. Walsh. Interpreting Multilinear Representations in Speech. In *Proceedings of the Eight International Conference on Speech Science and Technology*, Canberra, December 2000.
- [Erman *et al.*, 1996] L.D. Erman, F. Hayer-Roth, V.R. Lesser, and D.R. Reddy. The Hearsay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty. *Comp. Surveys* Vol. 12, pp 213-253, June, 1980.
- [Ferber, 1999] J. Ferber. Multi-Agent Systems - An Introduction to Distributed Artificial Intelligence. Addison-Wesley, 1999.
- [O'Hare *et al.*, 1999] G.M.P O'Hare, B.R. Duffy, R.W. Collier, C.F.B Rooney, and R.P.S. O'Donoghue. Agent Factory: Towards Social Robots. First International Workshop of Central and Eastern Europe on Multi - agent Systems (CEEMAS'99), St.Petersburg, Russia, 1999.
- [Rao and Georgeff, 1991] A.S. Rao and M.P. Georgeff. Modelling Rational Agents within a BDI Architecture, *Prin. of Knowl. Rep. & Reas.*. San Mateo, CA., 1991.

POSTER PAPERS

NATURAL LANGUAGE

The Knowledge Required to Interpret Noun Compounds

James Fan, Ken Barker and Bruce Porter

Department of Computer Sciences

University of Texas at Austin

Austin, TX 78712 USA

{jfan, kbarker, porter}@cs.utexas.edu

Abstract

Noun compound interpretation is the task of determining the semantic relations among the constituents of a noun compound. For example, “concrete floor” means a floor made of concrete, while “gymnasium floor” is the floor region of a gymnasium. We would like to enable knowledge acquisition systems to interpret noun compounds, as part of their overall task of translating imprecise and incomplete information into formal representations that support automated reasoning. However, if interpreting noun compounds requires detailed knowledge of the constituent nouns, then it may not be worth doing: the cost of acquiring this knowledge may outweigh the potential benefit.

This paper describes an empirical investigation of the knowledge required to interpret noun compounds. It concludes that the axioms and ontological distinctions important for this task are derived from the top levels of a hierarchical knowledge base (KB); detailed knowledge of specific nouns is less important. This is good news, not only for our work on knowledge acquisition systems, but also for research on text understanding, where noun compound interpretation has a long history. A more detailed version of this paper can be found in [Fan *et al.*, 2003].

1 Introduction

Knowledge acquisition involves building knowledge bases (KBs) from the information provided by standard sources of expertise, such as people and texts. In addition to extracting relevant information, knowledge acquisition involves re-expressing the information in a formal language suitable for machines to reason with. In general, this is difficult because the information is initially expressed in natural languages, and these expressions are notoriously imprecise and incomplete. However, the goal of our project is to improve knowledge acquisition methods by automating the translation of some kinds of expressions from natural languages to formal ones. We call this task *loose-speak interpretation*.

Several kinds of expressions are good candidates for loose-speak interpretation by knowledge acquisition systems. For

example, noun compounds omit information that can often be inferred, *e.g.* *concrete floor* is “a floor *made of* concrete”, while *gymnasium floor* is “the floor *region of* a gymnasium”. Another candidate is metonymic expressions. These expressions contain incompatible terms and must be expanded to make meaningful phrases. For example, the statement *Joe read Shakespeare*, means “Joe read *text written by* Shakespeare”.

This paper focuses on the first kind of loose-speak: interpreting noun compounds in the context of knowledge acquisition. A noun compound is a sequence of nouns composed of a head noun and one (or more) modifiers. The head noun determines the type of the whole compound (with few exceptions), and the modifiers specialize the type from the head noun. Although we limit our study to only pairs of nouns, our results can be applied to longer noun compounds by bracketing them into pairs of nouns (with few exceptions), and then interpreting each pair [Lieberman and Sproat, 1992; Pustejovsky and Bergler, 1993; Barker, 1998].

The computational linguistics community has studied noun compound interpretation extensively [Leonard, 1984; Downing, 1977; Levi, 1979; Finin, 1986; Fabre, 1996; Lauer and Dras, 1994; Barker, 1998; Vanderwende, 1994]. In these studies, the task is to select a single semantic category for each pair of nouns. The selection is usually made from a small list of semantic categories, for example category *part-of* is selected for noun compounds like *human lung*, category *material* for *marble statue*, and category *object-of* for *troop movement*.

Our task is more general. Rather than selecting a single semantic category, our task is to find a sequence of semantic relations that links two nouns in a compound. Semantic relations are a list of about 50 thematic roles such as *agent*, *object*, *has-part*, *location*, For example, given *animal virus*, a traditional interpretation may classify this as a location category (*animal virus* is a virus in an animal). A loose-speak interpretation may be composed of a combination of semantic relations, such as: “an *animal virus* is a virus that is the agent of an *invade*, such that the object of the *invade* is the cell part of an animal”.

Furthermore, computational linguists approach the noun compound interpretation task armed with lots of examples, but little or no knowledge about the constituent nouns. Typical solutions are based on statistical patterns discovered in

the corpus of examples. In contrast, we approach the task in the context of the constituent nouns knowledge – their taxonomic classification, at least – but few examples of noun compounds, let alone a corpus.

2 Interpreting noun compounds during knowledge acquisition

During knowledge acquisition, the domain expert (or, more generally, the knowledge source) may provide a noun compound in any dialogue that expects a noun. Our knowledge acquisition system successfully interprets the noun compound if it finds a correct sequence of semantic relations between the head noun and its modifier and builds a correct formal representation of the noun compound.

If noun compound interpretation requires *a priori*, detailed knowledge of the head noun and its modifier, then the cost of acquiring this knowledge may overshadow the benefit of interpreting the compound. If, on the other hand, noun compounds can be successfully interpreted without much knowledge about the specific constituent nouns, then the problem is avoided, and a knowledge acquisition system might interpret one concept (the noun compound) while related concepts (the constituent nouns) are only skeletal. Knowledge bases tend to grow in this uneven way – following the lead of the knowledge sources providing expertise – and a knowledge acquisition system should support it.

The purpose of this study is to determine what sort of knowledge is required to interpret noun compounds, and how this knowledge might be obtained. Before delving into the details of the study, it's important to understand what we are *not* attempting to do.

We are not presenting a novel algorithm for noun compound interpretation. Our algorithm is quite simple and is derived from previous research. Also, we are not introducing a new type of knowledge representation or a novel technique of automated reasoning. Finally, we are not using a new, comprehensive knowledge base. We built a couple of them rather quickly and we're using another – not built for this task – “off the shelf.”

In summary, what we *are* doing is evaluating the knowledge requirements of a standard search algorithm applied to a variety of typical knowledge bases through a series of ablation studies.

3 Experiments

The challenge in measuring an algorithm's sensitivity to knowledge base content is that the results may vary across domains and across knowledge bases. We attempt to neutralize these factors by replicating our study in three domains with quite different knowledge bases.

The noun compound interpretation task can be viewed as follows: given a knowledge base encoded as a conceptual graph, and a pair of nouns corresponding to two nodes in the graph, find a path of semantic relations between them. The algorithm we used is a breadth-first search algorithm on a knowledge base. The algorithm is given a noun compound of the form $\langle C_1, C_2 \rangle$ where C_1 and C_2 are the KB concepts that are mapped from the constituents of the given noun

compound. Each of the first two steps conducts a breadth-first search of the knowledge base along all semantic relation arcs. The first search starts from C_1 and looks for C_2 or any superclass or subclass of it. The second search starts from C_2 and looks for C_1 or any superclass or subclass of it. Step 3 combines the results, sorted by path length.

To avoid getting results that are skewed to a particular domain or knowledge representation, we used a variety of quite different data sets. The first consists of 224 noun compounds from a college-level cell biology text [Alberts *et al.*, 1998]. The second consists of 294 noun compounds from a small engine repair manual. The third data set consists of 224 compounds from a Sun Sparcstation manual. The nouns used in these data sets are mapped to the corresponding concepts in knowledge bases on these topics.

Despite these commonalities, the KBs differ significantly. First they differ in terms of how they were built. The knowledge base for the biology text was built using the generic Component Library [Barker *et al.*, 2001] to answer end-of-the-chapter style questions, as one of the challenge problems for DARPA's Rapid Knowledge Formation project [Clark *et al.*, 2001]. The knowledge bases for the other two data sets (the small engine repair manual and the Sparcstation manual) were built “on top of” the knowledge in WordNet [Fellbaum, 1998]. We augmented WordNet with the upper ontology of the generic Component Library plus about ten concepts that are important to each of the two domains whose paronyms are not complete in WordNet. Through this process, we encoded 416 concepts in about 50 man-hours. The advantage of using WordNet as the foundation for these knowledge bases is two-fold: it includes most of the terms used in the data sets, linked with both taxonomic and paronymic relations, and it is widely available and well used. The KBs also differ in content. Other than the shared upper ontology of the generic component library, they have few concepts in common.

The importance of each level of the ontology is measured through a series of ablations. When a level is ablated, the concepts on that level and all their axioms are deleted from the knowledge base. The superclasses of the subclasses of these concepts are changed to the superclasses of the concepts being deleted. As a special case, when the 0th level (the root level concept) is deleted, it is replaced by a generic concept of “Thing”. Because the root level concept is vacuous, deleting it has no affect.

4 Results

Ablating levels of the ontology shows that they differ in their importance for the noun compound interpretation task. Without any ablation, both precision and recall of the noun compound interpretation are around 80% across all three knowledge bases. Ablating level one causes a big drop in both precision and recall. Ablating level two introduces a big gap between precision and recall because the algorithm does not find interpretations for many noun compounds. As lower levels in the ontology are ablated one at a time, the impact diminishes and performance improves to the level of a knowledge base with no ablations.

The contribution of the first two levels of the ontology is

observed across all three data sets and knowledge bases. This pattern strongly suggests that top levels of the ontology are most important for the noun compound interpretation task, which is likely due to some combination of two factors:

1. Top levels of the ontology include concepts that make important ontological distinctions. For example, ablating the level that introduces *Entity* and *Event* blurs the distinction between obviously different concept, which causes the search to stop with erroneous results. Consequently, many more interpretations are returned, and because we only use the first interpretation, the possibility of it being correct is reduced.
2. Although they contain relatively few axioms, these axioms are important for the task. The top-level ontology contains the most frequently used axioms, such as that “every Action involves an object that is acted upon”. These axioms are used in the search as a step along the way. Deleting these axioms makes it difficult to find an interpretation for many of the noun compounds, thereby causing recall to lag behind precision.

5 Conclusion

This paper reports an encouraging result: interpreting noun compounds does not require detailed knowledge of the constituent nouns. Rather, it requires only that the nouns be correctly placed in a taxonomy, and that the taxonomy include the ontological distinctions and axioms commonly found in domain independent upper levels. These requirements are easily met in the context of knowledge acquisition, which is our focus.

We reached this conclusion using a novel experimental method. We measured the contribution of each level of the ontology to the task of interpreting noun compounds. We ablated successive levels of the ontology one at a time, thereby conflating ontological distinctions and removing the axioms associated with concepts at each level. We found that the upper levels of the ontology for the KBs we used are the most important for noun compound interpretation. As lower levels in the ontology are ablated one at a time, the impact diminishes to nil. A more detailed version of this paper can be found in [Fan *et al.*, 2003].

Acknowledgments

Support for this research is provided by a contract from Stanford Research Institute as part of DARPA’s Rapid Knowledge Formation project.

References

- [Alberts *et al.*, 1998] Bruce Alberts, Dennis Bray, Alexander Johnson, Julian Lewis, Martin Raff, Keith Robert, Peter Walter, and Keith Roberts. *Essential Cell Biology: An Introduction to the Molecular Biology of the Cell*. Garland Publisher, 1998.
- [Barker *et al.*, 2001] Ken Barker, B. Porter, and P. Clark. A library of generic concepts for composing knowledge bases. In *Proceedings of First International Conference on Knowledge Capture*, 2001.
- [Barker, 1998] Ken Barker. *Semi-automatic Recognition of Semantic Relationships in English Technical Texts*. PhD thesis, University of Ottawa, Ottawa, Ontario, 1998.
- [Clark *et al.*, 2001] Peter Clark, J. Thompson, K. Barker, B. Porter, V. Chaudhri, A. Rodriguez, J. Thomere, S. Mishra, Y. Gil, P. Hayes, and T. Reichherzer. Knowledge entry as the graphical assembly of components. In *Proceedings of First International Conference on Knowledge Capture*, 2001.
- [Downing, 1977] Pamela A. Downing. On the creation and use of English compounds. *Language*, 53:810 – 842, 1977.
- [Fabre, 1996] Cecile Fabre. Interpretation of nominal compounds: Combining domain independent and domain-specific information. In *Proceedings of Sixteenth International Conference on Computational Linguistics*, pages 364 – 369, 1996.
- [Fan *et al.*, 2003] James Fan, Ken Barker, and Bruce Porter. The knowledge required to interpret noun compounds. Technical Report UT-AI-TR-03-301, University of Texas at Austin, 2003.
- [Fellbaum, 1998] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. The MIT Press, Boston, 1998.
- [Finin, 1986] Timothy W. Finin. *Constraining the Interpretation of Nominal Compounds in a Limited Context*. Lawrence Erlbaum Assoc, New Jersey, 1986.
- [Lauer and Dras, 1994] Mark Lauer and Mark Dras. A probabilistic model of compound nouns. In *Proceedings of the 7th Australian Joint Conference on Artificial Intelligence*. World Scientific Press, 1994.
- [Leonard, 1984] Rosemary Leonard. *The Interpretation of English Noun Sequences on the Computer*. Elsevier Science, Amsterdam, 1984.
- [Levi, 1979] Judith Levi. *The Syntax and Semantics of Complex Nominals*. Academic Press, New York, 1979.
- [Lieberman and Sproat, 1992] Mark Lieberman and Richard Sproat. *Stress and Structure of Modified Noun Phrases*. CSLI Publications, 1992.
- [Pustejovsky and Bergler, 1993] James S. Pustejovsky and Anick P. Bergler. Lexical semantic techniques for corpus analysis. *Computational Linguistics*, 19(2):331 – 358, 1993.
- [Vanderwende, 1994] Lucy Vanderwende. Algorithm for automatic interpretation of noun sequences. In *Proceedings of Fifteenth International Conference on Computational Linguistics*, pages 782 – 788, 1994.

Improving Word Sense Disambiguation in Lexical Chaining

Michel Galley and Kathleen McKeown

Columbia University

Department of Computer Science

New York, NY 10027, USA

{galley, kathy}@cs.columbia.edu

Abstract

Previous algorithms to compute lexical chains suffer either from a lack of accuracy in word sense disambiguation (WSD) or from computational inefficiency. In this paper, we present a new linear-time algorithm for lexical chaining that adopts the assumption of one sense per discourse. Our results show an improvement over previous algorithms when evaluated on a WSD task.

1 Introduction

Passages from spoken or written text have a quality of unity that arises in part from the surface properties of the text; syntactic and lexical devices can be used to create a sense of connectedness between sentences, a phenomenon known as *textual cohesion* [Halliday and Hasan, 1976]. Of all cohesion devices, lexical cohesion is probably the most amenable to automatic identification [Hoey, 1991]. Lexical cohesion arises when words are related semantically, for example in reiteration relations between a term and a synonym or superordinate.

Lexical chaining is the process of connecting semantically related words, creating a set of chains that represent different threads of cohesion through the text. This intermediate representation of text has been used in many natural language processing applications, including automatic summarization [Barzilay and Elhadad, 1997; Silber and McCoy, 2003], information retrieval [Al-Halimi and Kazman, 1998], intelligent spell checking [Hirst and St-Onge, 1998], topic segmentation [Kan *et al.*, 1998], and hypertext construction [Green, 1998].

A first computational model of lexical chains was introduced by Hirst and St-Onge [1998]. This linear-time algorithm, however, suffers from inaccurate WSD, since their greedy strategy immediately disambiguates a word as it is first encountered. Later research [Barzilay and Elhadad, 1997] significantly alleviated this problem at the cost of a worse running time (quadratic); computational inefficiency is due to their processing of many possible combinations of word senses in the text in order to decide which assignment is the most likely. More recently, Silber and McCoy [2003] presented an efficient linear-time algorithm to compute lexical chains, which models Barzilay's approach, but nonetheless has inaccuracies in WSD.

In this paper, we further investigate the automatic identification of lexical chains for subsequent use as an intermediate representation of text. In the next section, we propose a new algorithm that runs in linear time and adopts the assumption of one sense per discourse [Gale *et al.*, 1992]. We suggest that separating WSD from the actual chaining of words can increase the quality of chains. In the last section, we present an evaluation of the lexical chaining algorithm proposed in this paper, and compare it against [Barzilay and Elhadad, 1997; Silber and McCoy, 2003] for the task of WSD. This evaluation shows that our algorithm performs significantly better than the other two.

2 Lexical Chaining with a Word Sense Disambiguation Methodology

Our algorithm uses WordNet [Miller, 1990] as a knowledge source to build chains of candidate words (nouns) that are related semantically. We assign a weight to each semantic relation; in our work semantic relations are restricted to hypernyms/hyponyms (e.g. between *cat* and *feline*) and siblings (hyponyms of hypernyms, e.g. *dog* and *wolf*). Distance factors for each type of semantic relation prevent the linkage of words that are too far apart in the text; these factors are summarized in Table 1.

The algorithm can be decomposed into three steps: building a representation of all possible interpretations of the text, disambiguating all words, and finally building the lexical chains. First, similarly to [Silber and McCoy, 2003], we process the whole text and collect all semantic relations between candidate words under *any* of their respective senses. No disambiguation is done at this point; the only purpose is to build a representation used in the next stages of the algorithm. Note that this is the only stage where the text is read; all later stages work on this implicit representation of possible interpretations, called a *disambiguation graph* (Figure 1). In this kind of graph, nodes represent word instances and weighted edges represent semantic relations. Since WordNet doesn't relate words but senses, each node is divided into as many senses as the noun has, and each edge connects exactly two noun senses.

This representation can be built in linear time by first building an array indexed by senses of WordNet and processing a text sequentially, inserting a copy of each candidate word into

Semantic relation	1 sent.	3 sent.	1 par.	other
synonym	1	1	0.5	0.5
hypernym/hyponym	1	0.5	0.3	0.3
sibling	1	0.3	0.2	0

Table 1: Weights of relations depending on the kind of semantic relationship and distance (in number of sentences or paragraphs) between two words.

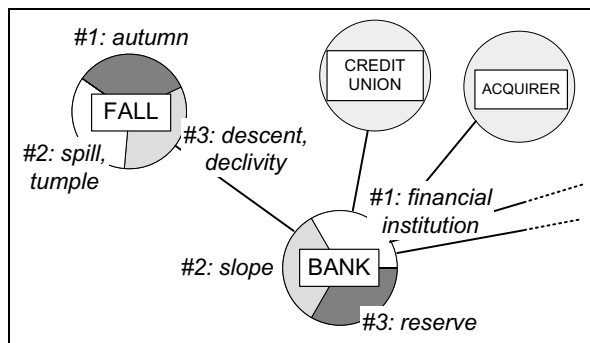


Figure 1: A *disambiguation graph*, an implicit representation of word-sense combinations (in this example, all weights are equal to 1.)

all entries that are valid senses of this word (for example, in Figure 2, the instances *car* and *auto* have been inserted under the same sense in the array). Then, we check whether the noun instance that was just inserted is semantically related to other nouns already present in the array. We do so by looking at hypernyms, hyponyms, and siblings, and if any of these related senses have non-empty entries in the array, we create the appropriate links in the disambiguation graph. For example, in Figure 2, the algorithm finds an hyponymy relation between the noun *taxi* (under its unique sense in the array) and another entry in the array containing *car* and *auto*, so semantic links are added to the disambiguation graph between these two words and *taxi* (shown here attached to the array). Processing all nouns this way, we can create all semantic links in the disambiguation graph in time $O(n)$ (where n is the number of candidate words.)

In the second step, we use the disambiguation graph to perform WSD, enforcing the constraint of one sense per

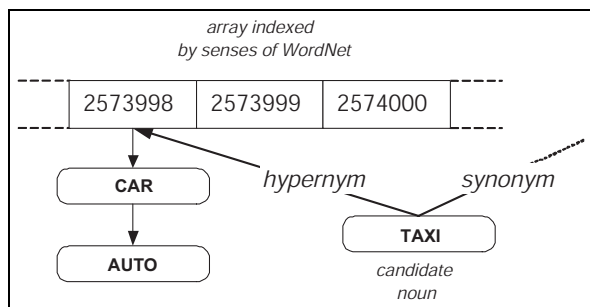


Figure 2: First pass of the algorithm: using an array, we can build the disambiguation graph in linear time.

discourse. We perform the disambiguation of every *word*, instead of disambiguating *word occurrences* as in e.g. [Hirst and St-Onge, 1998; Silber and McCoy, 2003]. We process all occurrences (nodes) of one word at a time, and sum the weight of all edges leaving these nodes under their different senses. The one sense of the word that is assigned the highest score (sum of weights) is considered the most probable sense. For example in Figure 1, the sense of *bank* that has the highest score is *financial institution*. That sense is assigned to all occurrences of the word; in other words, we *impose* the constraint of one sense per discourse. In case of a tie between two or more senses, we select the one sense that comes first in WordNet, since WordNet orders the senses of a word by decreasing order of frequency.

The final step is to build the actual lexical chains by processing the entire disambiguation graph. At this point, we have already assigned a sense to each word, so the last step is to remove from the disambiguation graph all semantic links that connect words taken under their (assumed) wrong senses. Once all such edges have been removed, we are left with the semantic links corresponding to a unique interpretation of the text, and the edges that remain in the graph are the actual lexical chains of our algorithm.¹

The separation of WSD and lexical chaining into two different sub-tasks is important. All semantic relations, whether correct or incorrect, can be investigated in WSD without necessarily creating incorrect semantic relations in the chaining process. Words are disambiguated by summing weights of semantic relations, but mistakenly counting edges relating words under wrong senses (as in Figure 2 between *fall* and *bank*) doesn't necessarily have the undesirable effect of linking the two words in the same chain. Our assumption is that summing edge weights generally helps in selecting the right senses, e.g. *bank* is disambiguated as a financial institution, and *fall* and *bank* are thus prevented from appearing in the same chain.

3 Evaluation

The evaluation of lexical chains is generally difficult. Even if they can be effectively used in many practical applications like automatic summarization, topic segmentation, and others, lexical chains are seldom desirable outputs in a real-world application, and it is unclear how to assess their quality independently of the underlying application in which they are used. For example, in summarization, it is hard to determine whether a good or bad performance comes from the efficiency of the lexical chaining algorithm or from the appropriateness of using lexical chains in that kind of application. In this section, we evaluate lexical chaining algorithms on the basis of WSD. This arguably is independent of any targeted

¹Our algorithm has some similarities with Silber and McCoy's algorithm, but it is actually quite different. First, they process each noun instance separately; thus, nothing prevents a noun from having different senses in the same discourse. Second, they process the entire text twice instead of once. In the second pass of their algorithm, they perform WSD and the actual chaining at the same time, whereas we postpone the chaining process until each word has been fully disambiguated.

Algorithm	Accuracy
Barzilay and Elhadad	56.56%
Silber and McCoy	54.48%
Galley and McKeown	62.09%

Table 2: Accuracy of noun disambiguation on semcor.

application, since *any* lexical chaining algorithm has to deal with the problem of WSD. We do not attempt to further evaluate other aspects of chains.

We tested three lexical chaining algorithms on the semantic concordance corpus (semcor), a corpus that was extracted from the Brown Corpus and semantically tagged with WordNet senses. We limited our evaluation to a set of 74 documents of that corpus; this represents about 35,000 nouns. WSD was evaluated on nouns, since all three algorithms that were tested (Barzilay and Elhadad; Silber and McCoy, and ours) build lexical chains with nouns only. We used the original implementation of Barzilay and Elhadad's algorithm, but had to implement Silber and McCoy's algorithm since we didn't have access to their source code. We tested the accuracy of WSD on the set of 35,000 nouns and obtained the results presented in Table 2;² accuracy by polysemy is displayed in Figure 3. We can see that our algorithm outperforms Barzilay and Elhadad's, and a one-sided t-test³ of the null hypothesis of equal means shows significance at the .001 level ($p = 4.52 \cdot 10^{-4}$). Barzilay and Elhadad in turn outperform Silber and McCoy, but this result is not significant at the basic .05 level ($p = 0.0537$).

4 Conclusions

In this paper, we presented an efficient linear-time algorithm to build lexical chains, showing that one sense per discourse can improve performance. We explained how the separation of WSD from the construction of the chains enables a simplification of the task and improves running time. The evaluation of our algorithm against two known lexical chaining algorithms shows that our algorithm is more accurate when it chooses the senses of nouns to include in lexical chains. The implementation of our algorithm is freely available for educational or research purposes at <http://www.cs.columbia.edu/~galley/research.html>.

Acknowledgments

We thank Regina Barzilay, the three anonymous reviewers, and the Columbia natural language group members for helpful advice and comments.

²In Barzilay and Elhadad's algorithm, a word can sometimes belong to two different chains. In order to map each word to one single sense, we applied the *strong chain* sense disambiguation strategy described in [Barzilay, 1997] (i.e. picking the word sense used in the strongest chain).

³The samples in the t-test are the WSD accuracies on each individual documents.

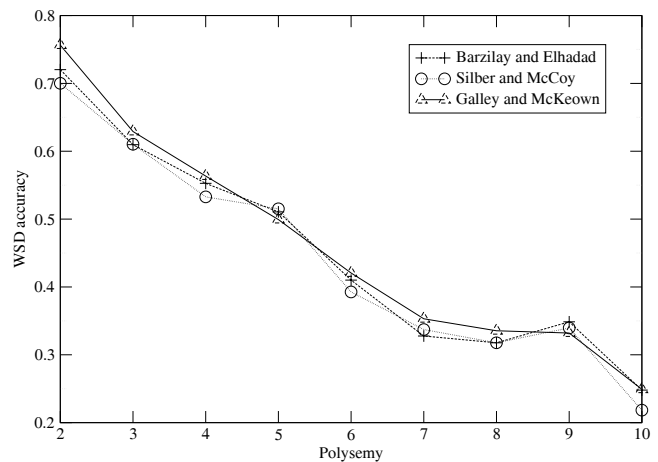


Figure 3: Accuracy by polysemy of the three algorithms.

References

- [Al-Halimi and Kazman, 1998] R. Al-Halimi and R. Kazman. Temporal indexing of video through lexical chaining. In *WordNet: An electronic lexical database*. MIT Press, 1998.
- [Barzilay and Elhadad, 1997] R. Barzilay and M. Elhadad. Using lexical chains for text summarization. In *Proc. of the Intelligent Scalable Text Summarization Workshop (ISTS'97), ACL*, 1997.
- [Barzilay, 1997] R. Barzilay. Lexical chains for summarization. Master's thesis, Ben-Gurion University, 1997.
- [Gale et al., 1992] W. Gale, K. Church, and D. Yarowsky. One sense per discourse. In *Proc. of the DARPA Speech and Natural Language Workshop*, 1992.
- [Green, 1998] S. Green. Automated link generation: Can we do better than term repetition? In *Proc. of the 7th International World-Wide Web Conference*, 1998.
- [Halliday and Hasan, 1976] M. Halliday and R. Hasan. *Cohesion in English*. Longman, London, 1976.
- [Hirst and St-Onge, 1998] G. Hirst and D. St-Onge. Lexical chains as representations of context for the detection and correction of malapropisms. In *WordNet: An electronic lexical database*. MIT Press, 1998.
- [Hoey, 1991] M. Hoey. *Patterns of lexis in text*. Oxford University Press, 1991.
- [Kan et al., 1998] M.-Y. Kan, J. Klavans, and K. McKeown. Linear segmentation and segment significance. In *Proc. of the 6th Workshop on Very Large Corpora (WVLC-98)*, 1998.
- [Miller, 1990] G. Miller. WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–312, 1990.
- [Silber and McCoy, 2003] G. Silber and K. McCoy. Efficiently computed lexical chains as an intermediate representation for automatic text summarization. *Computational Linguistics*, 29(1), 2003.

A Revised Algorithm for Latent Semantic Analysis

Xiangen Hu, Zhiqiang Cai, Max Louwerse, Andrew Olney, Phanni Penumatsa, Art Graesser, and TRG

Department of Psychology, The University of Memphis, Memphis, TN 38152

Abstract

The intelligent tutoring system AutoTutor uses latent semantic analysis to evaluate student answers to the tutor's questions. By comparing a student's answer to a set of expected answers, the system determines how much information is covered and how to continue the tutorial. Despite the success of LSA in tutoring conversations, the system sometimes has difficulties determining at an early stage whether or not an expectation is covered. A new LSA algorithm significantly improves the precision of AutoTutor's natural language understanding and can be applied to other natural language understanding applications.

1 Introduction

The use of intelligent technology in education is on the rise. Intelligent tutoring systems, once restricted to artificial intelligence labs at major universities, are migrating to mainstream schools [Koedinger *et al.*, 1997]. Intelligent tutoring systems (ITS) in this environment face a difficult challenge: to understand the student and manage the tutoring session in the face of vague or ungrammatical input. For most ITSs, language understanding and dialog management are core components. Particularly in a classroom, these systems live or die by their ability to understand what the student is trying to say. One technique, Latent Semantic Analysis has been successfully developed for such purposes [Landauer *et al.*, 1998b; Landauer and Dumais, 1997; Foltz *et al.*, 1998; Graesser *et al.*, 2002]. LSA, a statistical technique utilizing unsupervised learning, is both highly portable to other domains and adept at recognizing vague or incomplete input.

This paper outlines some problems inherent in the traditional LSA algorithm and solutions to this problem by using a different LSA algorithm. Not only does this new algorithm increase the precision of ITS language understanding, but it also offers a new perspective on a commonly used technique in cognitive science, computational linguistics and artificial intelligence.

1.1 Latent Semantic Analysis

Latent Semantic Analysis is a statistical, corpus-based language understanding technique that estimates similarities on

a scale of -1 to 1 between the latent semantic structure of terms and texts [Deerwester *et al.*, 1990]. The input to LSA is a set of corpora segmented into documents. These documents are typically paragraphs or sentences. Mathematical transformations create a large term-document matrix from the input. For example, if there are m terms in n documents (usually m and n are very large, for now, assume $n \geq m$), then a matrix of $\mathbf{A} = f_{ij} \times G(j) \times L(i, j)_{m \times n}$ is obtained. The value of f_{ij} is a function of the integer that represents the number of times term i appears in document j . $L(i, j)$ is a local weighting of term i in document j and $G(i)$ is the global weighting for term i . Such a weighting function is used to differentially treat terms and documents to reflect knowledge that is beyond the collection of the documents. This matrix of A has, however, lots of redundant information. Singular value decomposition reduces this noise by linearly decomposing the matrix $\mathbf{A}$ into three matrices $\mathbf{A} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T$; $\mathbf{U}$ is $m \times m$ and $\mathbf{V}$ is $n \times n$ square matrices, such that $\mathbf{U} \mathbf{U}^T = \mathbf{I}_{m \times m}$; $\mathbf{V} \mathbf{V}^T = \mathbf{I}_{n \times n}$, and $\mathbf{\Sigma}$ is $m \times n$ diagonal matrix with singular values on the diagonal. By removing dimensions corresponding to small singular values and keeping the dimensions corresponding to larger singular values, the representation of each term is further reduced to a smaller vector with only k dimensions. The new representation for the terms (the reduced $\mathbf{U}$ matrix) are no longer orthogonal, but the advantage of this is that only the most important dimensions that correspond to larger singular values are kept. This method of statistically representing knowledge has proven to be useful in a range of studies. For instance, studies have shown that LSA performs as well as students on TOEFL (test of English as a foreign language) tests [Landauer and Dumais, 1997], that it grades essays as reliably as humans [Landauer *et al.*, 1998a] and that it reliably measures the coherence between a sentence and successive sentences [Foltz *et al.*, 1998]. Finally, LSA has successfully been used in intelligent tutoring systems like AutoTutor [Graesser *et al.*, 2002; 1999].

1.2 AutoTutor

AutoTutor is a conversational agent that assists students in actively constructing knowledge by holding a conversation in natural language with them [Graesser *et al.*, 2001]. In addition to latent semantic analysis, at least four other com-

ponents can be distinguished: 1) a dialog management system guides the student through the tutor-student conversation. Fuzzy production rules and a Dialog Advancer Network form the basis of these conversational strategies; 2) curriculum scripts organize the pedagogical macrostructure of the tutorial. These scripts keep track of the topic coverage and follow up on any problems the student might have; 3) a talking head with facial expressions and synthesized speech is used for the interface. Parameters of the facial expressions and rudimentary gestures are generated by fuzzy production rules; 4) mixed-initiative dialog, including the appropriate use of discourse markers to make the conversation smoother; a speech act classifier that accounts for the pragmatics of incoming expressions; and a question answering tool that dynamically answers student questions on a variety of topics.

AutoTutor was originally developed for computer literacy. Over the last two years a web version of AutoTutor was developed that tutors in conceptual physics. In both domains world knowledge is provided by LSA spaces of domain specific text books.

2 An improved LSA algorithm

At the beginning of a session, AutoTutor presents a question to the student, and the student responds to this question. AutoTutor analyzes the accuracy of this answer by comparing the student's answer with a series of expected ideal answers. Over the course of the tutoring session, the student covers each of the expectations. The tutor allows the student to move to the next problem only once all expectations are covered.

Although possible in theory, a single contribution from a student usually does not cover all expectations at once. Instead, the student simply types one sentence at a time in the conversation with the tutor. Based on the student's responses, the tutor will then provide appropriate feedback based on the quality of responses. To provide feedback to a student's contributions, the tutoring system needs to know the following:

1. information related to the expected answer elements that is 1) new (what was not in the previous contributions); 2) old (what was in the previous contributions)
2. information not related to the expected answer elements that is 1) new (what was not in the previous contributions); 2) old (what was in the previous contributions)

Depending on these four components, AutoTutor chooses the most appropriate feedback. This mechanism for student contributions is illustrated in Table 1. For example, AutoTutor needs to provide highly positive feedback when students provide new relevant information (cell labeled ++). For relevant but repeated information (cell labeled +), AutoTutor needs to provide only some non-negative feedback. For irrelevant contributions, AutoTutor needs to point out the repeated misconceptions, eventually with negative feedback (cell with - -). The system returns non-positive feedback in cases of irrelevant information occurring the first time (cell with -).

The challenge for AutoTutor is to obtain information that belongs to each of the cells in the table. That is, the system needs to take into account both the relevance of the information and whether or not the information is new.

Type of feedback	relevant	irrelevant
New	++	-
Old	+	--

Table 1: Four types of feedback the system provides on the basis of relevance and newness of student contribution.

- **Question:** Suppose a runner is running in a straight line at constant speed and throws a pumpkin straight up. Where will the pumpkin land? Explain why.
- **Expectation:** The pumpkin will land in the runner's hands.
- **Student contributions:**
 - (1) I think, correct me if I am wrong, it will not land before or behind the man.
 - (2) The reason is clear, they have the same horizontal speed.
 - (3) The pumpkin will land in the runner's hand.
 - (4) Did I say anything wrong?
 - (5) Come on, I thought I have said that.

	Old LSA	New Infor.	New contribution	New LSA
(1)	0.431	100%	0.431	0.431
(2)	0.430	99%	0.175	0.466
(3)	0.751	88%	0.885	1.0
(4)	0.713	98%	0.000	1.0
(5)	0.667	97%	0.000	1.0

Table 2: Example of student contribution and evaluation based on two LSA methods

In earlier versions of the system, AutoTutor put all the student contributions (from the first response to the most recent response) together as one document and would then compare with the expectation. One of the reasons for this was that it has often been claimed that the best performance in LSA comes from paragraphs rather than sentences (see [Foltz *et al.*, 1998]). However, simple vector algebra shows that vector summation of term-vectors for the combined contributions may in fact reduce the similarity between the expectation and contributions when contributions are added. This reduction is evident in the example given in second column of Table 2. The tutor asks the student a question at the start of a conceptual physics problem. The student's answer is matched with an ideal expected answer. The question now is what happens to the LSA coverage scores if the student submits (new/old) (relevant/irrelevant) multiple contributions.

From the expected answer we know that a student's answer like (1) is almost correct. Now imagine that the student answers (2). The cosine match drops, resulting in AutoTutor asking for more information. Now assume that the student also answers (3), which is the exact ideal answer. Using a traditional vector addition algorithm, the similarity is not 1.0. This loss of precision results from the noise introduced by the irrelevant information in the student's answers. By adding the contributions' vectors, the system cannot distinguish be-

tween the different parts (new/old) (relevant/irrelevant) of the student's contributions. So although LSA effectively compares semantic similarities between two large documents, LSA lacks precision for comparing smaller documents in a progressive sequence. Under a vector-addition model, a student whose answers improve dramatically over the course of the tutoring session is "penalized" for an initial bad answer. To solve this limitation, we propose an alternative solution. Instead of simply combining contributions into a larger document, each contribution is treated as "independent" in a vector subspace. The combination of the contributions is then not represented as a simple vector summation, but instead as a span in the subspace. This way, the vector for any new contribution can be algebraically decomposed into two components. One component is the projection of the vector to the spanned subspace of the previous contributions, and the other component is perpendicular to the subspace. These two components of the most recent contribution correspond to relevant information (projection to the subspace) and new information (the perpendicular component). Finally, the cosine match of the new information with the expectation is the measure of new (additional) coverage of the expectations. We applied this method to the example such as Table 2 and observed desirable increase in LSA's precision, as illustrated in column 3, 4, and 5 of Table 2. The rows of Table 2 present the five student contributions. The column 'New Info' gives the percentage of new information for each of the contributions, compared to the previous contributions. The third column is the relevance of the new contribution to the span. The final two columns give the old and new LSA scores.

Although contribution (3) is identical to the expectation, it still is only 88% new. The reason is that contributions (1) and (2) already contain some of the information from (3). For example, although (2) contains 99% new information, it has only marginally (0.175) contributed as coverage. Notice that the quality in the subsequent student contributions does not deteriorate, but the old LSA values do. The new LSA values, on the other hand, account for additional relevant information, even bringing the coverage score to the maximum value of 1.

The method provided here can be used to compute all four cells in Table 1, because it differentiates whether the information is new or old, and whether it is relevant or not. Furthermore, since it provides information at every step, numerical information of the values can be used to provide secondary information for feedback. For example, the rate of increase in the new LSA algorithm provides us with information on the development student performance on a step-by-step basis. By being able to localize LSA scores, AutoTutor can now determine the effectiveness of its dialog moves.

The proposed new algorithm can potentially be used in applications like essay grading, where the student's composition covers the key elements for a given essay. The algorithm can measure development of student performance and can take into account whether information is old or new, relevant or irrelevant.

3 Conclusion

This paper addressed the use of latent semantic analysis in intelligent tutoring systems like AutoTutor. Despite the success of LSA in AutoTutor, previous versions were not able to differentiate between relevant/irrelevant or new/old information in student contributions. Replacing the vector-addition based algorithm with a span-based algorithm does not only improve AutoTutor's evaluation of student contributions, but is most likely to improve LSA performance in a wide range of other natural language understanding applications.

References

- [Deerwester *et al.*, 1990] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society For Information*, pages 391–407, 1990.
- [Foltz *et al.*, 1998] P. W. Foltz, W. Kintsch, and T. K. Landauer. The measurement of textual coherence with latent semantic analysis. *Discourse Processes*, pages 285–307, 1998.
- [Graesser *et al.*, 1999] A. C. Graesser, K. Wiemer-Hastings, P. Wiemer-Hastings, R. Kreuz, and the Tutoring Research Group. AutoTutor: A simulation of a human tutor. *Journal of Cognitive Systems Research*, 1:35–51, 1999.
- [Graesser *et al.*, 2001] A.C. Graesser, N. Person, D. Harter, and TRG. Teaching tactics and dialog in AutoTutor. *International Journal of Artificial Intelligence in Education*, 12:257–279, 2001.
- [Graesser *et al.*, 2002] A. C. Graesser, X. Hu, B. A. Olde, M. Ventura, A. Olney, M. Louwerse, D. R. Franceschetti, and N. Person. Implementing latent semantic analysis in learning environments with conversational agents and tutorial dialog. In *Proceedings of the 24th Annual Meeting of the Cognitive Science Society*, page 37. Mahwah, NJ: Erlbaum, 2002.
- [Koedinger *et al.*, 1997] K.R. Koedinger, J.R. Anderson, W.H. Hadley, and M.A. Mark. Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, 8:30–43, 1997.
- [Landauer and Dumais, 1997] T. K. Landauer and S. T. Dumais. A solution to plato's problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review*, pages 211–240, 1997.
- [Landauer *et al.*, 1998a] T. K. Landauer, P. W. Foltz, and D. Laham. Introduction to latent semantic analysis. *Discourse Processes*, pages 259–284, 1998.
- [Landauer *et al.*, 1998b] T. K. Landauer, D. Laham, and P. W. Foltz. Learning human-like knowledge by singular value decomposition: A progress report. In M. J. M. I. Jordan, Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems*, pages 45–51. Cambridge: MIT Press, 1998.

Identifying Synonyms among Distributionally Similar Words

Dekang Lin and Shaojun Zhao
Department of Computing Science
University of Alberta
Edmonton, AB, Canada, T6G 2E8

Lijuan Qin and Ming Zhou
Microsoft Research Asia
No.49, Zhichun Road
Hai Dian District, Beijing, China, 100080

Abstract

There have been many proposals to compute similarities between words based on their distributions in contexts. However, these approaches do not distinguish between synonyms and antonyms. We present two methods for identifying synonyms among distributionally similar words.

1 Introduction

The **distributional hypothesis** states that words with similar meanings tend to appear in similar contexts [Harris, 1968]. Consider the words *adversary* and *foe*. Both of them are often used as the objects of the verbs:

batter, crush, defeat, demonize, deter, outsmart, . . .

and modified by the adjectives:

ardent, bitter, formidable, old, tough, worthy, . . .

There have been many proposals for computing distributional similarity of words [Hindle, 1990; Pereira *et al.*, 1993; Lin, 1998]. The list (1) shows the top-20 distributionally similar words of *adversary*, obtained with Lin's method [Lin, 1998] on a 3GB newspaper corpus.

- (1) *adversary*: enemy, foe, ally, antagonist, opponent, rival, detractor, neighbor, supporter, competitor, partner, trading partner, accuser, terrorist, critic, Republican, advocate, skeptic, challenger

Compared with manually compiled thesauri, distributionally similar words often offer much better coverage. Compare (1) with the entry for *adversary* in Webster Collegiate Thesaurus [Kay, 1988]:

- (2) *adversary*: **Synonyms**: opponent, antagonist, anti, con, match, opposer, oppugnant; **Related Words**: assaulter, attacker **Contrasted Words**: backer, supporter, upholder; **Antonyms**: ally

The thesaurus entry missed many synonyms such as: *enemy*, *foe*, *rival*, *competitor* and *challenger*.

A problem with the distributionally similar words, however, is that many of them are antonyms, e.g., *ally* and *supporter* in (1). The problem gets worse if a word belongs to a semantic category with many members, since all of them tend to have similar distributions. This is demonstrated in

the following list of top-20 distributionally similar words for *orange*.

- (3) *orange*: yellow, lemon, peach, pink, lime, purple, tomato, onion, mango, lavender, avocado, red, pineapple, pear, blue, plum, cucumber, melon, turquoise, tangerine

In many applications, such as information retrieval and machine translation, the presence of antonyms or other types of semantically incompatible words (e.g., orange-pink) can be devastating. This paper presents two methods for identifying synonyms among distributionally similar words.

2 Methods

2.1 Patterns of Incompatibility

Consider the following phrasal patterns:

- (4) a. from *X* to *Y*
b. either *X* or *Y*

If two words *X* and *Y* appear in one of these patterns, they are very likely to be semantically incompatible. For example, the following table shows the queries and the hits (the number of return documents) from the search engine AltaVista:

(5)

Query	Hits
adversary NEAR ally	2469
"from adversary to ally"	8
"from ally to adversary"	19
"either adversary or ally"	1
"either ally or adversary"	2
adversary NEAR opponent	2797
"from adversary to opponent"	0
"from opponent to adversary"	0
"either adversary or opponent"	0
"either opponent or adversary"	0

Given a query *x NEAR y*, AltaVista returns documents where the words *x* and *y* appear close to each other. When two words are unrelated, the hits for the NEAR query tend to be low.

Motivated by the above examples, we propose to identify semantically incompatible word pairs by searching on the Web for instantiations of the patterns in (4). We define a score:

$$(6) \text{score}(x, y) = \frac{\text{hits}(x \text{ NEAR } y)}{\sum_{\text{pat} \in P} \text{hits}(\text{pat}(x, y)) + \epsilon}$$

where $\text{hits}(\text{query})$ is the number of hits returned by AltaVista for the *query*, P is the set of patterns in (4) and ϵ is a small constant to prevent the denominator of the above formula to be 0 (we set $\epsilon=0.0001$). The lower the score, the less likely that the words x and y are synonyms. To determine whether or not distributionally similar words x and y are synonyms, we compute $\text{score}(x, y)$. If the value is higher than $\theta=2000$, (x, y) is classified as a pair of synonyms.

2.2 Using Bilingual Dictionaries

The second method is based on the observation that translations of a word from another language are often synonyms of one another. For example, (7) contains the English translations of the French word *défenseur*. Many of them are synonyms.

- (7) advocate, attorney, counsel, fullback, intercessor, lawyer

When two such words are not synonyms, the reason is typically that the French word have multiple senses and the English words are translations of the French word in different senses. Under such circumstances, the distribution of the English words are usually quite different (e.g., *lawyer* and *fullback* appear in very different contexts). We can therefore identify synonyms of a word w by intersecting the set of words that share with w the same French (or any other language) translation and the set of distributionally similar words of w . For example, the top-20 distributionally similar words of *lawyer* are:

- (8) *lawyer*: attorney, counsel, prosecutor, doctor, official, judge, executive, manager, investigator, consultant, aide, agent, physician, expert, banker, officer, politician, lobbyist, teacher, accountant

The intersection of (7) and (8) gives us the synonyms of *lawyer*: *attorney* and *counsel*.

Since this method generally has high precision and low recall (see the next section), we can use this method with multiple bilingual dictionaries separately and take the union of the results. In our experiments, we used 7 dictionaries from <http://www.freewaresite.com/onldict/>

English-Swedish, English-Spanish, English-Japanese, English-German, English-French and English-Esperanto.

3 Evaluation

Using the algorithm in [Lin, 1998] on a 3GB newspaper corpus, we computed the distributional similarity between about 45,000 words. We randomly selected 80 pairs of synonyms and 80 pairs of antonyms from Webster's Collegiate Thesaurus [Kay, 1988] that are also among the top-50 distributionally similar words of each other. We then used the methods presented in the previous section to determine which pairs are synonyms. Let S be the set of true synonym pairs and S' be the set of pairs classified as synonyms. The precision and recall measures are defined as follows:

$$\text{precision} = \frac{|S \cap S'|}{|S'|} \quad \text{recall} = \frac{|S \cap S'|}{|S|}$$

The table in (9) shows the evaluation results:

(9)

Method	Precision	Recall
Pattern-based	86.4%	95.0%
Bilingual Dictionaries	93.9%	39.2%

4 Related Work

The problem we address here is related to semantic orientation. The **semantic orientation** of a word is positive (or negative) if it is generally associated with good (or bad) things. For example, The words *simple* and *simplistic* have similar meanings, but *simplistic* has a negative semantic orientation. The algorithm in [Hatzivassiloglou and McKeown, 1997] is based on the fact that conjoined adjectives generally have the same orientation. They use a small set of adjectives with known orientation to determine the orientations of other adjectives. [Turney, 2002] computed the degree of positive or negative semantic orientation of a word w with the hit counts from AltaVista for the queries $w \text{ NEAR excellent}$ and $w \text{ NEAR poor}$. While semantic orientation is bipolar, the problem we are dealing with is multipolar. For example, Turney's method is not able to tell that the words *red*, *orange*, *yellow*, *green*, ... have incompatible meanings.

5 Conclusion

Distributionally similar words include many antonyms and other semantically incompatible words, which minimizes their use in many applications. We have presented two methods for identifying synonyms among distributionally similar words. Our preliminary evaluation with known synonyms and antonyms extracted from Webster Collegiate Thesaurus has produced promising results.

References

- [Harris, 1968] Zelig S. Harris. *Mathematical Structures of Language*. Wiley, New York, 1968.
- [Hatzivassiloglou and McKeown, 1997] Vasileios Hatzivassiloglou and Kathleen R. McKeown. Predicting the semantic orientation of adjectives. In *Proceedings of ACL/EACL-97*, pages 174–181, Madrid, Spain, July 1997.
- [Hindle, 1990] Donald Hindle. Noun classification from predicate-argument structures. In *Proceedings of ACL-90*, pages 268–275, Pittsburg, Pennsylvania, June 1990.
- [Kay, 1988] Maire Weir Kay, editor. *Webster's Collegiate Thesaurus*. Merrian-Webster, 1988.
- [Lin, 1998] Dekang Lin. Automatic retrieval and clustering of similar words. In *Proceedings of COLING/ACL-98*, pages 768–774, Montreal, 1998.
- [Pereira et al., 1993] F. Pereira, N. Tishby, and L. Lee. Distributional Clustering of English Words. In *Proceedings of ACL-93*, pages 183–190, Ohio State University, Columbus, Ohio, 1993.
- [Turney, 2002] Peter D. Turney. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings of ACL-02*, pages 417–424, Philadelphia, July 2002.

A Logic Prover for Text Processing

Dan Moldovan and Christine Clark

Language Computer Corporation

Richardson, Texas

{moldovan,christine}@languagecomputer.com

Abstract

This paper demonstrates the applicability of automated reasoning to text processing, specifically to Question Answering. It is shown that the approach is feasible, effective, and scalable. A Logic Prover has been implemented and integrated into a state-of-the-art Question Answering System.

1 Introduction

Automated reasoning has been applied to many application areas [Wos 1988], but little work was done towards applying it to text processing. The challenges faced when using automated reasoning for text processing are: logic representation of open text, need of world knowledge axioms, logic representation of semantically equivalent linguistic patterns, and others.

In this paper we argue that considerable progress was made in key areas of natural language processing and it is possible to implement logic provers for some specific NLP applications. One such application is Question Answering where the problem is to find exact answers to questions expressed in natural language by searching a large collection of documents [Voorhees 2002]. We have implemented a Logic Prover for QA, called COGEX. Integrated into a QA system, the Prover captures the syntax-based relationships such as the syntactic objects, syntactic subjects, prepositional attachments, complex nominals, and adverbial/adjectival adjuncts provided by the logic representation of text. In addition to the logic representations of questions and candidate answers, the QA Logic Prover uses world knowledge axioms to link questions to answers. These axioms are provided by the WordNet glosses represented in logic forms. Additionally, the prover implements rewriting procedures for semantically equivalent lexical patterns. With this deep and intelligent representation, COGEX effectively and efficiently re-ranks candidate answers by their correctness and ultimately eliminates incorrect answers. In this way, the Logic Prover is a powerful tool in boosting the accuracy of the QA system. Moreover, the trace of a proof constitutes a justification for that answer.

2 Logic Prover Inputs and Outputs

A logic form provides a one to one mapping of the words of the text into the first order logic predicates. The pred-

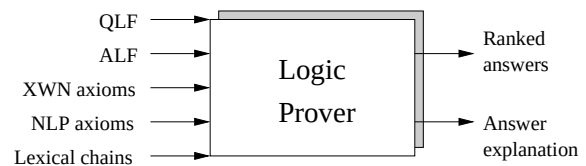


Figure 1: COGEX Architecture

icate names consist of the base form of the word concatenated with the part of speech of the word [Moldovan and Rus 2001]. An important feature of the Logic Form representation is the fixed-slot allocation mechanism of the verb predicates [Hobbs 1993].

The term Question Logic Form (QLF) refers to the questions posed to the Question Answering system in logic form.

Question: *Which company created the Internet Browser Mosaic ?*

QLF: `_organization_AT(x2) & company_NN(x2) & create_VB(e1,x2,x6) & Internet_NN(x3) & browser_NN(x4) & Mosaic_NN(x5) & nn_NNC(x6,x3,x4,x5)`

The term Answer Logic Form (ALF) refers to the candidate answers in logic form. Candidate answers are provided by the Question Answering system [Moldovan 2002].

Answer: *In particular, a program called Mosaic , developed by the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana - Champaign , is gaining popularity*

ALF: `In_IN(x1,x28) & particular_JJ(x29) & program_NN(x1) & call_VB(e1,x27,x30) & Mosaic_NN(x2) & develop_VB(e2,x2,x31) & by_IN(e2,x8) & National_NN(x3) & Center_NN(x4) & for_NN(x5) & Supercomputing_NN(x6) & application_NN(x7) & nn_NNC(x8,x3,x4,x5,x6,x7) & NCSA_NN(x9) & at_IN(e2,x15) & University_NN(x10) & of_NN(x11) & Illinois_NN(x12) & at_NN(x13) & Urbana_NN(x14) & nn_NNC(x15,x10,x11,x12,x13,x14) & Champaign_NN(x16) & gain_VB(e3,x1,x17) & popularity_NN(x17) ...`

Logic representation of WordNet glosses

A major problem in QA is that often an answer is expressed in words different from those in the question. World knowledge is necessary to conceptually link questions to answers. WordNet glosses contain a source of world knowledge. To be

useful in automated reasoning, the glosses need to be transformed into logic forms. Taking the same approach as for open text, we have parsed and represented in logic forms more than 50,000 WordNet glosses.

The question contains the verb *create* while the answer contains the verb *develop*. In order to prove that this answer is in fact correct, we need to detect and use a lexical chain between *develop* and *create*. WordNet supplies us with that chain such that: *develop* $\leftrightarrow$ *make* and *make* $\leftrightarrow$ *create*.

NLP Axioms

In addition to world knowledge axioms, a QA Logic Prover requires linguistic knowledge. This is what distinguishes an NLP prover from a traditional mathematical prover. General axioms that reflect equivalence classes of linguistic patterns need to be created and instantiated when invoked. We call these NLP axioms and show some examples below. More details are presented in [Moldovan 2003].

1. *Complex nominals and coordinated conjunctions.* An axiom is built such that the head noun of the complex nominal in the question implies the entire noun phrase:

all x_1 ($\text{mosaic_nn}(x_1) \rightarrow \text{internet_nn}(x_1) \ \& \ \text{browser_nn}(x_1) \ \& \ \text{mosaic_nn}(x_1)$)

2. *Appositions.* An apposition implies that the two noun phrases in the apposition “stand for” each other. “..Italian Andrea Pfister , designer of the 1979 ” bird cage ” shoe... uses an apposition to describe the designer. An axiom is built to link the head of the noun phrases in the apposition such that they share the same argument.

all $x_{12} \ x_{13} \ x_{14} \ x_{15} \ x_{16} \ x_{17} \ x_{18} \ x_{19}$ ($\text{italian_nn}(x_{12}) \ \& \ \text{andrea_nn}(x_{13}) \ \& \ \text{pfister_NN}(x_{14}) \ \& \ \text{nn_nnc}(x_{15}, x_{12}, x_{13}, x_{14}) \ \& \ \text{designer_nn}(x_{16}) \ \& \ \text{of_in}(x_{16}, x_{17}) \ \& \ \text{1979_nn}(x_{17}) \ \& \ \text{bird_nn}(x_{18}) \ \& \ \text{cage_nn}(x_{19})$) $\rightarrow \text{designer_nn}(x_{15}) \ \& \ \text{of_in}(x_{15}, x_{17})$)

3. *Part-of relations in location questions.* A location seeking question may have a candidate answer that identifies a location by referring to a part of the location. For example, an axiom is built to connect Wyoming to its part:

all $x_1 \ x_2 \ x_3$ ($\text{corner_nn}(x_1) \ \& \ \text{of_in}(x_1, x_2) \ \& \ \text{wyoming_nn}(x_2) \rightarrow \text{wyoming_nn}(x_1)$)

4. *Attribute of relations in quantity seeking questions.* An axiom is built to connect the quantity to its subject, redwood:

all $x_1 \ x_2$ ($\text{quantity_}(x_1) \ \& \ \text{redwood_NN}(x_2) \rightarrow \text{of_in}(x_1, x_2)$)

5. *Translating words from noun form to verb form.* Axioms are built to link nouns with verbs. For example the noun *seizure* is linked to the verb *seize*, by filling the object with the tail of the predicate attached to seizure, and assigning the event argument to be the argument of the noun predicate for seizure.

all $x_7 \ x_8 \ x_9$ ($\text{its_prp}(x_8, x_7) \ \& \ \text{seizure_nn}(x_8) \ \& \ \text{of_in}(x_8, x_9) \rightarrow \text{seize_vb}(x_8, x_7, x_9)$)

3 Control Strategy

The search strategy used is the Set of Support Strategy [Wos 1988]. Axioms placed in the Usable list are: (1) Extended WordNet axioms, (2) NLP axioms, and (3) axioms based on outside world knowledge, such as people and organizations. The inference rule sets are based on hyperresolution and paramodulation.

When the proof fails, we devised a way to incrementally relax some of the conditions that hinder the completion of the proof. This relaxation process puts weights on the proof such that proofs weaker than a predefined threshold are not accepted.

4 Results

COGEX was implemented and integrated into a state-of-the-art Question Answering system that participated in TREC 2002 [Moldovan 2003]. All questions are attempted by the prover, but if the proof fails the QA system resorts to other answer extraction methods that were part of the system before the prover. Thus, some questions are answered by the QA system without the prover, some only by the prover and some by both the non-prover system and the prover. The complete system answered 415 questions out of 500 TREC 2002 questions. Of these, 206 were answered by COGEX. A careful analysis indicates that the QA system without logic prover answered 317 questions and the prover can answer only 98 additional questions for which the system without prover failed. Table 1 summarizes these results. The added

Total number of TREC questions	500
Questions answered by the complete system	415
Questions answered by COGEX	206
Questions answered only by COGEX	98
Questions answered without COGEX	317

Table 1: Performance over 500 TREC 2002 questions

value of automated reasoning to the QA system is 30.9% (or 98/317). The failures of the prover are due primarily to the lack of linguistic axioms.

Acknowledgment

This work was supported in part by the ARDA AQUAINT program.

References

- [Hobbs 1993] J. Hobbs, M. Stickel, and P. Martin. Interpretation as abduction. *Artificial Intelligence*, 63 (1993), pp 69-142.
- [Moldovan and Rus 2001] D. Moldovan and V. Rus. Logic Form Transformation and its Applicability to Question Answering. In *Proceedings of ACL 2001*.
- [Moldovan et. al. 2002] Dan Moldovan et al. Performance issues and error analysis in an open-domain question answering system. In *Proceedings of ACL 2002*.
- [Moldovan et. al. 2003] D. Moldovan, C. Clark, S. Harabagiu, S. Maiorano. COGEX: A Logic Prover for Question Answering. In *Proceedings of HLT 2003*.
- [Voorhees 2002] Ellen Voorhees. Overview of the TREC 2002 Question Answering Track. In *TREC 2002* <http://trec.nist.gov>
- [Wos 1988] Larry Wos. Automated Reasoning, 33 Basic Research Problems. *Prentice Hall*, 1988.

Inducing criteria for lexicalization parts of speech using the Cyc KB

Tom O'Hara

Computer Science Department
New Mexico State University
Las Cruces, NM 88001

tomohara@cs.nmsu.edu

Michael Witbrock, Bjørn Aldag, Stefano Bertolo,

Nancy Salay, Jon Curtis and Kathy Panton

Cycorp, Inc.

Austin, TX 78731

{witbrock,bertolo,aldag,nancy,jonc,panton}@cyc.com

Abstract

We present an approach for learning part-of-speech distinctions by induction over the lexicon of the Cyc knowledge base. This produces good results (74.6%) using a decision tree that incorporates both semantic features and syntactic features. Accurate results (90.5%) are achieved for the special case of deciding whether lexical mappings should use count noun or mass noun headwords. Comparable results are also obtained using OpenCyc, the publicly available version of Cyc.

1 Introduction

In semantic lexicons, a *lexical mapping* describes the relationship between a concept and phrases used to refer to it. This mapping includes syntactic information for deriving variant phrases, including the part of speech of the headword. Selecting the part of speech for the lexical mapping is required so that proper inflectional variations can be recognized and generated for the term. Although often a straightforward task, there are special cases that can pose problems, especially when fine-grained speech part categories are used.

To reduce the need for linguistic expertise in producing these *lexicalizations* for a large knowledge base like Cyc [Lenat, 1995], linguistic criteria can be inferred from decisions that have been made by lexical knowledge engineers in *lexicalizing* preexisting terms.

The Cyc knowledge base, containing 120,000 concepts and over one million axioms,¹ divides roughly into three layers. The upper ontology formalizes fundamental distinctions (e.g., tangibility versus intangibility). The lower ontology collects specific facts, often related to concrete applications, and the middle ontology encodes commonsense knowledge about the world. The KB also includes a broad-coverage English lexicon mapping words and phrases to terms throughout the KB.

Natural language lexicons are integrated directly into the Cyc KB [Burns and Davis, 1999]. Binary predicates, as in (*nameString* *HEBCompany* "HEB"), map names to terms. A

¹These figures and the results discussed later are based on Cyc KB v576 and system v1.2577. See www.cyc.com for detailed documentation on the KB and [O'Hara et al., 2003] for more technical details related to this work.

denotational assertion maps a phrase, specified via a lexical concept with optional string modifiers, into a concept, usually a collection. The part of speech is specified by Cyc's *SpeechPart* constants. The simplest type of denotational mapping uses the *denotation* predicate. For example, (*denotation* *Device-Word CountNoun 0 PhysicalDevice*) indicates that sense 0 of the count noun 'device' refers to *PhysicalDevice* (via the wordforms "device" and "devices"). Three additional predicates account for phrasal mappings: *compoundString*, *head-MedialString*, and *multiWordString* are used for phrases with the headword at the beginning, the middle, and the end, respectively.

These denotational assertions, excluding lexical mappings for technical, informal and slang terms, form the training data for a lexicalization speech part classifier.

2 Inference of lexicalization part of speech

Our method of inferring the part of speech for lexicalizations is to apply decision tree learning over existing lexical mappings. For each target denotatum term, the corresponding definitional information (e.g., *isa*), asserted or inferable via transitivity, is extracted from the ontology. For simplicity, these definitional types are referred to as *ancestor terms*. Associations between the lexicalization parts of speech and common ancestor terms underlie the lexicalization speech part classifier and its special case, the mass-count classifier (distinguishing, e.g., "much *sand*" from "many *books*"). To reduce the size of the training feature vector, only the most frequent 256 atomic terms from the thousands of possible ancestor terms are selected, after excluding certain internal bookkeeping constants.

Given a training instance, such as a denotation from a lexeme into a specific Cyc concept using a particular *SpeechPart* (e.g., *MassNoun* or a *CountNoun*), the feature specification is derived by determining all the ancestor terms of the denotatum term and converting this into a vector of occurrence indicators, one indicator per reference term. Then the headword is checked for the occurrence of a set of commonly used suffixes. If found, the suffix itself is added to the vector (in a position set aside for suffixes of the same length). The part of speech serves as the classification variable.

We use decision trees for this classification. Part of the motivation is that the result is readily interpretable and can be incorporated directly by knowledge-based applications. A

	OpenCyc	Cyc
Instances	2607	30676
Classes	2	2
Entropy	0.76	0.90
Baseline	78.3	68.2
Accuracy	87.5	90.5

Table 1: Mass-count classification over Cyc lexical mappings, using Cyc reference terms and headword suffixes as features. *Instances* is size of the training data. *Classes* indicates number of choices. *Baseline* selects most frequent case. *Accuracy* is average in the 10-fold cross validation.

	OpenCyc	Cyc
Instances	3721	43089
Classes	16	33
Entropy	1.95	2.11
Baseline	54.9	49.0
Accuracy	71.9	74.6

Table 2: General speech part classification using Cyc.

simple fragment from the resulting decision tree shows how ontological features interact with morphological ones:

```

if (isa AbstractInformationalThing) then
  if (suffix = "-er") then
    if (not isa SomethingExisting) then AgentiveNoun
    if (isa SomethingExisting) then MassNoun
  if (suffix = "-ed") then MassNoun
  if (suffix = "-al") then Adjective
  if (suffix = "-or") then AgentiveNoun

```

Table 1 shows the results of 10-fold cross validation for the mass-count classification. This was produced using the J48 algorithm in the Weka machine learning package [Witten and Frank, 1999], which is an implementation of Quinlan’s C4.5 [Quinlan, 1993] decision tree learner. This shows that the system achieves an accuracy of 90.5%, an improvement of 22.3 percentage points over a baseline of always selecting the most frequent case. The OpenCyc version of the classifier also performs well. This suggests that sufficient data is already available in OpenCyc (available online at www.opencyc.org) to allow for good approximations for such classifications.

The mass/count noun distinction can be viewed as a special case of speech part classification. Running the same classifier using the full set of speech part classes yields the results shown in Table 2. Here the overall result is not as high, but there is a similar improvement over the baseline.

3 Discussion

Contextual part of speech tagging [Brill, 1995] has received substantial attention in the literature, but there has been relatively little written on automatically determining default lexicalization parts of speech. Woods [Woods, 2000] describes an approach to this problem using manually-constructed rules incorporating syntactic, morphological, and semantic tests

(via an ontology). Our rules are induced from the knowledge base, which alleviates the need for rule maintenance as well as rule construction.

This paper shows that an accurate decision procedure (90.5%) for determining the mass-count distinction in lexicalizations can be induced from the lexical mappings in the Cyc KB. The performance (74.6%) in the general case is also promising, given that it is a much harder task with over 30 part of speech categories to choose from. The features incorporate semantic information, in particular Cyc’s ontological types, in addition to syntactic information (e.g., headword morphology). Although the main approach incorporates Cyc’s conceptual distinctions, it can be extended to non-Cyc applications via the WordNet mapping [O’Hara *et al.*, 2003].

This work is just a small initial step in applying machine learning techniques to the massive amount of data in Cyc. The recent release of OpenCyc enables wider investigation and exploitation of the information in the Cyc knowledge base for intelligent applications.

Acknowledgements

The lexicon work at Cycorp has been supported by many staff members, and, in part by grants from NIST, DARPA, and ARDA. The first author is currently supported by a GAANN fellowship from the Department of Education. The work utilized NMSU resources obtained through MII Grants EIA-9810732 and EIA-0220590.

References

- [Brill, 1995] Eric Brill. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 21(4):543–565, 1995.
- [Burns and Davis, 1999] Kathy J. Burns and Anthony B. Davis. Building and maintaining a semantically adequate lexicon using Cyc. In Evelyn Viegas, editor, *The Breadth and Depth of Semantic Lexicons*, pages 121–143. Kluwer, 1999.
- [Lenat, 1995] D. B. Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11), 1995.
- [O’Hara *et al.*, 2003] Tom O’Hara, Nancy Salay, Michael Witbrock, Dave Schneider, Bjoern Aldag, Stefano Bertolo, Kathy Panton, Fritz Lehmann, Matt Smith, David Baxter, Jon Curtis, and Peter Wagner. Inducing criteria for mass noun lexical mappings using the Cyc KB, and its extension to WordNet. In *Proc. Fifth International Workshop on Computational Semantics (IWCS-5)*, 2003.
- [Quinlan, 1993] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, California, 1993.
- [Witten and Frank, 1999] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.
- [Woods, 2000] W. Woods. Aggressive morphology for robust lexical coverage. In *Proc. ANLP-00*, 2000.

POSTER PAPERS

NEURAL NETWORKS

Neural Executive Attentional Control in Robots

Jason Garforth¹, Sue McHale², Anthony Meehan¹

Department of Computing, The Open University, Milton Keynes, MK7 6AA, UK

Department of Psychology, Sheffield Hallam University, Sheffield, S1 1WB, UK

jpg@janus.demon.co.uk; s.l.mchale@shu.ac.uk; a.s.meehan@open.ac.uk

Abstract

We have developed a robot controller based upon a neural implementation of Norman and Shallice's model of executive attentional control in humans. A simulation illustrates how attentional control leads to the suppression of action selection errors in neurally controlled robots. A related demonstration illustrates how lesioning of the control architecture leads to behavioural pathologies that resemble those seen in human patients with damage to the prefrontal cortex.

1 Introduction

Selecting the right action at the right time is important for machines exhibiting higher-level behaviours. However, many researchers have found that robots exhibit behavioural pathologies in relation to action selection. Common examples include excessively frequent and inappropriate changes of behaviour (appearing either as distractedness or as indecision); inappropriate persistence of a behaviour; repetitive behaviour in which the robot appears to lack awareness of failure.

Humans suffering similar behavioural pathologies are often found to have suffered damage to an area of the pre-frontal cortex which is functionally labeled the *executive* [Parkin, 1996]. The *executive* initiates, monitors and modulates higher level behaviours. Several models of the executive exist, notably those of Baddeley and Weiskrantz [1993] and Norman & Shallice [Shallice, 1998]. Both these models contain an executive called the *Supervisory System or Supervisory Attentional System (SAS)*, respectively.

We have developed a neural implementation of the Norman & Shallice architecture as a controller for a simulated robot with the aim of demonstrating that such an architecture can reduce action selection errors.

2 Architecture

The control network (ca. 700 neurons) features clusters of highly interconnected neurons within functional

blocks corresponding to those of the original Norman and Shallice model (see below). These clusters are sparsely connected to other clusters within the same functional block or to clusters in neighbouring blocks. The inter-block connections are based on known neuroanatomical structures and pathways.

The functional architecture features a *Perception Layer* which fuses sensor signals before distributing these to an *Associative Layer* which maps perceptions to a behaviour layer. The *Behaviour Layer* holds a number of distinct networks which individually exhibit basis behaviours such as 'wander safely', 'aggregate', 'disperse' [Mataric, 1996]. Basis behaviours can be combined to provide higher-level behaviours. A *Contention Scheduler* takes input from stimulated behaviours in the behaviour network requesting access to the robots effector systems and selects which behaviors are granted expression in the real world. (Our contention scheduler is an independent implementation of that given by Prescott et al.[1999].)

Two *SAS* functions are currently implemented. The *monitor network* compares the currently intended behaviour with the currently expressed behaviour, generating an arousal stimulus if a disparity occurs. Arousal causes the *modulator network* to modulate the behavioral signals into the contention scheduler from the behaviour layer such that the intended behaviour is potentiated and the other behaviours are attenuated. It is important to recognise that this does not guarantee the selection of the intended behaviour as this risks overriding behaviours designed to prevent undesirable outcomes such as collision.

3 Simulation

This section illustrates the operation of the simulated robot which has both sonar and olfactory sensors and is equipped with independent drive wheels and a gripper for picking up objects of interest. The dynamics of the robot motion and the sensor are based techniques prescribed by Dudek and Jenkin [2000]. We use foraging behaviour (wander until detect food, collect food, take food to home), to illustrate the normal functioning of the SAS.

In Figure 1. the robot (Penny) has detected food in a region near the bottom of the world, oriented itself and then moved towards that food, collected it, and is now taking it towards 'home'.

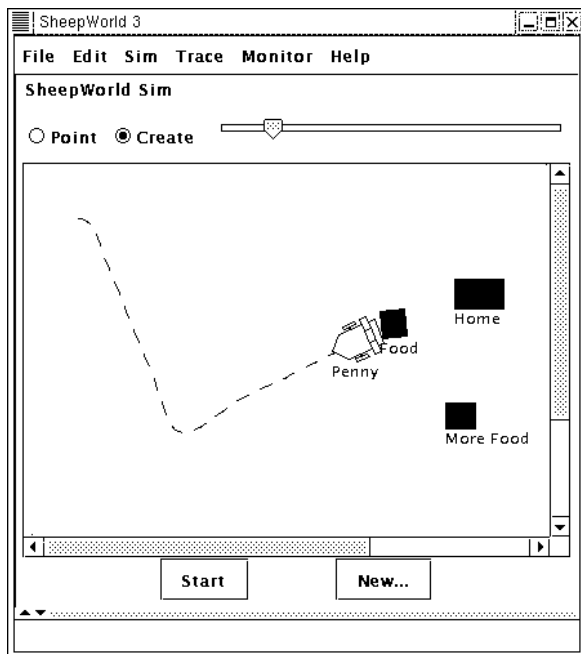


Figure 1. Simulated robot exhibiting foraging behaviour.

The successful operation of the SAS is illustrated in Figure 2. which provides a sequence of traces which are the outputs of clusters and/or functional units of the neural architecture. Whilst the robot is currently taking food home (as in Figure 1) an additional, distracting, food source is introduced into the environment. This occurs some 2 seconds into the trace. The top trace shows that the robot detects the new food and the 'orient to food' behaviour requests expression via the contention scheduler. Trace 3 shows that this is not the currently planned behaviour (planning input for this behaviour is 'low'). However, the strength of the conditioning to the 'orient to food' stimulus leads the Contention Scheduler to select (inappropriately) the 'orient to food' behaviour (rising spike in trace 4). But, the SAS monitor sees this (trace 5), and correctly generates a modulatory signal to suppress the level of excitation of this behaviour as seen by the Contention Scheduler (trace 6). This results in the falling spike of trace 4. Trace 7 illustrates that there was a *momentary* expression of the inappropriate behaviour at one of the motors.

4 Conclusion

We have demonstrated that a neural controller based upon the Norman and Shallice model of executive attentional control can express willed behaviour required suppress otherwise inappropriate behavior. In a related demonstration of this simulation, lesion studies are used to

reproduce robot behaviour that appears to correspond to similar pathologies exhibited by human patients .

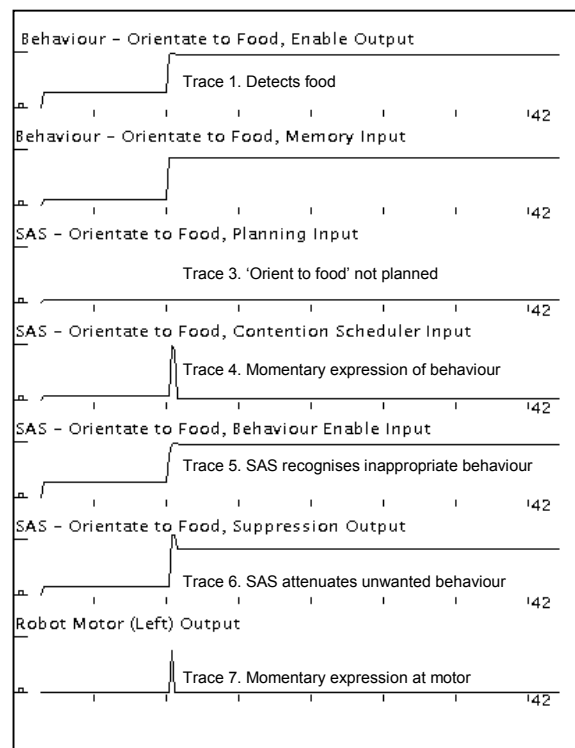


Figure 2. Suppression of inappropriate behaviour by the SAS.

References

- [Baddeley and Weiskrantz, 1993] A.Baddeley and L. Weiskrantz. *Attention: Selection, Awareness, & Control*. Oxford University Press, 1993.
- [Dudek and Jenkin, 2000] Gregory Dudek and Michael Jenkin. *Computational Principles of Mobile Robotics*. Cambridge University Press, 2000.
- [Mataric, 1996] Maya Mataric. Designing and Understanding Adaptive Group Behaviour. *Adaptive Behaviour* 4(1): 51-80, 1996.
- [Prescott et al, 1999] T.J. Prescott, P. Redgrave and K. Gurney. Layered control architectures in robots and vertebrates. *Adaptive Behaviour*, 7(1):99-127. 1999.
- [Robbins, 1991] T.W. Robbins. Cognitive Defecits in Schizophrenia and Parkinson's Disease: Neural basis and the Role of Dopamine. In P. Willner and J. Scheel-Kruger (eds) *The Mesolimbic Dopamine System: From Motivation to Action*. pages 497-526, John Wiley, 1991
- [Shallice, 1988] Tim Shallice. *From Neuropsychology to Mental Structure*. Cambridge University Press, 1988.

Boosting Face Identification in Airports

Liu Jiang Jimmy, Kia-Fock Loe

National University of Singapore, School of Computing,
Science Drive 2, Singapore 117559, Republic of Singapore
liujiang@pacific.net.sg, loekf@comp.nus.edu.sg

Abstract

Robust face identification system in complex airport environment, which can identify certain candidates from a crowd of people in real time, is in urgent demand. S-AdaBoost is discussed in this paper as a variant of AdaBoost to handle real world environment. The Face Identification System for Airports (FISA), based upon S-AdaBoost algorithm, is implemented in an international airport. Comparison of results obtained from FISA with those from other leading face identification approaches based on FISA database clearly demonstrates the effectiveness of FISA in real airport environment.

1 Introduction

When AdaBoost [Freund and Schapire 1996] is used to handle scenarios in complex environment with outliers, its limitations have been pointed out by researchers [Dietterich, 2000]. Some discussions and approaches [Rätsch, et al., 2001] have been proposed to address these limitations.

S-AdaBoost works by dividing the input space into a few sub-spaces and using dedicated classifiers to classify patterns in the sub-spaces. The final classification result is the combination of the outputs of the dedicated classifiers. The S-AdaBoost Machine is made up of an AdaBoost divider, an AdaBoost classifier, a dedicated classifier for outliers, and a non-linear combiner. S-AdaBoost can enhance AdaBoost's capability of handling scenarios in real world complex environment.

2 S-AdaBoost in Classification

S-AdaBoost applies the *Divide and Conquer Principle* through *dividing* the input pattern space $\hat{S}$ into a few sub-spaces and *conquering* the sub-spaces by finding simple fittings (decision boundaries) to the patterns in the sub-spaces. Input space can be denoted by:

$$\hat{S} = \{P = (X, Y)\}$$

where,

$X = \{x_i\}$ denotes the input patterns.

$Y = \{y_i\}$ denotes the classification results.

$P = \{p_i = \{(x_i, y_i)\}\}$ denotes the input pattern and classification result pairs.

In S-AdaBoost, patterns in $\hat{S}$ can be divided into a few sub-spaces relative to a classifier $F(x)$:

$$\hat{S} = \hat{S}_{no} + \hat{S}_{sp} + \hat{S}_{ns} + \hat{S}_{hd} \quad (1)$$

where,

$\hat{S}_{no} = \{P_{no}\}$: Normal Patterns (Patterns can be easily classified by $F(x)$).

$\hat{S}_{sp} = \{P_{sp}\}$: Special Patterns (Patterns can be classified correctly by $F(x)$ with bearable adjustment).

$\hat{S}_{ns} = \{P_{ns}\}$: Patterns with Noise (Noisy patterns)

$\hat{S}_{hd} = \{P_{hd}\}$: Hard-To-Classify Patterns (Patterns hard to be classified by $F(x)$).

A typical input pattern space is shown in Figure 1. The first two sub-spaces are further referred to as Ordinary Pattern Space, and the last two are called Outliers in S-AdaBoost:

$$\hat{S}_{od} = \hat{S}_{no} + \hat{S}_{sp} \quad (2)$$

$$\hat{S}_{ol} = \hat{S}_{ns} + \hat{S}_{hd} \quad (3)$$

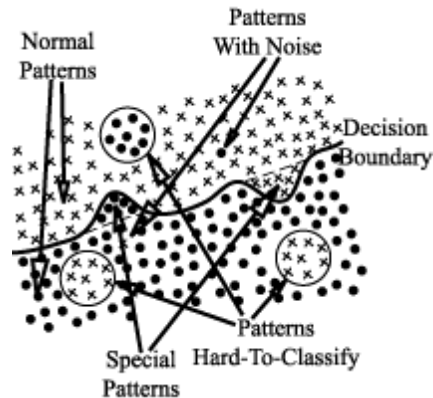


Figure 1: Input Pattern Space

During training, S-AdaBoost uses an AdaBoost $D(\mathbf{t})$ as a divider to divide the patterns in the original training input space $\hat{S}$ into two separate sets in $\hat{S}_{od}$ and $\hat{S}_{ol}$. One set in $\hat{S}_{od}$ is used to train the next AdaBoost $F_{od}(\mathbf{x})$, which has good generalization characteristic; and another set in $\hat{S}_{ol}$ is used to train a dedicated outlier classifier $IO(\mathbf{x})$, which has good localization characteristic. The structure of S-AdaBoost is shown in Figure 2. During testing, the divider $D(\mathbf{t})$ is no longer needed, testing patterns are fed directly to the two classifiers $F_{od}(\mathbf{x})$ and $IO(\mathbf{x})$ followed by the combiner C to obtain the classification results.

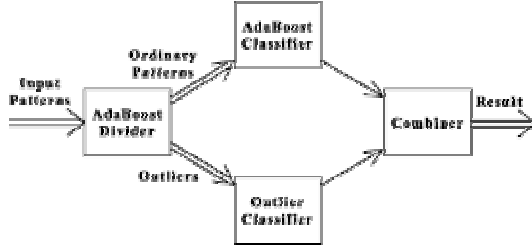


Figure 2: S-AdaBoost Machine

3 FISA

In FISA, 40 candidate's face images are stored in the candidate database. 5000 images with one or multiple face images in an airport environment are collected within 3 months; about 40% of the images contain one or a few candidate faces. 3000 images are randomly selected as the training set and the remaining 2000 images are chosen as the test set.

The pro-processor will detect potential faces and generate constant flow of 20 X 20 segmented potential face images to FISA. Back-Propagation (BP) Neural Network is used as the base learner for AdaBoost $D(\mathbf{t})$ and $F_{od}(\mathbf{x})$ in FISA. A three-layer RBF Neural Network with dynamic number of hidden nodes is chosen to implement $IO(\mathbf{x})$ due to RBF Neural Network's good localization characteristic. The radii of hidden nodes in RBF are also chosen to be very small to enhance RBF network's good local clustering characteristic, which helps to isolate the P_{ns} from the P_{hd} . A Multi-Layer Perceptron (MLP) Neural Network is used to implement the combiner C in FISA.

To test the effectiveness of S-AdaBoost (**S-AB**) on face identification in airports, we compared the performance of FISA (when Threshold $\mathbf{t}$ was set to $1/(M \times \mathbf{d}^2)$) with other leading approaches. We implemented the neural network based **EBGM** (Elastic Bunch Graph Matching) approach [Wyner et al., 2001], the statistical subspace **LDA** (Linear/Fisher Discriminant Analysis) approach [Friedman et al., 1998], and Probabilistic PCA (Principle Component Analysis) approach [Freund, 1999], **PPCA**. The False Negative Rate (FNR) and False Positive Rate (FPR) of the

four algorithms as well as those of the AdaBoost (**AB**) algorithm were used in our analysis.

The same training and testing face images (as used in FISA) were used in our experiment to compare the effectiveness of different approaches in real complex airport environment. In FISA testing, the pre-processed data (20 X 20 images) were fed directly to $F_{od}(\mathbf{x})$ and $IO(\mathbf{x})$. The testing results obtained from various approaches are listed in Table 1.

%	S-AB	AB	EBGM	LDA	PPCA
FPR	33.4	44.6	40.2	43.3	38.5
FNR	0.1	1.4	0.2	0.6	0.5

Table 1: Comparison of error rates of different approaches on FISA Database

4 Discussion and Conclusions

FISA is introduced as a practical system to handle face identification in real airport environment. S-AdaBoost, which is a variant of AdaBoost and is more effective than the conventional AdaBoost in handling outliers in real world complex environment, is also introduced as the algorithm behind FISA. Experimental results on FISA databases clearly show S-AdaBoost's effectiveness in handling classification in complex environment and FISA's capability in boosting Face Identification in Airport. Improvements of the algorithm will focus on utilizing more hybrid approaches to improve the overall identification rates of the system.

References

- [Dietterich, 2000] Dietterich T. G. (2000) An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40 (2) pp. 139-158
- [Freund and Schapire 1996] Freund, Y., R. E. Schapire. (1996) Experiment with a new boosting algorithm, *Machine Learning: proceeding of the 13th International conference*, pp. 148-156
- [Freund, 1999] Freund, Y. (1999) An Adaptive Version of the Boost by Majority Algorithm. *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*
- [Friedman et al., 1998] Jerome Friedman, Trevor Hastie, and Robert Tibshirani (1998). Additive logistic regression: a statistical view of boosting. *Stanford University Technical Report*.
- [Rätsch, et al., 2001] G. Rätsch, T. Onoda, and K.-R. Müller (2001). Soft margins for AdaBoost. *Machine Learning Journal*, 42(3):287-320, March. Kluwer Academic Publishers
- [Wyner et al., 2001] Abraham Wyner, Abba Kriege and Chuan Long Boosting Noisy Data. *Proceeding of 8th ICML*

Action Selection for Single- and Multi-Robot Tasks Using Cooperative Extended Kohonen Maps

Kian Hsiang Low[†], Wee Kheng Leow[†], and Marcelo H. Ang, Jr.[‡]

[†]Dept. of Computer Science, National University of Singapore

3 Science Drive 2, Singapore 117543, Singapore

[‡]Dept. of Mechanical Engineering, National University of Singapore

10 Kent Ridge Crescent, Singapore 119260, Singapore

[†]ieslkh, leowwk@comp.nus.edu.sg, [‡]mpeangh@nus.edu.sg

Abstract

This paper presents an action selection framework based on an assemblage of self-organizing neural networks called *Cooperative Extended Kohonen Maps*. This framework encapsulates two features that significantly enhance a robot's action selection capability: self-organization in the continuous state and action spaces to provide smooth, efficient and fine motion control; action selection via the cooperation and competition of Extended Kohonen Maps to achieve more complex motion tasks. Qualitative and quantitative comparisons for single- and multi-robot tasks show our framework can provide better action selection than do potential fields method.

1 Introduction

A central issue in the design of behavior-based control architectures for autonomous agents is the formulation of effective action selection mechanisms (ASMs) to coordinate the behaviors. This paper describes a neural network-based ASM for autonomous non-holonomic mobile robots. Our motivation is to develop a motion control strategy that can perform distributed multi-robot surveillance in unknown, dynamic, and unpredictable environments. By implementing the ASM using an assemblage of self-organizing neural networks, it induces the following key features that significantly enhance the agent's action selection capability: *self-organization of continuous state and action spaces* to provide smooth, efficient and fine motion control, and *action selection via the cooperation and competition of Extended Kohonen Maps* to achieve more complex motion tasks.

2 Action Selection Framework

Our proposed ASM, termed *Cooperative Extended Kohonen Maps* (EKMs), is implemented by connecting an ensemble of EKMs. An EKM extends the Kohonen Self-Organizing Map. Besides encoding a set of input weights that self-organize in the sensory input space, the neurons also produce outputs that vary with the incoming sensed inputs. Our implementation extends the work of [Low *et al.*, 2002] by connecting several EKMs to form cooperative EKMs. These neural networks cooperate and compete to produce an appropriate motor action

for the robot to approach targets, negotiate unforeseen, possibly concave, obstacles, and keep away from robot kins when it is tracking moving targets (Fig. 1).

Our ASM framework consists of four types of EKMs: target localization, obstacle localization, robot kin localization, and motor control EKMs. In the presence of a target, neurons in the target localization EKM, which encodes target location in the local sensory input space $\mathcal{U}'$, are activated (Fig. 1a). A *target field* with the shape of an elongated Gaussian is produced (Fig. 1b) such that the neurons at and near the target location have the strongest activities. The elongated target field is crucial to the robot's avoidance of concave obstacles.

Similarly, the presence of an obstacle activates neurons in the obstacle localization EKMs. The neurons in these EKMs at and near the obstacle locations will be activated to produce *obstacle fields* (Fig. 1c). These obstacle fields are stretched along the obstacle directions such that neurons beyond the obstacle locations are also inhibited to indicate inaccessibility. *Robot kin fields* are activated in a similar way in the robot EKMs in the presense of robot kins.

In activating the motor control EKM, the obstacle fields are subtracted from the target field (Fig. 1d). If the target lies within the obstacle fields, the activation of the motor control EKM neurons close to the target location will be suppressed. Consequently, another neuron at a location that is not inhibited by the obstacle fields becomes most highly activated (Fig. 1d). This neuron produces a control parameter that moves the robot away from the obstacle. While the robot moves around the obstacle, the target and obstacle localization EKMs are continuously updated with the current locations and directions of the target and obstacles. Their interactions with the motor control EKM produce fine and smooth motion control of the robot to negotiate the obstacle and reach the target. In the case of multi-robot tracking of multiple targets, multiple target fields and robot kins fields are activated. The robots act like highly repulsive obstacles to other robots, thus separating them from each other.

3 Experiments and Discussions

Two qualitative tests were conducted to demonstrate the capabilities of cooperative EKMs in performing complex motion tasks. The experiments were performed using Webots, an embodied simulator for Khepera mobile robots, which incorporated 10% noise in its sensors and actuators.

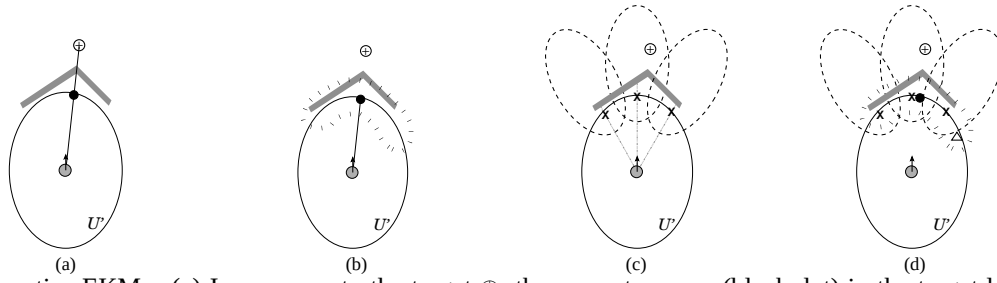


Figure 1: Cooperative EKMs. (a) In response to the target $\oplus$, the nearest neuron (black dot) in the target localization EKM (ellipse) of the robot (gray circle) is activated. (b) The activated neuron produces a target field (dotted ellipse) in the motor control EKM. (c) Three of the robot's sensors detect obstacles and activate three neurons (crosses) in the obstacle localization EKMs, which produce the obstacle fields (dashed ellipses). (d) Subtraction of the obstacle fields from the target field results in the neuron at Δ to become the winner in the motor control EKM, which moves the robot away from the obstacle.

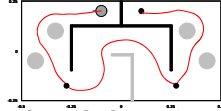


Figure 2: Motion of robot (dark gray) in an environment with unforeseen static obstacles (light gray). The robot successfully navigated through the checkpoints (small black dots) located at the doorways to reach the goal.

The environment for the first test consisted of three rooms connected by two doorways with unforeseen static obstacles (Fig. 2). The robot began in the top corner of the left-most room and was tasked to move into the narrow corner of the right-most room via checkpoints plotted by a planner. The robot with cooperative EKMs was able to move through the checkpoints to the goal by traversing between narrowly spaced convex obstacles in the first and the last room, and overcoming an concave obstacle in the middle room. A robot with potential fields would be trapped by these local minimas.

The second test (Fig. 3) illustrates how two robots endowed with cooperative EKMs cooperate to track four moving targets. When the targets were moving out of the robots' sensory range, the robot below chose to track the two targets moving to the bottom left while the robot above responded by tracking the two targets moving to the top right. In this manner, all targets could be observed by the robots. This test shows that the two robots can cooperate to track multiple moving targets without communicating with each other.

Two quantitative tests were conducted to determine the overall tracking performance of the robot team based on the following performance index:

$$\text{observation coverage} = \sum_{t=1}^T 100 \frac{n(t)}{NT} \quad (1)$$

where N is total number of targets, n is number of targets being tracked at time t , and the experiment lasts T amount of time. For both tests, N and T were fixed respectively as 10 targets and 1000 time steps at intervals of 128 ms.

The first test compared the mean observation coverage of robots adopting four different tracking strategies: cooperative EKMs, potential fields, fixed deployment, and random deployment. The environment or arena was an enclosed obstacle-free region that varied in size. The mobile targets were forward-moving, obstacle-avoidance vehicles that changed their direction and speed with 5% probability. Five

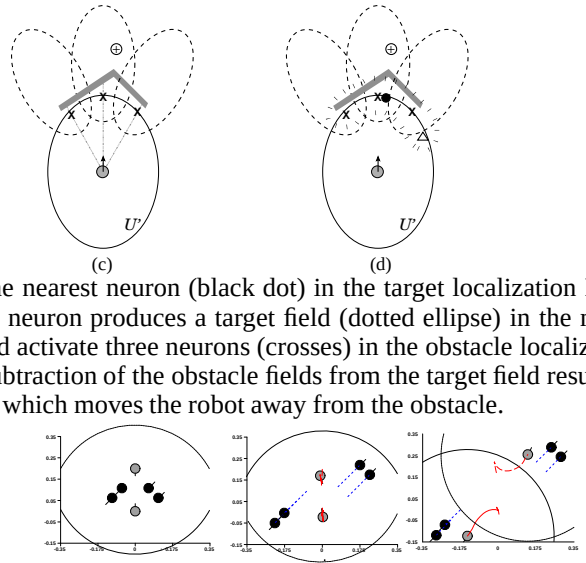


Figure 3: Cooperative tracking of moving targets. When the targets were moving out of the robots' sensory range, the robot below decided to track the targets moving to the bottom left while the robot above responded by tracking the targets moving to the top right. In this way, all targets could still be observed by the robots.

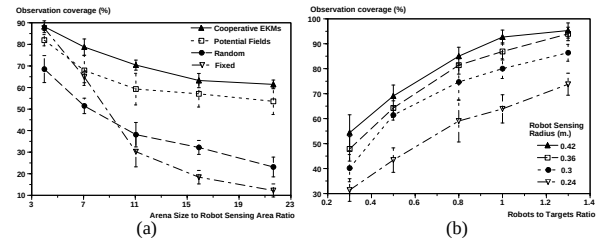


Figure 4: Comparison of observation coverage for (a) robots using different tracking strategies in varying arena size, and (b) varying number of robots with different sensing ranges.

robots, each with target and robot sensing radius of 0.3 m, were deployed in this task. The fixed deployment approach distributed stationary robots uniformly over the arena. The random deployment approach allowed the robots to move randomly in a manner similar to the moving targets. Test results in Fig. 4(a) reveal that, in very large arenas, tracking strategies that respond dynamically to targets' motion (cooperative EKMs and potential fields) are significantly better than those that do not (fixed and random). In particular, cooperative EKMs offered the highest observation coverage as it could overcome local minimas posed by targets and robots.

The second test compared the mean observation coverage of the cooperative-EKM robots with different sensing ranges and number of robots. The size of the arena was 6.4 m^2 , which corresponded to the largest arena used for the first test. Test results in Fig. 4(b) show that observation coverage increases with increasing number of robots and sensing range.

References

[Low *et al.*, 2002] K. H. Low, W. K. Leow, and M. H. Ang Jr. A hybrid mobile robot architecture with integrated planning and control. In *Proc. AAMAS*, pages 219–226, 2002.

Artificial Neural Network for Sequence Learning

Sorin Moga

IASC / ENST Bretagne / GET
BP 832, Brest, 29285, France
sorin.moga@enst-bretagne.fr

Philippe Gaussier

ETIS / Neurocyber / U. of Cergy
6, av. Ponceau, Cergy, 95014, France
gaussier@ensea.fr

Abstract

This poster shows an artificial neural network capable of learning a temporal sequence. Directly inspired from a hippocampus model [Banquet *et al.*, 1998], this architecture allows an autonomous robot to learn how to imitate a sequence of movements with the correct timing.

1 Introduction

This article considers the problem of learning to predict events, i.e. to forecast the future behavior of a system using past experience. This problem has often been viewed and formalized in neural network theory as so-called temporal sequence learning. Studying such sequences is a topic of research in several domains such as robotic trajectory planning, speech or vision processing. For most of these, neural networks provide two distinct mechanisms: one for spatial information and the other for temporal information. The main mechanism stores the sequence events regardless of the temporal dependences between them. In parallel, or later, the second mechanism, the so-called short time memory (STM), extracts and learns the temporal relationships between the events. Moreover, we have shown in previous works [Gaussier and Moga, 1998; Andry *et al.*, 2001] that the capability of learning a temporal sequence is one of most important features of a learning by imitation system. As a capability of learning by observation, imitation is a strong learning paradigm for autonomous systems. Imitation can improve and accelerate the learning of sensory/motor associations. In our work, oriented to the design of a neural network architecture allowing learning by imitation, we are involved in the first level of imitation [Whiten and Ham, 1992]. This “proto-imitation” level plays a key role in understanding the principles of the perception/action mechanisms necessary to perform higher order behaviors and it is likely that the proto-imitation is triggered by a perception ambiguity. In our approach, the starting point for an “imitating behavior” implementation is the capability of learning temporal sequences of movements.

2 Temporal sequence learning model

Almost all neural models of sequence learning use a discrete temporal dimension by sampling the continuous time at reg-

ular intervals. In these models, time proceeds by intervals of Δt , and the interval between 2 items of a sequence is considered as a few units of Δt (usually less than 10). In this section, we introduce the model¹ for timing sequence learning with a variable and a long range time interval and we evaluate it. Our model (Fig. 1-left) is based on the idea that a prediction (P-type) neuron learns the timing between 2 items, or, to be more precise, learns to predict the end of this time interval. This time interval starts with the firing of a derivation (D-type) neuron and ends with the firing of an input (E-type) neuron. The P-type neuron learns this interval using the activity of the granular (G-type) group of neurons.

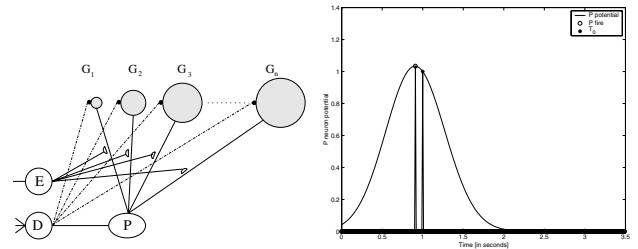


Figure 1: Left : The overview of the neural model allowing time prediction. The G_i are time base neurons, P is the prediction neuron, D and E are formal neurons. Right : detailed activity of the P-type neuron. The P neuron firing predict the learned time interval.

Let us consider a simple example: we present the first item and, one second later, we present the second item. The length of the interval to be learned is $T_0 = 1s$. The first item forces the D neuron to fire. The firing of the D neuron resets all the activity of the G neurons. Starting at this instant, the G neuron’s activity is expressed by Eq. 1. One second later, the second item forces the E neuron to fire. The firing of the E neuron enables the update of the weight between the P and the G neurons i.e. enables the learning of the T_0 interval. Finally, when the first item is presented again the D neuron fires again. 920 milliseconds later, i.e. 80 milliseconds before

¹The model is inspired by the functions of two brain structures involved in memory and time learning: the cerebellum and the hippocampus (see [Banquet *et al.*, 1998] for further neurobiological references)

T_0 , the P neuron fires (Fig. 1-right) and predicts an imminent firing of the E neuron.

$$G_i(t) = \frac{m_0}{m_i} \exp - \frac{((t - \tau_i) - m_i)^2}{2 \cdot \sigma_i^2} \quad (1)$$

i - position of the neuron in the group (the i th cell of the battery); m_i and σ_i - time constant and the standard deviation associated with the i th neuron; τ_i - instant of the last reset of the battery.

3 Sequence learning architecture

Timing learning models are currently used by neurobiologist modelers for conditioning simulation [Bullock *et al.*, 1994; Grossberg and Merrill, 1992]. Alternatively, the proposed model permits the temporal sequence of events to be learned and predicted. In our context, a simple sequence is defined as an enumeration of events with the associated timing (eg. "A,B,C") and a cyclic sequence as periodic simple sequence (eg. "A,B,C,A,B,C,A,...") with the associated timing. The main idea is to use several batteries of G neurons for learning the timing between two consecutive events and a group of P type neurons for learning the event sequencing. The global architecture is shown in Fig. 2. The input group (CC) can be viewed as the input interface. Any neuron of this group represents a sequence event and it is ON while the corresponding input event is present; otherwise it is OFF. Each CC neuron is one-to-one linked with EC group of neurons. The EC group is made up of D-type neurons. Each EC neuron is linked with unconditional links to all neurons of a battery in the DG group. In the same way, an EC neuron is connected with unconditional links to all neurons of the corresponding column of the CA3 group. The DG group integrates several batteries

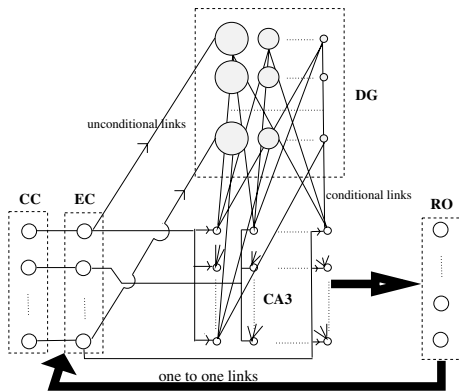


Figure 2: Detailed connectivity of the event prediction network. The circle size in DG is associated with the time constants (m_i) of the G type neurons.

of G -type neurons. A DG battery is equivalent to a G_i group of neurons shown in section 2. Each DG neuron is connected via conditional links to all neurons of the corresponding row in the CA3 group. The size of the CC group (respectively EC) is constrained by the maximum length of sequences. Alternatively, the size of a DG battery is a function of the prediction precision of the time interval between two events of the sequence. This architecture can learn all event combinations, in

other words, all possible sequences. Consequently, the size of the CA3 group is the square of the input group (CC). The output group (RO) is a Winner Take All neurons group. A neuron of the RO group has the same signification as a CC neuron. The RO outputs are connected to the EC inputs via one-to-one secondary unconditional links.

The architecture allows the learning of all kind of simple or cyclic sequences. The size of learned sequences is limited only by the system memory capacity.

4 Conclusion

The timing learning model and the associated neural networks were successfully utilized in [Gaussier and Moga, 1998; Moga, 2000] to teach an autonomous robot different "dances". The proposed model allows the correct timing to be predicted and it concords with Weber's law. Even if Weber's law concordance is not a prerequisite, it corresponds to a strong constraint of neurobiological inspired models of perception and learning: if the prediction precision is constant then it is not possible to have reinforcement due to repetitive experiments. In addition, this model was successfully employed [Andry *et al.*, 2001] to build a learning model based on the prediction of rhythms as a reward signal and for spatio-temporal transition learning in autonomous robot navigation. These results prove that the proposed neural network architecture can serve both as a starting point for understanding imitation mechanisms, and as an effective learning algorithm for autonomous robotics.

References

- [Andry *et al.*, 2001] P. Andry, P. Gaussier, S. Moga, and J. Nadel. Learning and communication via imitation: an autonomous robot perspective. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 31(5):431–442, 2001.
- [Banquet *et al.*, 1998] J.P. Banquet, P. Gaussier, J.L. Contreras-Vidal, and Y. Burnod. The cortical-hippocampal system as a multirange temporal processor: A neural model. In R. Park and D. Levin, editors, *Fundamentals of neural network modeling for neuropsychologists*, Boston, 1998. MIT Press.
- [Bullock *et al.*, 1994] D. Bullock, J.C. Fiala, and S. Grossberg. A neural model of timed response learning in the cerebellum. *Neural Networks*, 7(6/7):1101–1114, 1994.
- [Gaussier and Moga, 1998] P. Gaussier and S. Moga. From perception-action loops to imitation processes: A bottom-up approach of learning by imitation. *Applied Artificial Intelligence: An international Journal*, 12(7–8):701–727, 1998.
- [Grossberg and Merrill, 1992] S. Grossberg and J. W. L. Merrill. A neural network model of adaptively timed reinforcement learning and hippocampal dynamics. *Cognitive Brain Research*, 1:3–38, 1992.
- [Moga, 2000] S. Moga. *Apprendre par imitation : une nouvelle voie d'apprentissage pour les robots autonomes*. PhD thesis, Université de Cergy-Pontoise, 2000.
- [Whiten and Ham, 1992] A. Whiten and R. Ham. On the nature and evolution of imitation in the animal kingdom: Reappraisal of a century of research. In P.J.B. Slater, J.S. Rosenblatt, C. Beer, and M. Milinski, editors, *Advances in the study of behavior*, pages 239–283, San Diego, CA, 1992. Academic Press.

POSTER PAPERS

PLANNING

Comparison of Different Grid Abstractions for Pathfinding on Maps

Yngvi Björnsson, Markus Enzenberger, Robert Holte, Jonathan Schaeffer and Peter Yap

Department of Computing Science, University of Alberta,
Edmonton AB, Canada T6G 2E8
{yngvi, emarkus, holte, jonathan, petery}@cs.ualberta.ca

Abstract

Pathfinding on a map is a fundamental problem in many applications, including robotics and computer games. Typically a grid is superimposed over the map where each cell in the grid forms a unique state. A state-space-based search algorithm, such as A* or IDA*, is then used for finding the optimal (shortest) path. In this paper we analyze the search behavior of both A* and IDA* using different grid representations, providing various new insights via analytical and empirical results.

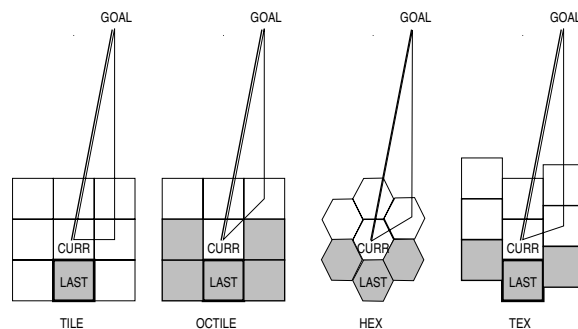


Figure 1: Different grid representations

1 Introduction

Commercial games are a multi-billion-dollar industry and still growing. In the past better graphics have been one of the main driving forces of sales; however, this is no longer true and good graphics alone is not sufficient to fuel sales. Instead consumers are increasingly looking for a more realistic gaming experience. For many type of games, especially real-time-strategy (RTS) and role-playing games (RPG), realistic real-time pathfinding is one of the fundamental hurdles to overcome. In many state-of-the-art games a grid is typically superimposed over the map and an algorithm such as A* or IDA* is used to find a path in the abstract grid space.

In this paper we compare four different grid topologies for abstraction: *tiles*, *octiles*, *hexes*, and *texes*. Tiles are squares where movement is restricted to the four cardinal compass directions (N, E, S, W). Octiles are also squares but allow additional diagonal moves (NE, SE, SW, NW). A hex grid consists of hexagons with six degrees of movement. One problem with a hex grid is that it cannot be implemented as effectively as a square-based grid (an important consideration in computer games). Thus we introduce *texes*, a square-based representation of a hexagonal grid. The four different grid structures are depicted in Figure 1.

2 Analytical Comparison

In this section we analyse the search complexity of IDA* for each of the different grid topologies discussed above. In our analysis we assume unit-cost edge weights (as is customary).

The asymptotic search complexity of A* on a grid is $O(D^2)$ independent of grid representation, where D is the

depth of the solution. On the other hand, the search complexity of IDA* is $O(b^D)$ where b is the effective branching factor (in this analyses we exclude the effect of the heuristic estimate). The choice of a grid representation affects not only the branching factor b , but also the solution depth D . This poses the question: which grid representation is the most efficient?

As for the branching factor, there is no need to explore again the cell that the search just departed from, thus the effective branching factor of IDA* is at least one less than the number of adjacent cells. However, for hexes, texes and octiles we can further reduce the branching factor by observing that *none of the neighbors shared with the previously visited cell need to be expanded*. For example, in Figure 1 the shaded cells need not be expanded at node *curr* given that it was last entered via node *last*. Intuitively, because of unit-cost terrain, the cost of reaching these neighboring cells from *curr* is necessarily greater than if they were instead visited directly from *last*. The effective branching factor of hexes and texes is thus only 3, same as for tiles, whereas for octiles the asymptotic branching factor is 4.2 (3 for horizontal/vertical moves and 5 for diagonal moves)[1]. Surprisingly, this observation has been overlooked in previous studies.

As for the search depth D , it can be mathematically shown that given the same distance from a *start* to a *goal*, if a tile grid searches to a depth D then a hex grid searches only to a depth $0.81D$ and an octile grid to a depth $D/\sqrt{2}$. The analytical result is summarized in Table 1.

Table 1: Summary of Grids

Grid Type	Adj. Nodes	Branch. Factor	Average Depth	Complexity	
				A*	IDA*
Hex	6	3	$0.81D$	$O(D^2)$	$O(2.42^D)$
Tex	6	3	$0.81D$	$O(D^2)$	$O(2.43^D)$
Octile	8	4.2	$0.71D$	$O(D^2)$	$O(2.77^D)$
Tile	4	3	$1.00D$	$O(D^2)$	$O(3.00^D)$

3 Testing Framework

The above analytical study gives us a good idea of the merits of the different grid representations, at least when used with IDA*. However, for A* it does not provide much insight. An empirical study of these algorithms is thus important, especially given that no comparison study exists on their behavior when pathfinding on a map. A problem with previous work is that different studies use disparate testing environments making individual results hard to compare. Also, the “random” maps typically used are not necessarily representative of maps one would encounter in practice.

To address both of the above mentioned problems we developed a generic framework for testing state-space based search algorithms. This framework consists of three layers, allowing implementation of each layer independently of the others. The bottom layer abstracts away the details of the grid topology, the middle layer consists of a state-space-based search algorithm, whereas the top layer provides facilities for automatic testing, collection of statistics, and tools for importing game maps. The plan is to make this software public to encourage more realistic and uniform test-beds for pathfinding research.

In this framework, we implemented both A* and IDA* in the middle-layer, and tile-, octile-, and tex-representations of maps in the bottom layer (there is no need to include hex grids because tex grids have the same search properties while allowing a more efficient implementation).

4 Empirical Comparison

We used the aforementioned test environment to empirically evaluate A* and IDA* using different grid structures. The following map categories were used: obstacle-free maps (*E*), maps with randomly placed obstacles (*R*), and maps extracted from the popular computer game *Baldur’s Gate II* by *BioWare* (*B*). For each category 120 maps were used, ranging in size from approximately 50×50 to 300×300 . For each map 400 randomly chosen *start/goal* states were searched.

The result of the A* experiment is summarized in Table 2. Each entry shows the average number of nodes expanded over 48,000 searches (400 trials $\times$ 120 maps) as well as the standard error. The heuristic used for each grid representation is perfect given that no obstacles are present (e.g. Manhattan distance for tiles). We can see that the exact graph representation does not make a big difference for A*. However, any improvement is beneficial because multiple units are typically traversing a game world simultaneously, each requiring a path be found in a matter of milliseconds. For example, we can state with a high statistical significance for all three map

Table 2: Summary of A* Experiments

	Tile		Tex		Octile	
E	72.4	(± 0.3)	58.5	(± 0.2)	50.8	(± 0.2)
R	493.7	(± 5.0)	516.9	(± 5.0)	466.7	(± 4.9)
B	606.0	(± 8.2)	588.6	(± 8.0)	511.4	(± 7.5)

categories that octile grids expand fewer nodes on average than the alternative grid structures.

Identical experiments were conducted for IDA*. Because of its exponential growth rate, normal IDA* performed poorly on the larger maps, often unable to find a path even after expanding millions of nodes. Instead we used a memory-enhanced variant of IDA* for our evaluation (a simplified version of the transposition table enhancement described in [2]). The result is summarized in Table 3. When comparing this result with the A* result we notice that IDA* expands many more nodes on average (because it needs to revisit nodes on each iteration). However, despite expanding more nodes, the overall run-time for IDA* was considerably less than A*’s for both empty and random maps (less work per-node). Also of interest is that tiles clearly work better than the other grid topologies when used in conjunction with enhanced IDA*. The data also shows a big performance difference between random and game maps. On game maps IDA* searches an order of magnitude more nodes than on random maps, resulting in A* being the preferred algorithm for these game maps (even when considering run-time).

Table 3: Summary of IDA* Experiments

	Tile		Tex		Octile	
E	72.4	(± 0.3)	58.5	(± 0.2)	50.8	(± 0.2)
R	1514.9	(± 25.2)	2346.7	(± 38.6)	1904.7	(± 38.4)
B	19597.1	(± 501.3)	28899.4	(± 757.8)	21188.7	(± 568.9)

5 Conclusions

Analytical results show that normal IDA* performs best on tex (or hex) grid abstraction. Unfortunately, normal IDA* is impractical for larger maps, including realistic game maps. In practice, A* performs best on octile grids and memory-enhanced IDA* on tile grids. This result is based on maps typical of computer games in terms of size and level of detail. For IDA* and A* we use the fact that only a subset of neighboring nodes need to be visited, an often overlooked fact. The result also shows that one must be careful when carrying insights gained from random maps over to “realistic” maps.

Acknowledgments

This research was supported by grants from NSERC, iCORE, and IRIS.

References

- [1] P. Yap. Grid-based Pathfinding. *Lecture Notes in Artificial Intelligence*, vol 2338, 44–55, 2002.
- [2] A. Reinefeld and T. Marsland. Enhanced Iterative-Deepening Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol 16, 701–710, 1994.

Multiagent Planning with Partially Ordered Temporal Plans

Michael Brenner

Institut für Informatik, Universität Freiburg, 79110 Freiburg, Germany
brenner@informatik.uni-freiburg.de

Abstract

This paper* discusses the specifics of planning in multiagent environments. It presents the formal framework MAPL (“maple”) for describing multiagent planning domains. MAPL allows to describe both qualitative and quantitative temporal relations among events, thus subsuming the temporal models of both PDDL 2.1 and POP. Other features are different levels of control over actions, modeling of agents’ ignorance of facts, and plan synchronization with communicative actions. For single-agent planning in multiagent domains, we present a novel forward-search algorithm synthesizing MAPL’s partially ordered temporal plans. Finally, we present a general distributed algorithm scheme for solving MAPL problems with several coordinating planners. These different contributions are intended as as step towards a simple, yet expressive standard for the description of multiagent planning domains and algorithms. Such a standard could in the future allow cross-evaluation of Multiagent Planning algorithms on standardized benchmarks.

In this paper, we discuss the specific properties of planning in Multiagent Systems (MAS). With the term Multiagent Planning (MAP), we denote any kind of planning being performed in multiagent environments, meaning on the one hand that the planning process itself may be distributed among several *planning* agents, but also that individual plans can (and possibly must) take into account concurrent actions by several *executing* agents.

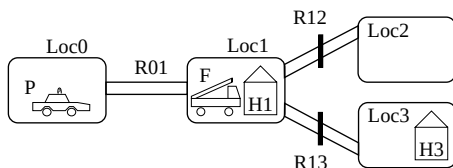


Figure 1: A multiagent planning problem

As a motivating example, Fig. 1 shows a simple MAP problem as appearing in the RoboCupRescue challenge [7]. Two autonomous agents, police force P and fire brigade F, are working in a city devastated by an earthquake. While F’s goal is to extinguish all burning houses, it is P’s capability and goal to clear the blocked roads. P’s position being Loc0 we will assume him being unaware of R12 and R13 being blocked. The agents’ actions have specific durations that may be exactly known only at execution time, sometimes because of specific execution parameters of the agents, sometimes because of intrinsic unpredictability of the environment: while, for example, moving through the town may take between 2

and 4 minutes and depend only on the map distance and speed of the agent, extinguishing a fire may take 1 to 4 hours depending on conditions unknown to the agents.

This trivial example shows several general features of MAP problems and plans: (1) Agents may be *unaware* of parts of the world state (P does not know whether R13 is blocked). (2) *Concurrent acting* is central to MAP (P can move to Loc1 and start clearing R13 while F is extinguishing H1, although both agents using the same road at the same time may be prohibited to avoid collisions). Modeling concurrency necessitates (2a) a description of which events may occur concurrently and which not, (2b) *metric time* for realistic descriptions of action durations and their relations, but (2c) synchronizing on actions of unknown (at least to some agent) duration demands *qualitative* use of time (e.g. “after P has cleared R13”). A specific usage of qualitative time is (3) synchronization on *communicative acts*, as in “F moves to Loc3 after P has informed him that R13 is clear”.

In their plans, agents must take other agents actions into account: F may “exploit” P’s clearing of R13 in his own plan but must also assure that he does not try to use a road that is also used by P at the same time. Especially, (4) cannot *control* occurrence or duration of other agents’ actions.

To address the representation problems (1)–(4) we introduce the Multiagent Planning Language MAPL (“maple”). Instead of propositional state representations MAPL allows non-boolean state variables (cf. also [5]). To model feature (1) each state variable may have the special value *unknown*, thereby avoiding representation of belief states as sets of possible states. A number of other advantages comes with the introduction of state variables; especially, for feature (2a), an intuitive definition of mutual exclusivity (i.e. the impossibility to execute some actions concurrently, cf. [1]) can be given that describes mutexes as *read-write locks* on state variables. According to this perspective, distributed planning can then be seen as detection or, even better, prevention of possible read-write locks *before* execution.

MAPL’s temporal model allows to combine (2b) quantitative and (2c) qualitative temporal information in plans, thus subsuming both the purely metric temporal model of PDDL 2.1 [4] and the purely qualitative model of Partial Order Planning [8]. At its core MAPL represents a multiagent plan as a Simple Temporal Network [3] in which each durative action is modeled by start and end events, possibly extended by invariant conditions. In the STN, both action durations and qualitative ordering relations are treated as constraints represented by closed, semi-open or open intervals. In so doing, not only can imprecisely known action durations be represented as intervals of the form $[\delta_1, \delta_2]$, but qualitative constraints like “after” can be described by (semi-)open intervals

*Definitions of the formal semantics of our Multiagent Planning Language as well as the algorithms for single and multi-agent planning are given in the long version of this paper [2].

like $(0, \infty)$. Fig. 2 shows F's plan for the problem of Fig. 1, the interval $(0, \infty)$ being represented as $<$.

Another new concept shown in Fig. 2 is (3) the use of speech acts as reference events for plan synchronization. On the one hand, this allows agents to refer to facts (especially those achieved by others) the change of which they do not influence or witness themselves. On the other hand, as explained later, speech acts allow agents to reveal only the minimum of information about their plans needed for coordination.

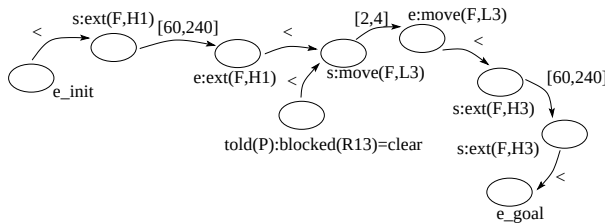


Figure 2: F's plan (including a communicative action by P)

A plan is only fully specified with (4) a *control function* describing which of the agents (or the environment) controls the occurrence of each event. With this function we can describe, e.g., that a specific agent is allowed to add and remove an action from his plan (control of the start event), but has no influence on its duration (end event controlled by the environment). During planning, having control of an event or not fundamentally changes its possible use and evaluation. For example, being able or unable to control the duration of an action will lead the planner to a fundamentally different heuristic evaluation of its use.

For a plan to be executable, it must be both temporally and logically consistent. The former criterion is reducible to consistency of the underlying STN. The latter, logical consistency, can be defined similarly to POP as the plan having no open conditions and no unsafe links, with the additional criterion that the plan must ensure that no mutex events may occur concurrently. For a plan to solve a certain agent's problem it must achieve his goals and also be consistent with the control function, i.e. only constraints involving events controlled by the respective agent must have been tightened by the planner.

How is planning in MAS carried out? It is obvious that the easiest way is to find a plan alone: assumed that F knows about P's capabilities, F can find a plan that solves his problems. Even if F does not know about P's concrete actions, this plan will provide clues about where help is needed and thus triggers cooperation. We see that (5) the capability for single-agent synthesis of multiagent plans is a basic requirement for MAP. We have developed a plan-space forward-search algorithm that can be used with any standard forward branching scheme and arbitrary plan metrics. We present two such metrics, the well-known makespan and the new min-MaxMakespan, the latter of which extends the former by assigning maximal possible duration to uncontrolled durative actions. We also describe how heuristic forward planning in the style of FF [6] can be extended to find MAPL plans. The current simple algorithm is sound, but not complete, i.e. there is a set of clearly distinguished MAPL problems it cannot solve yet. We are working on a sound, yet more complex version of the algorithm.

When several agents are planning and acting individually in a common environment, they will probably run into one of the following problems: (6) They won't be able to find individual plans solving their problems or (7) the plans found will conflict at execution time. MAP Literature has mostly treated only problem (7), implicitly assuming that plans can be found and that therefore separating planning from coordination is possible. In our opinion, coordination *during* the planning phase is indispensable in the case of problem (6) and advantageous for problem (7). We have therefore developed a general distributed planning algorithm that uses single-agent planning to synthesize partial plans and to trigger cooperation and coordination efforts as early as possible.

A key concept is the use of a *responsibility function* that assigns to each state variable an agent managing and controlling its changes over time. This agent will detect read-write conflicts in the agents' plans, i.e. possible execution conflicts, but will also provide information when another agent cannot achieve a (sub)goal involving that variable. In the basic form of the algorithm, the responsibility is static, but similar to approaches in Distributed CSP research [9] we will relax this assumption in future work. The idea of the algorithm is simple: in a reachability analysis the planning agent detects goals involving state variables he does not know about, cannot manipulate or could if only some earlier condition were satisfied. He contacts the responsible agents to receive more information or delegate a subgoal concerning the variable. The responsible agent answers the question or adopts a temporary goal to help the asking agent.

All contributions of this paper aim at clarifying the specific representational and algorithmic needs of MAP research. We hope that our representation will allow to conveniently describe diverse MAP domains for which researchers can propose and cross-evaluate algorithmic approaches just as diverse, thus promoting the field of Multiagent Planning.

References

- [1] A. Blum and M. Furst. Fast planning through planning graph analysis. *AIJ*, 90(1-2), 1997.
- [2] M. Brenner. Multiagent planning with partially ordered temporal plans. Technical report, Institut für Informatik, Universität Freiburg, Germany, 2003.
- [3] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *AIJ*, 49, 1991.
- [4] M. Fox and D. Long. *PDDL 2.1: an Extension to PDDL for Expressing Temporal Planning Domains*, 2002.
- [5] H. Geffner. Functional STRIPS: a more flexible language for planning and problem solving. In *Logic-Based Artificial Intelligence*. Kluwer, 2000.
- [6] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *JAIR*, 14, 2001.
- [7] H. Kitano et al. RoboCupRescue: Search and rescue in large-scale disasters as a domain for autonomous agents research. In *IEEE Intl. Conf. on Systems, Man and Cybernetics*, 1999.
- [8] Daniel Weld. An introduction to least commitment planning. *AI Magazine*, 15(4), 1994.
- [9] M. Yokoo and K. Hirayama. Algorithms for distributed constraint satisfaction: a review. *Autonomous Agents and Multi-Agent Systems*, 3(2), 2000.

Recognizing Plan/Goal Abandonment*

Christopher W. Geib,

Honeywell Laboratories

3660 Technology Drive

Minneapolis, MN 55418

christopher.geib@honeywell.com

Robert P. Goldman,

SIFT Inc.

2119 Oliver Ave. South

Minneapolis, MN 55405

rpgoldman@siftech.com

Abstract

The ability to recognize when an agent abandons a plan is an open problem in the plan recognition literature and is a significant problem if these methods are to be applied in real systems. This paper presents an explicit, formal, and implemented solution to the problem of recognizing when an agent has abandoned one of its goals based on a theory of probabilistic model revision.

1 Introduction

There is a large body of research on the topic of intent recognition or task tracking, focused on identifying the goals of a user from observations of their actions. However, none of this work has directly addressed the problem of recognizing when a user has abandoned a goal.

In general, the ability to infer abandoned goals is an operational requirement for any plan recognition system that is executing incrementally and continuously. Abandoning goals is something that any real observed agent will do. If a plan recognition system is unable to recognize this fact, the system will build up an ever increasing set of active or *open plans* that the agent has no intention of completing. A system attempting to find completions for these open plans will wind up considering unreasonable situations such as the first step of a time critical two step plan simple plan are taken but the plan is not elder a two step plan with a required seconds or minutes duration but not attempting the second step of the plan until days or weeks later. Unfortunately, existing plan recognition systems cannot draw such inferences.

2 Background

Recent work in execution based plan/intent recognition [Bui *et al.*, 2002; Geib and Goldman, 2001; Goldman *et al.*, 1999] has been based on a model of the execution of simple hierarchical task network (HTN) plans [Erol *et al.*, 1994].

The idea behind this approach is that initially the executing agent has a set of goals and chooses a set of plans to execute to achieve these goals. The set of plans chosen determines a

*This material is based upon work supported by DARPA/IPTO and the Air Force Research Laboratory under Contract No. F30602-02-C-0116

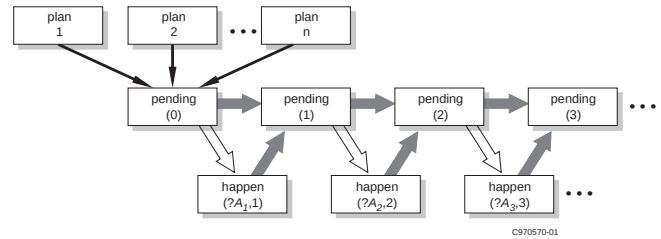


Figure 1: A simple model of plan execution.

pending set of primitive actions. The agent executes one of the pending actions, generating a new pending set from which the next action will be chosen, and so on.

This process is illustrated in Figure 1. In this light, the observed actions are nothing more than the observations of a hidden Markov model and the process of plan recognition is the inference of the underlying state of the model. Space prohibits a full exposition of this approach. We refer readers to [Geib and Goldman, 2001; Goldman *et al.*, 1999] for a more complete discussion.

3 Exact Solutions

Given our model of plan execution, a formal and general model of probabilistic abandonment of goals, will be forced to deal with an exponentially larger model and will be required to obtain the prior probabilities that each of the goals is abandoned. For real world applications this approach is simply untenable. A complete discussion of these issues is provided in the full paper.

4 Model Revision

Rather than explicitly considering all of the possible plans that could be abandoned, the problem can be looked at as a question of model revision. If we are using a model of plan execution that does not consider plan abandonment to recognize observation streams in which the agent is abandoning plans, we expect that the computed probabilities for the observation streams will be quite low. Laskey [1991], Jensen [Jensen *et al.*, 1990], and others have suggested that cases of an unexpectedly small $P(\text{observations}|\text{Model})$ should be used as evidence of a model mismatch.

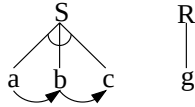


Figure 2: A very simple example plan library.

Instead of the general $P(\text{observations}|\text{model})$ statistic we propose the probability that none of the observed actions in a subsequence (from say s to t) contribute to one of the goals (call it G), and we denote it $P(\text{notContrib}(G, s, t)|\text{model}, \text{observations})$. If this probability gets unexpectedly small, we consider this as evidence of a mismatch between the model and the real world. Namely the model predicts that the agent is still working on the goal, while the agent may have abandoned it.

4.1 Computing *notContrib*

Consider the plan library shown in Figure 2. The first plan is a very simple plan for achieving S by executing a , b , and c and the second plan for R has only the single step of g . Next, assume the following sequence of observations:

$$\text{happen}(a, 0), \text{happen}(b, 1), \text{happen}(g, 2), \text{happen}(g, 3).$$

In this case we know that at time 0 and 1 that the agent has as a goal achieving S . Let us assume that all of the elements of the pending set are equally likely to be selected for execution next. Note that this is an assumption that we will make for the rest of this paper. Nothing about the algorithm hinges on this uniformity assumption. It is made solely for ease of computation and discussion.

Given this assumption, the probability of seeing c at time 2 is given by: $(m/|PS_2|)$ where m is the number of elements in the pending set that have c as the next action. The probability that we don't see c (that is the probability that any other element of the pending set is chosen at time 2) is just:

$$1 - (m/|PS_2|)$$

or more generally the probability that we have seen b at time $(s-1)$ and not seen c by time t :

$$\prod_{i=s}^t 1 - (m/|PS_i|)$$

To handle partially ordered plans, this formula must be generalized slightly. With partially ordered plans it is possible for more than a single next action to contribute to the specified root goal. Thus, if $m_{q,i}$ represents the number of elements (with any next action) in the pending set at time i that contribute to goal q , $(s-1)$ is the last time we saw an action contribute to q and t is the current time, $P(\text{notContrib}(q, s, t)|\text{model}, \text{obs}) =$

$$\prod_{i=s}^t 1 - (m_{q,i}/|PS_i|)$$

Thus, under the assumptions that we have made we can compute the probability of the subsequence of actions not

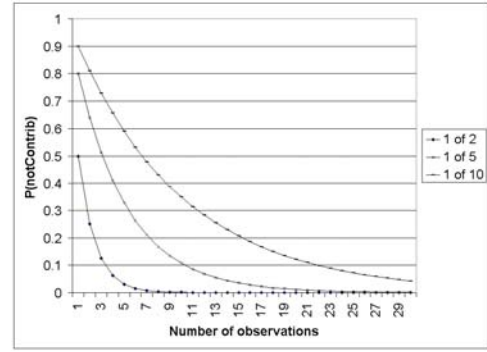


Figure 3: Required Evidence Theoretical Curves.

contributing to a given plan or goal. By computing this value and setting a threshold, we can consider any drop in this probability below the threshold as sufficient evidence of a model mismatch and revise the model to reflect the goals abandonment. This requires removing all the elements from the current pending set that contribute to the abandoned goal. Modeling the rest of the plans continues as before.

4.2 Evidential Requirements

This approach creates an interesting linkage between the size of the pending set, the number of elements that contribute to the goal of interest, and the number of actions that **don't** contribute to the goal that must be observed before the goal is considered abandoned.

Figure 3 shows three theoretical curves for the probability of *notContrib* for different sets of values. The curves are labeled with the number of actions that contribute to a goal and the size of the pending set. Thus the curve labeled "1 of 2" shows the drop in the probability given each observation if there is one action that contributes to the desired goal out of a pending set of size two. Note that in these curves, we are again making the assumption that all of the actions in the pending set are equally likely to be chosen.

Notice that as the ratio of the number of contributing actions to the size of the pending set drops the number of actions required to drive *notContrib* down to a particular threshold value increases significantly. We will see the effects of this in the empirical results.

4.3 Estimating $P(\text{abandoned}(g)|Obs)$

If we compute $P(\text{notContrib}(g, s, t)|\text{model}, \text{obs})$ for each g and threshold our explanations as described in the previous section, we can now produce explanations of the observations in which goals have been abandoned. By considering the complete and covering set of such explanations for the observations we can estimate the probability of a specific goal's abandonment. It is given by:

$$P(\text{abandoned}(g)|Obs) \approx \frac{\sum_e \text{Exp}_{A(g)} P(e|Obs)}{\sum_e \text{Exp} P(e|Obs)}$$

where Exp represents the set of all explanations for the observations, and $\text{Exp}_{A(g)}$ represents the set of explanations in which goal g is marked as abandoned.

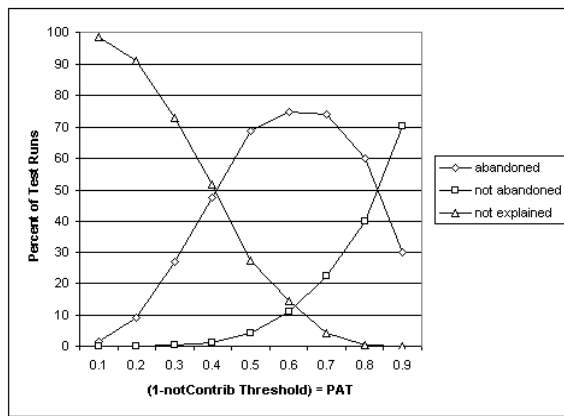


Figure 4: Empirical Accuracy

5 Accuracy

To test this theory we have extended Geib and Goldman's Probabilistic Hostile Agent Task Tracker (PHATT) to estimate goal abandonment. We will not cover the details of the PHATT algorithm here; instead we refer the interested reader to [Geib and Goldman, 2001]. We used a very simple plan library with three root goals each having eight unordered steps. To generate test cases, we chose an ordering for the actions for each of the three goals. These plans were then randomly interleaved preserving the intra-plan ordering. To simulate goal abandonment, at each time step one of the goals could be chosen for abandonment. If chosen, all remaining steps of the plan were removed from the observation stream. When given to PHATT, each of these test sequences produced one of three possible results:

abandoned where PHATT believed with probability greater than 0.5 that the correct goal had been abandoned.

not abandoned where PHATT did not believe with probability greater than 0.5 that the correct goal had been abandoned

not explained where PHATT was unable to explain the set of observations. In all cases, this was a result of the system believing that a goal was abandoned before it actually had been. The system was therefore unable to account for the remaining actions in that test data point.

The results of one thousand such randomly generated data points at each of nine *notContrib* threshold values between 0.1 and 0.9 can be seen in Figure 4. To aid in understanding, the X-axis plots $(1 - \text{the notContrib threshold})$ which we will call the *probability of abandonment threshold* (PAT). The y-axis plots the percentage of test points for each of the possible results at that threshold value. The results confirm our intuitions.

The number of test sequences that are **not explained** drops to zero as the PAT is raised. The PAT is specifying how much evidence the algorithm needs to have before it can consider a goal abandoned. Since the system's failure to explain a test sequence is a result of prematurely believing a goal has been abandoned, as the PAT rises and more evidence is required this number should drop to zero.

The number of test sequences that are **not abandoned** rises as the PAT is raised. As the threshold rises the system requires more and more evidence to be convinced that the goal has been abandoned. This allows more data points to reach the end of the observation sequence without being convinced of the goal's abandonment.

Finally, **abandoned** rises and peaks giving the algorithm a maximum accuracy of about seventy five percent at a PAT of between 0.6 and 0.7. The algorithm's subsequent dip is again a result of the increasing confidence in abandonment required by the rising PAT. As with **not abandoned** at this point the system's accuracy is falling prey to the limited length of the test sequences. Since each test sequence is of limited length the test runs are ending before enough evidence can be observed for the higher PAT values. Figure 3 will show us why.

Consider the curve in Figure 3 labeled "1 of 5." This is approximately the ratio of contributing actions to the size of the pending set in this example. At a PAT of 0.9 the system will require approximately ten actions in a row that do not contribute to the goal in order to convince itself of the goals abandonment. Since the longest any of the tests can be is twenty three actions if the plan is abandoned more than half way through the observation stream it will have a hard time producing enough evidence to convince the system.

6 Conclusions

This paper presents a solution to the problem of recognizing when an agent has abandoned a goals based on probabilistic model revision. A number of issues are covered in more detail in the full version of this paper available from the author.

References

- [Bui et al., 2002] Hung H. Bui, Svetha Venkatesh, and Geoff West. Policy recognition in the abstract hidden markov model. In *Technical Report 4/2000 School of Computer Science, Curtin University of Technology*, 2002.
- [Erol et al., 1994] Kutluhan Erol, James Hendler, and Dana S. Nau. UMCP: A sound and complete procedure for hierarchical task network planning. In *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems (AIPS 94)*, pages 249–254, 1994.
- [Geib and Goldman, 2001] Christopher W. Geib and Robert P. Goldman. Plan recognition in intrusion detection systems. In *Proceedings of DISCEX II, 2001*, 2001.
- [Goldman et al., 1999] Robert P. Goldman, Christopher W. Geib, and Christopher A. Miller. A new model of plan recognition. In *Proceedings of the 1999 Conference on Uncertainty in Artificial Intelligence*, 1999.
- [Jensen et al., 1990] Finn Verner Jensen, Bo Chamberlain, Torsten Nordahl, and Frank Jensen. Analysis in hugin of data conflict. In *Proceedings of the 6th Conference on Uncertainty in Artificial Intelligence*, 1990.
- [Laskey, 1991] Kathy Blackmond Laskey. Conflict and surprise: Heuristics for model revision. In *Proceedings of the 7th Conference on Uncertainty in Artificial Intelligence*, 1991.

Automated Generation of Understandable Contingency Plans

Max Horstmann and Shlomo Zilberstein

University of Massachusetts

Department of Computer Science

Amherst, MA 01003, U.S.A.

{horstman, zilberstein}@cs.umass.edu

Abstract

Markov Decision Processes (MDPs) and contingency planning (CP) are two widely used approaches to planning under uncertainty. MDPs are attractive because the model is extremely *general* and because many algorithms exist for deriving *optimal* plans. In contrast, CP is normally performed using heuristic techniques that do not guarantee optimality, but the resulting plans are more *compact* and more *understandable*. The inability to present MDP policies in a clear, intuitive way has limited their applicability in some important domains. We introduce an anytime algorithm for deriving contingency plans that combines the advantages of the two approaches.

1 Introduction

Two closely related decision-theoretic planning paradigms have emerged in the area of planning under uncertainty: Markov Decision Processes (MDPs) have become a general framework for planning and reinforcement learning. Contingency Planning (CP) on the other hand emphasizes compactness and understandability of the solution. In both cases, an agent is manipulating an environment by performing actions with uncertain outcomes. In each move, the agent receives some reward and the overall goal is to maximize the cumulative reward.

MDPs are solved by deriving a *policy* which maps domain states to actions, typically represented as a large table. Several existing algorithms can construct optimal policies, but the results are not easy to visualize or understand. CP is another widely used approach in stochastic domains which allows plans to include branches that may depend on arbitrary memory states.

It is clear why optimal plans are desirable, but the issue of understandability is less obvious. In fact, for some applications, a plan represented as a large table mapping states to actions may be perfectly suitable. However, the lack of clarity has limited the adoption of MDP planning in some application domains such as space exploration with unmanned rovers. In this domain, communication with the agent is restricted and due to high mission costs and associated risks, understanding and verifying plans are crucial.

We examine the relationship between policies and contingency plans and define a precise measure of complexity of contingency plans. Then, we introduce a technique for automated contingency plan generation and briefly describe our experimental results. Finally, we conclude with a summary and future research directions.

2 Formal Problem Description

A **Markov Decision Process** (MDP) with goal states is a tuple (S, A, P, R, s_0, G) , where $S = S_1 \times \dots \times S_n$ is a factored state space and S_i is the (finite) domain of feature i . A is a set of actions, P the transition probabilities, R the reward expectations, s_0 a known initial state and G a (possibly empty) set of absorbing terminal states.

A **policy** $\pi : S \rightarrow A$ is a mapping from states to actions. Following the notation in [Littman et al., 1998], a **Contingency Plan** for an MDP is a tuple $(V, E, v_0, \rho, \delta)$, where (V, E) is a directed graph with start state $v_0 \in V$. $\rho : V \rightarrow A$ associates an action with each node of the graph. $\delta : E \rightarrow 2^S$ labels each edge with a set of states.

An **interval label descriptor** maps state sets to sets of intervals over the feature domains defined as follows: $\mathcal{D}_{int}(\delta) : E \rightarrow \{I_1, \dots, I_k\}$ where I_j is the set: $\{[L_1; U_1], \dots, [L_n; U_n] \mid 1 \leq L_i \leq U_i \leq |S_i| \forall 1 \leq i \leq n\}$.

The **complexity** of a contingency plan does not reflect the *computational* complexity of constructing it, but is a measure of how hard it is to understand. Formally: $Complex(MDP, CP, \mathcal{D}_{int}(\delta)) = |V| \cdot ABF \cdot ALDS$, where ABF is the average branching factor of the graph and $ALDS$ is the average label descriptor size. The size of a single label descriptor is the number of constrained intervals.

The **value** of a policy reflects the expected discounted return when the agent selects action using π , starting from the initial state. It can be computed with standard algorithms such as value iteration. Similarly, the value of a contingency plan is the expected discounted return when the agent selects action using ρ , starting from the initial node. By considering the Markov chain induced by the state set $S \times V$, the same methods used to evaluate MDP policies can be used to evaluate a contingency plan.

3 Automated Contingency Plan Generation

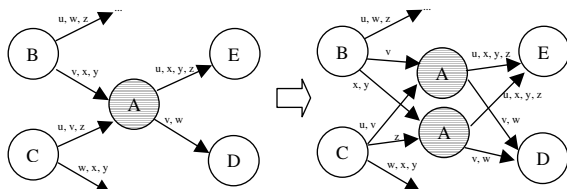
Our approach to automatically generate more understandable (less complex) contingency plans relies on solving first the underlying MDP, obtaining an optimal policy π and creating an equivalent initial contingency plan from it: $V = \{v_1, \dots, v_{|A|}\}$, $E = V \times V$, $\rho(v_i) = \pi(a_i)$, $\delta(v_i, v_j) = \{s \in S | \pi(s) = a_j\}$. Then, the following operators are used to reduce the complexity of the plan.

By performing a depth-first search over the set of all reachable edges between the state-node pairs in $S \times V$, a **reachability analysis** can be applied to remove states from label descriptors, edges from nodes and nodes from the graph.

Since a benefit of interval label descriptors is the ability to capture easily large sets of states and simplify the branch conditions, n -dimensional boxes to describe them more compactly can be generated by **merging intervals** along the dimensions of the state space. This is in particular very effective if the order of feature values corresponds to some natural ordering.

By **trading off optimality for clarity**, interval descriptors can be merged even if there are a few states that make the merge operator not admissible.

Most importantly, the **node-split** operator splits a node of the plan graph into two and adapts the incoming and outgoing edges according to a given partition of the state space. This gives contingency plans their real strength: The current node of the graph then represents partial information about the current state (“context information”), so less details of the actual state may be needed to make the branch decision on the next step. For instance, with a state set $S = \{u, v, w, x, y, z\}$ and the partition $S_1 = \{u, v, w\}$, $S_2 = \{x, y, z\}$, the result of a split operator is illustrated below:



4 Experimental Results

It turns out that combining these operators can significantly reduce the complexity of a contingency plan. This leads to the following search problem: Given the initial contingency plan defined above, find a contingency plan with the lowest complexity, not giving up more than a certain fraction ϵ of the optimality of the policy π . Unfortunately, two major obstacles make an exhaustive search intractable: The node-split operator is parameterized by a partition of the MDP state space, leading to a large number of possible splits and thus to a large branching factor. In addition, the computational complexity of reachability analysis is fairly high.

As a result, we decided to relax the requirement of minimizing complexity and developed a method to combine the operators and automatically generate some understandable plans. Performance improvement can be obtained by choosing a partition of the state space, split the nodes according

to this partition, perform a reachability analysis and merge the interval descriptors subsequently. We implemented a hill-climbing algorithm that used some domain-specific heuristics for choosing a partition and performed random restarts, leading to an interruptible anytime algorithm that performed well on a simple set of test problems.

For instance, in a maze domain with stochastic move actions, the algorithm took advantage of bottlenecks, tunnels and alternative branches. In a simulated planetary rover domain with uncertain action durations, some hints for good split partitions lead to dramatically decreased complexity. The resulting plan instructed the agent to perform certain actions until a resource (one feature of the factored state space) drops below a certain threshold. This was very easy to understand for humans unlike the optimal policy given in the form of a huge lookup-table.

5 Conclusions

The main objective of this work has been to find solutions for decision-theoretic planning problems that are optimal (or near-optimal) and also compact and understandable. We defined a measure of the complexity of contingency plans, reflecting their size, branching factor and the size of the label descriptors. The results we gained by experimenting with several plan-transformation operators are encouraging and show the potential of automated generation of understandable contingency plans.

There are several interesting directions for future research. First, a better measure of the plan complexity could be developed. Second, additional operators could be added. Third, the language for representing edge labels could be enriched. Finally, methods for finding good split partitions have to be found to create more reliable algorithms. The results reported in this paper provide a good framework for future exploration of these research directions.

Acknowledgments

Support for this work was provided in part by NASA under grant NAG-2-1463. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of NASA.

References

- [Bresina et al., 2002] J. Bresina, R. Dearden, N. Meuleau, S. Ramakrishnan, D. Smith, and R. Washington. Planning Under Continuous Time and Resource Uncertainty: A Challenge for AI. *Conference on Uncertainty in Artificial Intelligence*, 2002.
- [Feng and Hansen, 2002] Z. Feng and E.A. Hansen. Symbolic Heuristic Search for Factored Markov Decision Processes. *Eighteenth National Conference on Artificial Intelligence*, 2002.
- [Hansen and Zilberstein, 2001] E.A. Hansen and S. Zilberstein. LAO*: A Heuristic Search Algorithm that Finds Solutions with Loops. *Artificial Intelligence*, 129(1-2):35-62, 2001.
- [Littman et al., 1998] M.L. Littman, J. Goldsmith, and M. Mundhenk. The Computational Complexity of Probabilistic Planning. *Journal of Artificial Intelligence Research*, 9:1-36, 1998.

A Planning Algorithm for Predictive State Representations

Masoumeh T. Izadi and Doina Precup

School of Computer Science
McGill University
Montreal, Canada

Abstract

We address the problem of optimally controlling stochastic environments that are partially observable. The standard method for tackling such problems is to define and solve a Partially Observable Markov Decision Process (POMDP). However, it is well known that exactly solving POMDPs is very costly computationally. Recently, Littman, Sutton and Singh (2002) have proposed an alternative representation of partially observable environments, called predictive state representations (PSRs). PSRs are grounded in the sequence of actions and observations of the agent, and hence relate the state representation directly to the agent's experience. In this paper, we present a policy iteration algorithm for finding policies using PSRs. In preliminary experiments, our algorithm produced good solutions.

1 Predictive State Representation

We assume that we are given a system consisting of a discrete, finite set of n states S , a discrete finite set of actions A , and a discrete finite set of observations $\mathcal{O}$. The interaction with the system takes place at discrete time intervals. The initial state of the system s_0 is drawn from an initial probability distribution over states I . On every time step t , an action a_t is chosen according to some policy. Then the underlying state changes to s_{t+1} and a next observation o_{t+1} is generated. The system is Markovian, in the sense that for every action, the transition to the next state is generated according to a probability distribution described by an $(n \times n)$ transition matrix T^a . Similarly, for a given observation o and action a , the next observation is generated according to an $(n \times n)$ diagonal observation matrix $O^{a,o}$, where $O_{ii}^{a,o}$ is the probability of observation o when action a is selected and state i is reached. Since we are interested in optimal control, rather than prediction, we also assume that there exists a set of reward vectors R^a for each action a , where R_i^a is the reward for taking action a in underlying state i .

PSRs are based on the notion of tests. A *test* is an ordered sequence of action-observation pairs $q = a_1 o_1 \dots a_k o_k$. The *prediction* for test q is the probability of the sequence of observations $o_1 \dots o_k$ being generated, given the sequence of ac-

tions $a_1 \dots a_k$. The prediction for a test q given prior history h , denoted $p(q|h)$, is the probability of seeing the sequence of observations of q after seeing history h and taking the sequence of actions specified by q . For any set of tests Q , its prediction vector is:

$$p(Q|h) = [p(q_1|h), \dots, p(q_{|Q|}|h)]$$

A set of tests Q is a PSR if its prediction vector forms a sufficient statistic for the dynamical system, i.e., if all tests can be predicted based on $p(Q|h)$. Of particular interest is the case of linear PSRs, in which there exists a projection vector m_q for any test q such that

$$p(q|h) = p(Q|h)^T m_q$$

Littman et al. also define an *outcome* function u mapping tests into n -dimensional vectors defined recursively by: $u(\epsilon) = e_n$ and $u(a o q) = (T^a O^{a,o} u(q)^T)^T$, where ϵ represents a null test and e_n is the $(1 \times n)$ vector of all 1s. Each component $u_i(q)$ indicates the probability of the test q when its sequence of actions is applied from state s_i . A set of tests $Q = \{q_i | i = 1, 2, \dots, k\}$ is called linearly independent if the outcome vectors of its tests $u(q_1), u(q_2), \dots, u(q_k)$ are linearly independent. Using this definition, such a set Q can be found by a simple search algorithm in polynomial time, given the POMDP model of the environment. Littman, Sutton and Singh (2002) showed that the outcome vectors of the tests in Q can be linearly combined to produce the outcome vector for any test.

2 Policy evaluation using PSRs

We assume that we are given a policy $\pi : \mathcal{H} \rightarrow \mathcal{A}$ and that the initial start state of the system, s_0 , is drawn according to the starting probability distribution I . If we consider a given horizon t , only a finite number of tests of length t are possible when starting from I . Let Ψ_t be this set of possible tests.

The value of a memoryless policy π with respect to a given start state distribution I is the expected return over all possible tests that can occur when the starting state is drawn from I and then behavior is generated according to policy π :

$$V^\pi = \sum_{q \in \Psi_t} V(q|\pi) = \sum_{q \in \Psi_t} P(q|I, \pi) R(q|I, \pi),$$

where $R(q|I, \pi)$ is the expected return for test q given that the initial state is drawn from I and policy π is followed.

Let U be the $(|S| \times |\Psi_t|)$ matrix formed by concatenating the outcome vectors for all tests in Ψ_t , and U^+ be its pseudoinverse. The columns of U define the probability of each test in Ψ_t when applied from each underlying state. Consequently, IU represents the probability of the tests in Ψ_t when starting in I .

For each action-observation combination ao , we can define a $(|\Psi_t| \times |\Psi_t|)$ projection matrix:

$$M^{ao} = (U^+ T^a O^{ao} U)^T$$

and a projection vector:

$$m^{ao} = (U^+ T^a O^{ao} e_n^T)^T.$$

Considering the probability distribution over tests that π generates and the expected return of a test q given I and π , we have:

$$V(q|I, \pi) = \sum_{i=1}^t \frac{IU \prod_{j=1}^i (M^{a_j o_j})^T U^+}{IU m_{a_1 o_1 \dots a_i o_i}^T} \\ * R^{a_i} P(a_i | \pi, a_1 o_1 \dots a_{i-1} o_{i-1})$$

This evaluation method requires a large amount of precomputation, but it can be useful if a small horizon suffices to get a good policy.

3 Policy iteration for PSRs

Similar to POMDPs in PSRs we can define action-value functions on the level of tests. The action-value function for taking action a after test q can be computed as:

$$Q_i^\pi(q, a) = R^a u(q) + \sum_{o \in O} \sum_{q' \in \Psi_{i-1}} p(q'|qao) V(q' | \pi, u(qao))$$

Then, the policy of the agent can be improved by choosing an action greedily with respect to this action-value function:

$$\pi^*(t) = \arg \max_a Q_i^\pi(a, t)$$

The agent must, in other words, select the best policy tree rooted at each decision point and each time step. Therefore the total running time of the algorithm is $O(|S|^3(|A||O|)^h)$ and the complexity of the algorithm is only single-exponential in the horizon time.

4 Experimental results

We experimented with a standard gridworld navigation task used in the POMDP literature (Cassandra, 1994; Parr & Russell, 1996). The environment is a (4×4) grid. The agent has four actions, N, S, E, and W, which change its location deterministically to one of the four neighboring states. There is one goal state, in the lower right corner, which generates a distinct observation and a reward of +1. All the other states are perceptually aliased and generate no reward. The initial probability distribution is uniform over all states except the goal. Taking any action in the goal state moves the agent uniformly randomly to one of the other states.

For this problem we have 6 possible combinations of action-observation (ao) pairs: 1-Nnothing, 2-Wnothing, 3-Enothing, 4-Snothing, 5-Egoal, 6-Sgoal. The set of core tests

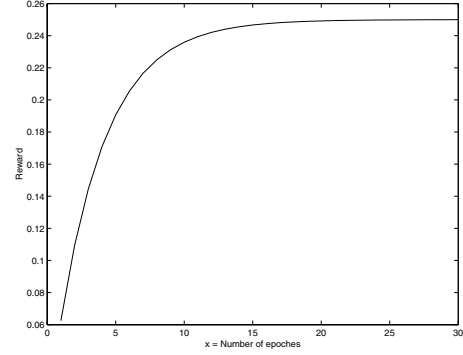


Figure 1: Policy quality vs. number of iterations for the 4x4 grid world

Q found consists of 16 linearly independent tests: 4, 5, 6, 15, 36, 46, 436, 446, 4436, 3336, 1336, 44336, 43336, 31336. We tried this problem for finite horizon case with a discount factor $\gamma = 0.8$. Figure 1 indicates the performance of the algorithm on this problem.

5 Conclusion and future work

The key difficulty for our planning algorithm is that the number of possible tests grows exponentially with the horizon length. Hence, the algorithm cannot be used for large problems or for the infinite horizon case. However, this is not worse than other existing exact solution methods for solving POMDPs. Our hope is that good approximations of the optimal solution can be found efficiently.

References

- [Cassandra *et al.*, 1994] A. R. Cassandra, L. P. Kaelbling, and M. L. Littman. Acting Optimally in Partially Observable Stochastic Domains. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 1994.
- [Littman, 1996] . L. Littman. Algorithms for Sequential Decision Making. PhD thesis, Brown University, Providence, RI, March 1996.
- [Littman *et al.*, 2002] M. L. Littman, R. S. Sutton, and S. Singh. Predictive representations of state. *Advances in Neural Information Processing Systems 14. (Proceedings of the 2001 conference)*. MIT Press.
- [Parr and Russell, 1995] R. Parr, S. Russell. Approximating Optimal Policies for Partially Observable Stochastic Domains. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*.

Parallelizing State Space Plans Online

Romeo Sanchez Nigenda & Subbarao Kambhampati

Department of Computer Science and Engineering

Arizona State University, Tempe AZ 85287-5406

Email: {rsanchez,rao}@asu.edu

Abstract

Searching for parallel solutions in state space planners is a challenging problem, because it would require the planners to branch on all possible subsets of parallel actions, exponentially increasing their branching factor. We introduce a variant of our heuristic state search planner *AltAlt*, which generates parallel plans by using greedy online parallelization of partial plans. Empirical results show that our online approach outperforms post-processing (offline) techniques in terms of the quality of the solutions returned.

1 Introduction

Parallel plans allow concurrent execution of multiple actions at each time step. While disjunctive planners such as Graphplan [Blum and Furst, 1997], SATPLAN [Kautz and Selman, 1996] and GP-CSP [Do and Kambhampati, 2000] seem to have no trouble generating such parallel plans, planners that search in the space of states are overwhelmed by this task [Haslum and Geffner, 2000]. The problem for state space planners is the exponential branching factor of the search space. One approach to solve this problem is to post-process the sequential plans generated by the state space planners. This could easily be done—using approaches such as those explored by Backstrom [Backstrom, 1998], but they are limited to transforming the sequential plan given as input. An alternative that we introduce in this article, involves incremental online parallelization. Specifically, our planner *AltAlt^p* [Sanchez and Kambhampati, 2003], which is a variant of the *AltAlt* planner [Sanchez et al., 2000; Nguyen et al., 2000], attempts to parallelize a heuristically selected search branch with other independent actions, and then tries to rearrange the evolving parallel plan using a plan-compression algorithm.

We will provide a brief background on *AltAlt* planning system on Section 2. Section 3 describes the generation of parallel plans in *AltAlt^p*. Section 4 presents some empirical evaluation of *AltAlt^p*. Finally, Section 5 summarizes our contributions.

2 *AltAlt* Background

The *AltAlt* planning system is based on a combination of Graphplan [Blum and Furst, 1997] and heuristic state space search [Haslum and Geffner, 2000] technology. The problem specification and the action template description are first fed to a Graphplan-style planner, which constructs a planning graph for that problem in polynomial time. This planning graph structure is then fed to a heuristic extractor module that is capable of extracting a variety of effective heuristics [Nguyen et al., 2000]. These heuristics, along with the problem specification, and the set of ground actions in the final action level of the planning graph are then fed to a regression state-search planner.

3 Plan Compression Algorithm

AltAlt^p considers at each step during search only those (pairwise parallel) actions that individually regress to states with lower heuristic estimates than their parent node to parallelize (fatten) a search branch. This greedy nature of the fattening procedure while useful in avoiding the addition of irrelevant actions to the plan, may also sometimes preclude actions that are ultimately relevant. When this happens, the parallel length of the solution plan is likely to be worsened, as more steps may be needed to support the preconditions of such actions. In order to offset this negative effect of greediness, *AltAlt^p* re-arranges the partial plan to promote such actions higher up the search branch using a plan-compression algorithm (See Figure 1). Specifically, before expanding a given node S , *AltAlt^p* checks to see if any of the actions in A_s leading to S from its parent node, can be pushed up to higher levels in the search branch. The push-up procedure is called each time before a node gets expanded, and it will try to compress the current partial plan. For each of the actions $a \in A_s$ we find the highest ancestor node S_x of S in the search branch to which the action can be applied (i.e., it gives some literal in S_x without deleting any other literals in S_x , and it is pairwise independent of all the actions currently leading out of S_x). Once S_x is found, a is then removed from the set of actions leading to S and introduced into the set of actions leading out of S_x (to its child in the current search branch). Next, the states in the search branch below S_x are adjusted to reflect this change. The adjustment involves recomputing the regressions of all the search nodes below S_x . At first glance,

```

pushUP(S)
   $A_s \leftarrow \text{get actions leading to } S$ 
  forall  $a \in A_s$ 
     $x \leftarrow 0$ 
     $S_x \leftarrow \text{get parent node of } S$ 
    Loop
       $A_x \leftarrow \text{get actions leading to } S_x$ 
      If ( $\text{parallel}(a, A_x)$ )
         $x \leftarrow x + 1$ 
         $S_x \leftarrow \text{get parent node of } S_{x-1}$ 
      Else
         $A_{x-1} \leftarrow A_{x-1} + a$ 
         $A_s \leftarrow A_s - a$ 
    End Loop
    /** Adjusting the partial plan
     $S_x \leftarrow \text{get highest ancestor } x \text{ in history}$ 
    createNewBranchFrom( $S_x$ )
    while  $x > 0$ 
       $S_{new} \leftarrow \text{regress } S_x \text{ with } A_{x-1}$ 
       $S_x \leftarrow S_{new}$ 
       $x \leftarrow x - 1$ 
  END;

```

Figure 1: Push Up Procedure

this might seem like a transformation of questionable utility since the preconditions of a (and their regressions) just become part of the descendants of S_x , and this does not necessarily reduce the length of the plan. We however expect a length reduction because actions supporting the preconditions of a will get “pushed up” eventually during later expansions.

4 Empirical Evaluation

We compare our online parallelization approach to post-processing, we have implemented Backstrom’s ‘Minimal Deordering Algorithm’ from [Backstrom, 1998], and used it to post-process the sequential plans produced by *AltAlt*. We can see on Figure 2 some empirical results on the Zeno domain [Long and Fox, 2002].

As expected, the original *AltAlt* has the longest plans since it allows only one action per time step. The plot shows that post-processing techniques do help in reducing the makespan of the plans generated by *AltAlt*. However, we also notice that *AltAlt^p* outputs plans with better makespan than either *AltAlt* or *AltAlt* followed by post-processing. This shows that online parallelization is a better approach than post-processing sequential plans. *AltAlt^p* also does not result in any additional run-time overhead with respect to *AltAlt*. Further experiments reported in [Sanchez and Kambhampati, 2003] show that our approach is competitive or outperforms other state of the art approaches in generating parallel plans.

5 Conclusions

We have introduced an approach to generate parallel plans in the context of state space search. Our approach tries to avoid the branching factor blow up by greedy and online parallelization of the evolving partial plans. A plan compression procedure is used to offset the ill effects of the greedy search. Our empirical results show that our approach seems to return better quality plans than post-processing the sequential plans.

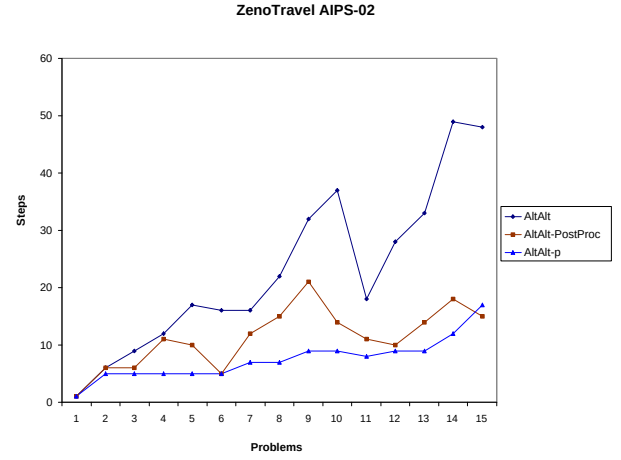


Figure 2: *AltAlt* and Post-Processing vs *AltAlt^p*

In future, we plan to adapt the *AltAlt^p* approach to metric temporal domains, where the need of concurrency is more pressing. It is in these scenarios where more powerful heuristics based on resource usage can be further investigated to improve the parallelism of state space planners.

References

- [Blum and Furst, 1997] A. Blum and M.L. Furst. Fast planning through planning graph analysis. In *Artificial Intelligence*. 90(1-2). 1997.
- [Backstrom, 1998] C. Backstrom. Computational Aspects of Reordering Plans In *Journal of Artificial Intelligence Research*, 1998.
- [Long and Fox, 2002] D. Long and M. Fox. Results of the AIPS 2002 Planning Competition. URL: <http://www.dur.ac.uk/d.p.long/competition.html>.
- [Kautz and Selman, 1996] H. Kautz and B. Selman. Pushing the envelope: Planning propositional logic and stochastic search. In *Proc. AAAI 1996*. Portland, OR.
- [Do and Kambhampati, 2000] Minh B. Do, and S. Kambhampati. Solving Planning Graph by Compiling it into a CSP. In *Proc. AIPS 2000*
- [Haslum and Geffner, 2000] P. Haslum and H. Geffner. Admissible Heuristics for Optimal Planning. In *Proc. AIPS 2000*.
- [Sanchez et al., 2000] R. Sanchez, X. Nguyen and S. Kambhampati. *AltAlt*: Combining the advantages of Graphplan and Heuristics State Search. In *Proc. KBCS 2000*. Bombay, India.
- [Sanchez and Kambhampati, 2003] R. Sanchez and S. Kambhampati. *AltAlt^p*: Online Parallelization of Plans with Heuristic State Search. To appear in *JAIR*.
- [Nguyen et al., 2000] X. Nguyen, S. Kambhampati, and R. Sanchez Nigenda. Planning Graph as the Basis for deriving Heuristics for Plan Synthesis by State Space and CSP Search. In *Journal of Artificial Intelligence*, 2002.

A lookahead strategy for solving large planning problems*

Vincent Vidal

CRIL - Université d'Artois
rue de l'Université - SP 16
62307 Lens Cedex, France
vidal@cril.univ-artois.fr

Abstract

Relaxed plans are used in the heuristic search planner FF for computing a numerical heuristic and extracting helpful actions. We present a novel way for extracting information from the relaxed plan and for dealing with helpful actions, by considering the high quality of the relaxed plans. In numerous domains, the performance of heuristic search planning and the size of the problems that can be handled have been drastically improved.

1 Computing and using lookahead states

In classical forward state-space search algorithms, a node in the search graph represents a planning state and an arc starting from that node represents the application of one action to this state, that leads to a new state. In order to ensure completeness, all actions that can be applied to one state must be considered. The order in which these states will then be considered for development depends on the overall search strategy: depth-first, breadth-first, best-first. . .

Let us now imagine that for each evaluated state S , we knew a valid plan P that could be applied to S and would lead to a state closer to the goal than the direct descendants of S . It could then be interesting to apply P to S , and use the resulting state S' as a new node in the search. This state could be simply considered as a new descendant of S .

We have then two kinds of arcs in the search graph: the ones that come from the direct application of an action to a state, and the ones that come from the application of a valid plan to a state S and lead to a state S' reachable from S . We will call such states *lookahead states*, as they are computed by the application of a plan to a node S but are considered in the search tree as direct descendants of S . Nodes created for lookahead states will be called *lookahead nodes*. Plans labeling arcs that lead to lookahead nodes will be called *lookahead plans*. Once a goal state is found, the solution plan is then the concatenation of single actions for arcs leading to classical nodes and lookahead plans for arcs leading to lookahead nodes.

*This work has been supported in part by the IUT de Lens, the CNRS and the Region Nord/Pas-de-Calais under the TACT Programme.

The determination of an heuristic value for each state as performed in the FF planner [Hoffmann and Nebel, 2001] offers a way to compute such lookahead plans. FF creates a planning graph [Blum and Furst, 1997] for each encountered state S , using the relaxed problem obtained by ignoring deletes of actions and using S as initial state. A relaxed plan is then extracted in polynomial time and space from this planning graph. The length in number of actions of the relaxed plan corresponds to the heuristic evaluation of the state for which it is calculated. Generally, the relaxed plan for a state S is not valid for S , as deletes of actions are ignored during its computation. In numerous benchmark domains, we can observe that relaxed plans have a very good quality because they contain a lot of actions that belong to solution plans. We propose a way of computing lookahead plans from these relaxed plans, by trying as most actions as possible from them and keeping the ones that can be collected into a valid plan.

The lookahead algorithm and the modifications to the search algorithm are the following (all details can be found in [Vidal, 2002]). Each time a state S is evaluated, it is entered into the open list. The relaxed plan extracted by the evaluation function is used to compute a lookahead plan P which leads to a state S' reachable from S . If P is more than one action long, S' is evaluated and added to the open list. Let P be a relaxed plan for a state S . A lookahead plan P' is computed as follows: all actions of P are observed in turn. When an action a is applicable to S , it is added to the end of P' and S is updated (by the application of a). When all actions of P have been tried, this process is iterated without the actions that have been applied, until no action can be used.

Completeness and correctness of search algorithms are preserved by this process, because no information is lost: all actions that can be applied to a state are still considered, and because the nodes that are added by lookahead plans are reachable from the states they are connected to. The only modification is the addition of new nodes, corresponding to states that can be reached from the initial state.

2 Using helpful actions: the “optimistic” best-first search algorithm

In classical search algorithms, all actions that can be applied to a node are considered the same way: the states that they lead to are evaluated by an heuristic function and are then or-

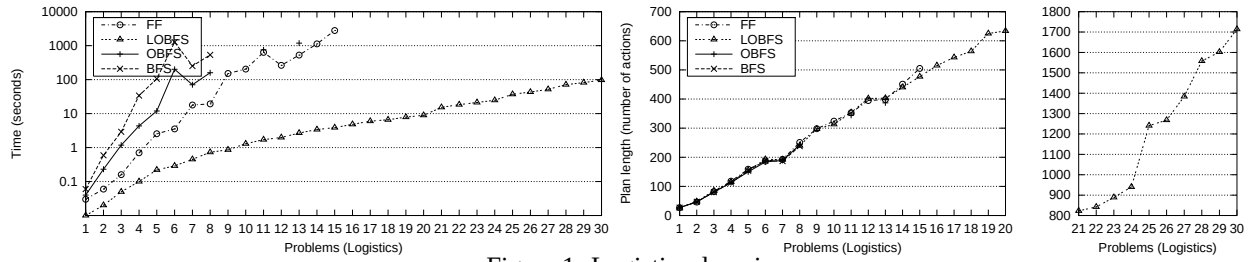


Figure 1: Logistics domain

	Rovers 24			Rovers 30	Driver 15			Driver 21	Driver 30
Init atoms	5920			35791	227			607	2130
Goals	33			127	10			38	163
	FF	OBFS	LOBFS	LOBFS	FF	OBFS	LOBFS	FF	LOBFS
Plan length	130	133	145	560	44	44	54	184	193
Evaluated nodes	3876	2114	9	24	161	273	4	3266	8
Search time	418.32	430.95	1.97	44.35	0.21	0.84	0.02	207.89	0.45
Total time	422.48	437.92	8.87	219.13	0.26	0.97	0.14	209.40	1.96
	Satellite 21			Satellite 30	Logistics 13			Logistics 15	Logistics 30
Init atoms	971			10374	320			364	1140
Goals	124			768	65			75	200
	FF	OBFS	LOBFS	LOBFS	FF	OBFS	LOBFS	FF	LOBFS
Plan length	140	125	151	2058	398	387	403	505	477
Evaluated nodes	22385	20370	5	5	16456	16456	4	45785	4
Search time	188.69	328.42	0.12	33.73	527.21	1181.95	0.29	2792.51	0.42
Total time	188.82	328.70	0.40	94.24	528.10	1184.35	2.68	2793.82	3.88

Table 1: Largest problems solved by OBFS, LOBFS and FF

dered, but there is no notion of preference over the actions themselves. Such a notion of preference during search has been introduced in the FF planner, with the concept of *helpful actions*. Once a relaxed plan is extracted for a state S , the actions of the relaxed plan that are executable in S are considered as *helpful*, while the other actions are forgotten by the local search algorithm of FF. This strategy appeared to be too restrictive, so the set of helpful actions is augmented in FF by all actions executable in S that produce fluents that were considered as subgoals at the first level of the planning graph, during the extraction of the relaxed plan. The main drawback of this strategy, as used in FF, is that it does not preserve completeness: the actions executable in a state S that are not considered as helpful are simply lost. FF switches to a complete best-first algorithm when no solution is found.

We present a way to use such a notion of helpful actions in complete search algorithms, that we call *optimistic search algorithms* because they give a maximum trust to the informations returned by the computation of the heuristic. The principles are the following:

- Several classes of actions are created. In our implementation, we only use two of them: *helpful actions* (the restricted ones), and *rescue actions* that are all the actions that are not helpful.
- When a newly created state S is evaluated, the heuristic function returns the numerical estimation of the state and also the actions executable in S partitioned into their different classes. For each class, one node is created for the state S , that contains the actions of that class.
- Nodes containing helpful actions are always preferred for development over nodes containing rescue actions, whatever their numerical heuristic values are.

No information is lost by this process. The way nodes are developed is simply modified: a state S is developed first with helpful actions, some other nodes are developed, and then

S can potentially be developed with rescue actions. As the union of helpful actions and rescue actions is equal to the set of all the actions that can be applied to S , completeness and correctness are preserved.

3 Experimental evaluation

We compare four planners: FF v2.3, and three different settings of our planning system called YAHSP (which stands for Yet Another Heuristic Search Planner¹) implemented in Objective Caml: BFS (Best First Search: classical WA* search, with $W = 3$). The heuristic is based on the computation of a relaxed plan as in FF), OBFS (Optimistic Best First Search: identical to BFS, with preference of helpful actions over rescue actions), and LOBFS (Lookahead Optimistic Best First Search: identical to OBFS, with lookahead states).

We report here complete results for Logistics domain (see Figure 1) and show in Table 1 some data about the largest problems solved by FF, OBFS and LOBFS, in order to realize the progress accomplished in the size of the problems that can be solved by a STRIPS planner, for five different domains.

Acknowledgments

Thanks a lot to Pierre Régner for his help...

References

- [Blum and Furst, 1997] A. Blum and M. Furst. Fast planning through planning-graphs analysis. *Artificial Intelligence*, 90(1-2):281–300, 1997.
- [Hoffmann and Nebel, 2001] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *JAIR*, 14:253–302, 2001.
- [Vidal, 2002] V. Vidal. A lookahead strategy for heuristic search planning. Technical Report 2002-35-R, IRIT, Université Paul Sabatier, Toulouse, France, 2002.

¹<http://www.cril.univ-artois.fr/~vidal/yahsp.html>

Using Available Memory to Transform Graphplan’s Search

Terry Zimmerman & Subbarao Kambhampati

Department of Computer Science and Engineering

Arizona State University, Tempe AZ 85287-5406

{zim,rao}@asu.edu

Abstract

We present a major variant of the Graphplan algorithm that employs available memory to transform the depth-first nature of Graphplan’s search into an iterative state space view in which heuristics can be used to traverse the search space. When the planner, PEGG, is set to conduct exhaustive search, it produces guaranteed optimal parallel plans 2 to 90 times faster than a version of Graphplan enhanced with CSP speedup methods. By heuristically pruning this search space PEGG produces plans comparable to Graphplan’s in makespan, at speeds approaching state-of-the-art heuristic serial planners.

1 Motivation and Approach

Despite the recent dominance of heuristic state-search planners over Graphplan-style planners, the Graphplan approach [Blum and Furst 1997] is still one of the most effective ways to generate so-called “optimal parallel plans”. While state-space planners are drowned by the exponential branching factors of the search space of parallel plans, Graphplan excels due to the way it combines an IDA* style iterative search [Bonet and Geffner, 1999] with a highly efficient CSP-based, incremental generation of valid action subsets. We present here a system called PEGG, that addresses weaknesses in Graphplan’s approach by employing available memory to: 1) reduce the redundant search Graphplan conducts in consecutive iterations, and 2) more importantly, to transform Graphplan’s IDA* search into iterative expansion of a select set of states that can be traversed any order.

A shortfall of the IDA*'s approach to search is the fact that it regenerates many of the same nodes in each of its iterations. This can be traced to using too little memory in many cases; the only information carried over from one iteration to the next is the upper bound on the f -value. Given that consecutive iterations of search overlap significantly, we investigated using additional memory to store a trace of the explored search tree to avoid repeated re-generation of search nodes. With a representation of the explored search space, we can transform the way this space is extended during the next iteration. In particular, we can (a) expand search trace nodes in the order of their heuristic merit and (b) we may also consider iteratively expanding a select set of states. This strategy is too costly for normal IDA* search, but Graphplan's type of IDA* search is particularly well-suited to

these changes as the k th level planning graph provides a compact way of representing the search space traversed by the corresponding IDA* search in its k th iteration. The state space view provided by the search trace allows us to transform Graphplan’s search from its depth-first default to a more informed traversal of the space.

2 Design and Experiments

As would be expected for IDA* search there is great similarity (redundancy) in the search space for successive search episodes as the plan graph is extended. In fact, the search conducted at any level $k+1$ of the graph is essentially a replay of the search conducted at the previous level k with certain well-defined extensions. Specifically, *every* set of subgoals reached in the backward search of episode n , starting at level k , will be generated again by Graphplan in episode $n+1$ starting at level $k+1$.

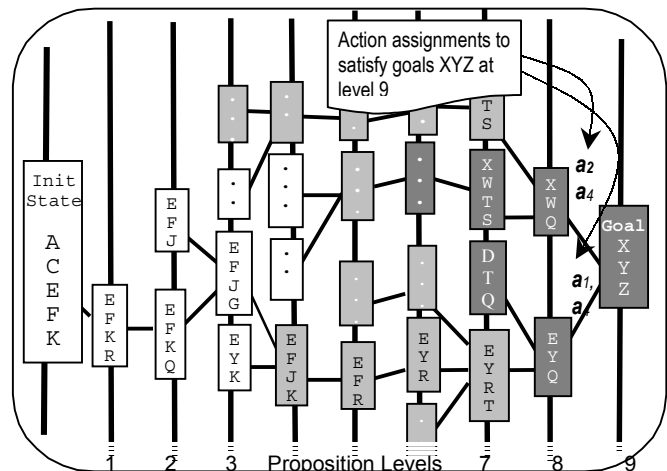


Figure 1. State space view of Graphplan’s search space:
3 consecutive search episodes leading to a solution

Figure 1 depicts the state space tree structure corresponding to Graphplan’s search over three consecutive search iterations on a hypothetical problem. The dark shaded states are first produced during Graphplan’s attempt to satisfy the XYZ goal at level 7. They are generated again in the next episode, each at one level higher, along with some new states (shown in lighter shade). Finally, in the third episode Graphplan regenerates the dark and lighter shaded states, attempting to satisfy XYZ at level 9, and finds a solution.

EGBG [Zimmerman, Kambhampati, 1999] used memory to aggressively record the experience in each search episode, essentially avoiding all redundant effort. That approach confronted memory constraints on larger problems, but it suggests a more powerful use for a pared-down search trace: exploiting the snapshot view of the entire search space of a Graphplan iteration to focus on the most promising areas. This transformation frees us from the depth-first nature of Graphplan's search, permitting movement about the search space to focus on its most promising sections first -or even exclusively.

A summary of PEGG (for details; [Zimmerman and Kambhampati, 2003]) relies on these definitions: **Search segment**: a state generated during Graphplan's regression search from the goal state, indexed to a specific plan graph level. It holds the state's goal list, a pointer to the parent search segment, and the actions assigned in satisfying the parent's goals. **Search trace (ST)**: the linked set of search segments (states) representing the search space visited in a Graphplan backward search episode. The sets of states in each of the three shadings of Figure 1 can be seen as the ST after each of the three episodes. **Transposition**: The extant trace of search segments (states) after search episode n is *transposed* up one planning graph level for episode $n+1$ as follows: For each ST search segment associated with graph level j associate it with level $j+1$ for episode $n+1$. **Visiting** a search segment: the goals of segment S_p are memo checked at their associated level and if valid, PEGG initiates Graphplan's CSP-style search to satisfy them. The process is enhanced by a cadre of efficiency techniques such as a bi-level plan graph, domain preprocessing, explanation based learning (EBL), dependency directed backtracking (DDB), and goal & action ordering. Whenever the goals of S_p are validly assigned, a child segment is created containing S_p 's goals regressed over the assigned actions, and linked to S_p (thus extending the ST).

The PEGG algorithm: the graph is built until the problem goals appear non-mutex, and the first regression search episode is conducted ala Graphplan fashion. During this search, the initial trace is constructed, concisely capturing all 'states' generated during the search process. If no solution is found, the ST directs the search process for future iterations. This search is 2-phased: select a promising ST state, then Graphplan's depth-first, CSP-type search on the state's subgoals is conducted. Another ST search segment is heuristically selected if search fails. Since the ST provides a state space view, PEGG can use 'distance based' heuristics (c.f. HSP-R [Bonet and Geffner, 1999] and AltAlt [Nguyen and Kambhampati, 2000]). For results reported here, the 'adjusted sum' heuristic from the latter is used.

Table 1 compares PEGG against standard Graphplan and a highly enhanced version (GP-e), which has been augmented with the efficiency methods mentioned above. Two PEGG modes of operation are reported; 1) *so-PEGG*: make-span optimal, ST search segments ordered according to a state space heuristic, *all* segments visited, 2) *PEGG*: Ordering the ST search segments as for 1, beam

Problem	Graphplan		so-PEGG cpu sec (steps/acts)	PEGG cpu sec (steps/acts)	Speedup (PEGG vs. GP-e)
	cpu sec (steps/acts) Std.	GP-e			
bw-large-B	194.8	13.4 (18/ 18)	12.2	3.1 (18/ 18)	4.3x
bw-large-D	~	~ (36/ 36)	~	340 (38 / 38)	> 5x
att-log-b	~	~	~	120 (11/ 79)	>15x
gripper-15	~	~ (36/ 45)	47.5	16.7 (36/ 45)	>107x
gripper-20	~	~ (40/ 59)	~	44.8 (40/ 59)	> 40x
tower-9	~	~ (511/ 511)	118	23.6 (511/ 511)	> 76x
TSP-12	~	~ (12/ 12)	7.2	6.5 (12/ 12)	>277x
<i>AIPS '98, '00, '02 Competition Problems</i>					
gripper-x-4	~	190 (19/ 29)	73.9	30.9 (19/ 29)	6.1x
gripper-x-5	~	~	512	110 (23/ 35)	> 16x
log-y-4	~	470 (11/ 60)	366	330 (11/ 58)	1.4x
blocks-10-1	~	95.4 (34/ 34)	18.7	11.0 (34/ 34)	8.7x
blocks-16-2	~	~ (54/ 54)	~	58.7 (54/ 54)	> 31 x
logistics-10-0	~	30.0 (15/ 56)	21	8.9 (15/ 55)	3.4x
logistics-12-1	~	~	1101 (15/75)	119 (15/ 75)	>15.1x
freecell-2-1	~	98.0 (6/ 10)	80.1	62.9	1.5x
depot-6512	239	5.1	5.0	2.1	2.4x
depot-1212	~	~ (22/ 55)	~	127 (22/ 56)	> 14x
driverlog-2-3-6	~	27.5 (7/ 20)	1.9	1.9 (7/ 20)	14.5x
driverlog-4-4-8	~	~	~	589 (10/ 24)	> 3x
ztravel-3-7a	~	~	1434 (10/23)	222 (10/ 21)	> 8x
ztravel-3-8a	~	972 (7/ 25)	11.2	2.3 (7/ 25)	423x

Table 1. PEGG vs. Graphplan and enhanced Graphplan (GP-e)
GP-e: enhanced Graphplan (see text) *so-PEGG*: step-optimal, search via the ST. *PEGG*: beam search on best 20% of search segments in ST
~ indicates failure in 30 min limit, cpu time
Parentheses next to cpu time give # of steps/ # of actions in solution.
Allegro Lisp, runtimes (excl. gc time) on Pentium 900 mhz, 384 MB RAM

search on only the heuristically 'best' fraction. The first approach maintains Graphplan's guarantee of step optimality while the latter sacrifices the guarantee of optimality in favor of pruning search in *all* search episodes and bounds the size of the search trace that is maintained in memory. Empirically we find that optimal make-span plans are generally found by PEGG *regardless*, (one exception shown in bold) and speedups as high as two orders of magnitude over enhanced Graphplan are achieved.

References

- Blum A. and Furst M.1997. Fast planning through planning graph analysis. *Artificial Intelligence* 90(1-2).1997.
- Bonet, B. and Geffner, H. 1999. Planning as heuristic search: New results. In *Proceedings of ECP-99*, 1999.
- Nguyen, X. and Kambhampati, S. 2000. Extracting effective and admissible state space heuristics from the planning graph. In *Proceedings of AAAI-2000*.
- Zimmerman, T. and Kambhampati, S. 1999. Exploiting Symmetry in the Planning-graph via Explanation-Guided Search. In *Proceedings of AAAI-99*, 1999.
- Zimmerman, T. and Kambhampati, S. 2003. Using memory to transform search on the planning graph. ASU Technical Report (available at <http://rakaposhi.eas.asu.edu/pegg-tr.pdf>)

POSTER PAPERS

SEARCH

Lookahead Pathologies for Single Agent Search

Vadim Bulitko and Lihong Li and Russ Greiner and Ilya Levner

Department of Computing Science

University of Alberta

Edmonton, Alberta T6G 2E8

CANADA

{bulitko|lihong|greiner|ilya}@cs.ualberta.ca

Abstract

Admissible and consistent heuristic functions are usually preferred in single-agent heuristic search as they guarantee optimal solutions with complete search methods such as A* and IDA*. Larger problems, however, frequently make a complete search intractable due to space and/or time limitations. In particular, a path-planning agent in a real-time strategy game may need to take an action before its complete search has the time to finish. In such cases, incomplete search techniques (such as RTA*, SRTA*, RTDP, DTA*) can be used. Such algorithms conduct a limited ply lookahead and then evaluate the states envisioned using a heuristic function. The action selected on the basis of such evaluations can be suboptimal due to the incompleteness of search and inaccuracies in the heuristic. It is usually believed that deeper lookahead increases the chances of taking the optimal action. In this paper, we demonstrate that this is not necessarily the case, even when admissible and consistent heuristic functions are used.

1 Lookahead Pathologies in Real-time Single-agent Search

Complete search methods such as A* [Hart *et al.*, 1968] and IDA* [Korf, 1985] produce optimal solutions when based on an admissible and monotonic heuristic function. The primary drawbacks are the exponential running time and the necessity to wait until the search completes before the first action can be taken [Korf, 1990]. This limits the applicability of complete search in practice as the deliberation time per action can be severely limited [Higgins, 2002], the domain model can be expensive [Bulitko and Wilkins, 2002], the goal states can be difficult to recognize [Levner *et al.*, 2002]. Consequently, despite numerous advances in improving heuristic functions [Korf and Taylor, 1996; Culberson and Schaeffer, 1994; Reinefeld, 1993; Korf, 1997], incomplete real-time/on-line search methods remain the practical choice for complex real-life problems.

Various incomplete search methods have been proposed including: RTA* [Korf, 1990], RTDP [Barto *et al.*, 1995], SRTA*, and DTA* [Russell and Wefald, 1991]. Such algorithms base their decisions on heuristic information collected

from a partial tree expansion (lookahead) prior to reaching the goal state. Since the heuristic function is generally inaccurate and the search is incomplete, suboptimal solutions can be produced even with admissible and consistent heuristics.

It is widely believed that looking ahead deeper improves the chances of taking the right action. Consequently, a considerable amount of effort has been put into increasing the lookahead depth by using selective search (search extensions) and hardware and software optimizations.

In this paper we demonstrate that looking ahead deeper can actually *decrease* the chances of taking the optimal action even when admissible and consistent heuristic functions are used.

2 Related Past Research & Our Novel Contributions

Lookahead pathologies within the mini-max search in two-player games have been investigated extensively in the past. In [Nau, 1982; 1983; Beal, 1980; 1982; 1983; Bratko and Gams, 1982; Pearl, 1983], the primary cause of pathologies was deemed to be the independence of heuristic values of the leaf nodes. Such games were called *non-incremental*. Large branching factors were also considered contributing to a pathology. Later, [Michon, 1983] added *non-inertness* (i.e., the constant branching factor) to the “black list” of properties that can cause a pathology.

The efforts presented in this paper differ from the past work in several key areas: (i) we demonstrate lookahead pathologies in *single-agent* heuristic search; (ii) pathologies are shown to take place even when the true and heuristic values of sibling states are correlated (i.e., in *incremental domains*); (iii) unlike research in [Schrüfer, 1986; Beal, 1980] that dealt with two-valued (win/loss) heuristics, we consider *real-valued heuristic functions*; (iv) we show pathologies with the *smallest non-trivial branching factor* (two); (v) pathologies are demonstrated for *admissible and consistent* heuristic functions.

3 Framework

In this study we consider a single-agent heuristic search in a discrete state domain with a finite number of deterministic actions. The states (set S) and actions (set A) form a directed graph with certain specified vertices representing the goal states. The edges (actions) are weighed with action costs:

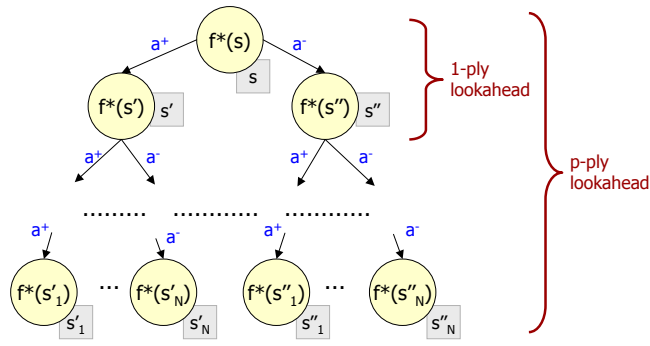


Figure 1: Illustration of p -ply binary lookahead tree. Throughout the paper we display the true f^* in the circles, the states or heuristic f values are shown next to the circles, and the edges are labeled with actions. Here a^+ and a^- are the optimal and suboptimal actions respectively.

$c : A \rightarrow R$. The agent is provided with a perfect domain model: $\delta : S \times A \rightarrow S$.

We define the true distance-to-goal function $h^*(s)$ as the sum of action costs along the shortest path from state s to the closest goal state. Generally speaking, the agent uses an approximation h to the unavailable h^* . The approximation is typically inaccurate inasmuch as: $\exists s \in S [h^*(s) \neq h(s)]$.

For a fixed starting state s , function $g(s')$ is defined as the sum of action costs along the shortest path from s to s' . Finally, the sum of h or h^* and g is typically denoted by f or f^* . It is easy to see that f^* remains constant along any optimal path from a fixed state s to the closest goal. Also note that, for any state s' , action a_1 is inferior to action a_2 iff $f^*(\delta(s', a_1)) > f^*(\delta(s', a_2))$.

Located in state s , the agent can use its perfect model δ to predict the states it will get to upon taking various sequences of actions. A binary lookahead tree is illustrated in Figure 1. For the sake of clarity and simplicity we will be using small binary lookahead trees to illustrate the discussion throughout the paper.

The lookahead policy $\pi(s, p)$ operates as follows: (i) start in the current state s ; (ii) construct the lookahead search tree of p plies deep by envisioning terminal states of all action sequences of p actions (whenever possible); (iii) evaluate the leaf nodes of the lookahead tree using the f function and select the minimum-valued state; (iv) output the single action leading to selected leaf state (resolve ties randomly).

Depending on the lookahead tree, random tie resolution, and the approximate heuristic f , the action $a = \pi(s, p)$ output by $\pi(s, p)$ can be suboptimal: $\exists a^* \neq a [f^*(\delta(s, a^*)) < f^*(\delta(s, a))]$. The probability of such an error for state $s \in S$ is denoted by $Err(s, p)$. Additionally, we will consider the expected value of $Err(s, p)$ over the states s . Such state-independent quantity will be referred to as $Err(p)$.

The combination of domain and heuristic function h (f) is called *pathological* iff $\exists p_1 < p_2 [Err(s, p_1) < Err(s, p_2)]$. The corresponding state-independent version is: $\exists p_1 < p_2 [Err(p_1) < Err(p_2)]$. The intuitive meaning is quite transparent: lookahead search is pathological iff looking deeper ahead is expected to *increase* the chances of taking

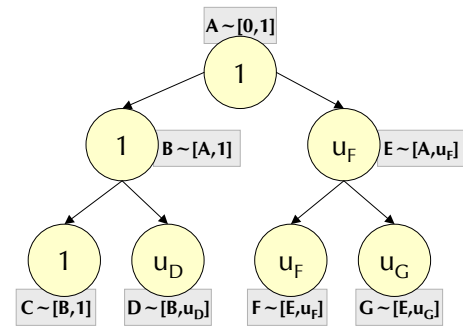


Figure 2: The 7-node 2-ply lookahead tree used in the analysis of pathologies with admissible and consistent heuristics.

a suboptimal action.

We adopt the following standard terminology. Function h is called *admissible* iff $\forall s \in S [h(s) \leq h^*(s)]$. It is called *consistent* iff $\forall a \in A, s \in S [h(s) \leq c(a) + h(\delta(s, a))]$. It can be shown that consistency of h is equivalent to non-decreasing monotonicity of $f = g + h$: $\forall a \in A, s \in S [f(s) \leq f(\delta(s, a))]$. In the following we will assume that the true h^* satisfies the mini-min relation: $\forall s [h^*(s) = \min_{a \in A} (c(a) + h^*(\delta(s, a)))]$ and h^* is, therefore, consistent and, trivially, admissible.

4 Admissible & Consistent Heuristics

Complete searches (e.g., A* and IDA*) produce optimal solutions with admissible heuristics. Thus, much effort has gone into derivation of admissible heuristics either manually or automatically (e.g., [Korf and Taylor, 1996; Prieditis, 1993]). Consistency of h leads to non-decreasing monotonicity of f and often speeds up complete searches conducted by A*/IDA*. Incomplete searches (e.g., RTA*) can produce suboptimal solutions even with consistent and admissible heuristics. It is widely believed, however, that deeper lookahead reduces chances of such errors. Remarkably, this is not necessarily the case.

Consider the binary lookahead tree in Figure 2. Each node is shown with its true value $f^*(s)$ inside the circle. For each state s , heuristic $f(s)$ is drawn uniformly from $[f(s_{\text{parent}}), f^*(s)]$ (indicated with the ' $\sim$ ' in the boxes next to the circles). The value (A) of the root state is drawn from $[0, f^*(s_{\text{root}})]$. This makes the f function admissible and monotonically non-decreasing.

We will refer to the drawn values of f as A, B, C, D, E, F, G (illustrated in the figure). Assume that $u_D > 1, u_G > u_F > 1$. By definition, the lookahead will be pathological iff $Err(s_{\text{root}}, 2) > Err(s_{\text{root}}, 1)$. The probability of taking the suboptimal action (i.e., going to the right child of the root node) with the lookahead of one is:

$$\begin{aligned} Err(s_{\text{root}}, 1) &= Pr(E < B) \\ &= \int_0^1 \int_A^1 \int_E^1 \frac{1}{(1-A)(u_F-A)} dB dE dA \\ &= \frac{1}{2} \left[1 + (u_F - 1) \ln \left(1 - \frac{1}{u_F} \right) \right]. \end{aligned}$$

Similarly we derive an expression for the lookahead of two: $Err(s_{\text{root}}, 2)$ which can not be reproduced in this short pa-

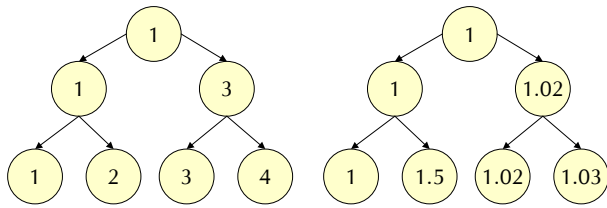


Figure 3: Admissible and consistent heuristics: non-pathological and pathological lookahead trees.

per*. By analyzing the difference of the two expressions we arrive at a criterion of pathology.

A concrete illustration of this phenomenon can be found in Figure 3. The tree on the left is non-pathological as: $Err(s_{root}, 1) \approx 0.0945 > 0.0226 \approx Err(s_{root}, 2)$. The tree on the right, however, is indeed pathological. Averaged over 10 million trials: $Err(s_{root}, 1) \approx 0.4607 < 0.4861 \approx Err(s_{root}, 2)$.

5 Future Work & Conclusions

In practice, lookahead pathologies with real-time (single-agent or two-player) search are *infrequently* observed even when inadmissible heuristics are used (cf., [Thompson, 1982] for the game of chess). Consequently, future work directions include: (i) identification of the properties of practically used heuristic functions (e.g., Manhattan distance in the 8 puzzle) that are responsible for the *lack* of pathologies, (ii) extension of the current formal analysis to lookahead trees with a variable branching factor and a deeper lookahead, (iii) methods for on-line and off-line pathology detection and correction and (iv) connections between the probability of taking sub-optimal actions and a decrease in the cumulative expected reward.

Acknowledgements

Rene Levia, Dan Lizotte, Omid Madani, Xin Li, and Yiping Zhou have contributed in various ways. We are grateful for the funding from the University of Alberta, NSERC, and the Alberta Ingenuity Centre for Machine Learning.

References

- [Barto *et al.*, 1995] A. G. Barto, S. J. Bradtko, and S. P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1):81–138, 1995.
- [Beal, 1980] D.F. Beal. An analysis of minimax. In M.R.B. Clarke, editor, *Advances in Computer Chess*, volume 2, pages 103–109. Edinburgh University Press, 1980.
- [Beal, 1982] D.F. Beal. Benefits of minimax search. In M.R.B. Clarke, editor, *Advances in Computer Chess*, volume 3, pages 17–24. Oxford : Pergamon, 1982.
- [Beal, 1983] D.F. Beal. Recent progress in understanding minimax search. In *Proceedings of ACM Annual Conference*, pages 164–169, New York, 1983. Association for Computing Machinery.

*The interested reader is referred to the project website at <http://www.cs.ualberta.ca/ircl>.

- [Bratko and Gams, 1982] I. Bratko and M. Gams. Error analysis of the minimax principle. In M.R.B. Clarke, editor, *Advances in Computer Chess*, volume 3, pages 1–15. Oxford : Pergamon, 1982.
- [Bulitko and Wilkins, 2002] V. Bulitko and D.C. Wilkins. Real-time decision making for shipboard damage control. In *Proceedings of the AAAI/KDD/UAI-2002 Joint Workshop on Real-Time Decision Support and Diagnosis Systems*, pages 37 – 46, Edmonton, Alberta, 2002. AAAI Press.
- [Culberson and Schaeffer, 1994] J. Culberson and J. Schaeffer. Efficiently searching the 15-puzzle. Technical Report TR 94-08, Department of Computing Science, University of Alberta, 1994.
- [Hart *et al.*, 1968] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. on Systems Science and Cybernetics*, 4(2):100–107, 1968.
- [Higgins, 2002] Dan Higgins. Pathfinding design architecture. In Steve Rabin, editor, *AI Game Programming Wisdom*, pages 122–132. Charles River Media, 2002.
- [Korf and Taylor, 1996] Richard E. Korf and Larry A. Taylor. Finding optimal solutions to the twenty-four puzzle. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 1202–1207, Portland, Oregon, USA, 1996. AAAI Press / The MIT Press.
- [Korf, 1985] R. Korf. Depth-first iterative deepening : An optimal admissible tree search. *Artificial Intelligence*, 27(1), 1985.
- [Korf, 1990] Richard E. Korf. Real-time heuristic search. *Artificial Intelligence*, 42(2-3):189–211, 1990.
- [Korf, 1997] R. Korf. Finding optimal solutions to rubik’s cube using pattern databases. In *Proceedings of the Workshop on Computer Games (W31) at IJCAI-97*, pages 21–26, Nagoya, Japan, 1997.
- [Levner *et al.*, 2002] I. Levner, V. Bulitko, O. Madani, and R. Greiner. Performance of lookahead control policies in the face of abstractions and approximations. In S. Koenig and R. Holte, editors, *Lecture Notes in Artificial Intelligence (LNAI)*, volume 2371, pages 299–308. Springer-Verlag, Berlin, Heidelberg, 2002.
- [Michon, 1983] G. Michon. *Recursive Random Games: A Probabilistic Model for Perfect Information Games*. PhD thesis, University of California at Los Angeles, 1983.
- [Nau, 1982] D.S. Nau. An investigation of the causes of pathology in games. *Artificial Intelligence*, 19:257–278, 1982.
- [Nau, 1983] D.S. Nau. Pathology on game trees revisited, and an alternative to minimaxing. *Artificial Intelligence*, 21(1,2):221–244, 1983.
- [Pearl, 1983] J. Pearl. On the Nature of Pathology in Game Searching. *Artificial Intelligence*, 20:427–453, 1983.
- [Prieditis, 1993] A. E. Prieditis. Machine discovery of effective admissible heuristics. *Machine Learning*, 12:117–141, 1993.
- [Reinefeld, 1993] Alexander Reinefeld. Complete solution of the eight-puzzle and the benefit of node ordering in IDA. In *IJCAI*, pages 248–253, 1993.
- [Russell and Wefald, 1991] Stuart Jonathan Russell and Eric H. Wefald. *Do the right thing: Studies in limited rationality*. MIT Press, 1991.
- [Schrüfer, 1986] G. Schrüfer. Presence and absence of pathology on game trees. In D.F. Beal, editor, *Advances in Computer Chess*, volume 4, pages 101–112. Oxford: Pergamon, 1986.
- [Thompson, 1982] K. Thompson. Computer chess strength. In M.R.B. Clarke, editor, *Advances in Computer Chess*, volume 3, pages 55–56. Oxford : Pergamon, 1982.

Real-Time Strategy Games: A New AI Research Challenge

Michael Buro

Department of Computing Science, University of Alberta, Edmonton, AB, T6J 2E8, Canada
email: mburo@cs.ualberta.ca

Abstract

This poster motivates AI research in the area of real-time strategy (RTS) games and describes the current status of a project whose goals are to implement an RTS game programming environment and to build AIs that eventually can outperform human experts in this challenging and popular domain.

1 Real-Time Strategy Games

Commercial computer games are a growing part of the entertainment industry and simulations are a critical aspect of modern military training. The two fields have much in common, cross-fertilize, and are driving real-time AI research [Herz and Macedonia, 2002]. With the advent of fast personal computers, simulation-based games have become very popular. Today, these games constitute a multi-billion dollar enterprise. Examples are sports games and real-time strategy games. The common elements of simulation games are severe time constraints and a strong demand of real-time AI which must be capable of solving real-world decision tasks quickly and satisfactorily. Popular simulation games are therefore ideal test applications for real-time AI research.

Real-Time-Strategy (RTS) games – such as the million-sellers *Starcraft* by Blizzard Entertainment and *Age of Empires* by Ensemble Studios – can be viewed as simplified military simulations. Several players struggle over resources scattered over a 2D terrain by setting up an economy, building armies, and guiding them into battle in real-time. RTS games offer a large variety of fundamental AI research problems, unlike other game genres studied by the AI community so far:

- **Resource management.** Players start off by gathering local resources to build up defenses and attack forces, to upgrade weaponry, and to climb up the technology tree. Proper resource management is a vital part of any successful strategy.
- **Decision making under uncertainty.** Initially, players are not aware of the enemies' base locations and intentions. They have to gather intelligence by sending out scouts. If no information is available yet, the players must form plausible hypotheses and act accordingly.
- **Spatial and temporal reasoning.** Static and dynamic terrain analysis as well as understanding temporal relations of actions is of utmost importance in RTS games – and yet, current game AIs largely ignore these issues and fall victim to simple common-sense reasoning [Forbus *et al.*, 2002].
- **Collaboration.** In RTS games groups of players can join forces and intelligence. How to coordinate actions effectively by communication among the parties is a challenging research problem.

- **Opponent modeling, Learning.** One of the biggest shortcomings of most (RTS) game AI systems is their inability to learn from experience. Human players only need a couple of games to spot opponents' weaknesses and to exploit them in upcoming games. Current machine learning approaches in this area are inadequate.

- **Adversarial real-time planning.** In fine-grained simulations, agents cannot afford to think in terms of micro actions. Instead, abstractions have to be found which allow a machine to conduct forward searches in a manageable abstract space and to translate found solutions back. Because the environment is also dynamic, hostile, and smart – adversarial real-time planning approaches need to be investigated.

Playing RTS games is challenging. Even more challenging is the creation of autonomous real-time systems capable of outperforming human experts in this domain. Because search space abstraction, real-time planning, and temporal and spatial reasoning are central to many other problems, the scope of applications seems endless. One example is high-performance combat simulators which are in large demand for training military personnel today and will become the core of automated battlefield decision-support systems of tomorrow. [von der Lippe *et al.*, 1999] predicts that 20% of the US armed forces will be robotic by 2015.

2 An RTS Game Programming Environment

The lack of AI interfaces even in upcoming RTS game titles makes it hard to conduct real-time AI research in this area and to compare the strength of the resulting AI systems with that of human experts. In order to solve this problem we launched an open source RTS game programming project [Buro, 2002] with the following goals:

- **Building a hack-free server-client RTS game system.** At the core of the system is a simulator to which players connect via UNIX sockets (Fig. 1). The unique system features include: **server-side simulation** – which only sends visible information to clients thereby rendering common map-revealing client hacks useless – and an **open message protocol** that allows AI researchers and players to connect *whatever* client software they like.
- **Sparkling competition among players and researchers.** Popular games in which human players still have the upper hand are ideal test-domains for AI research. Unlike the confined GUIs of commercial RTS games, our open design allows the construction of hybrid AI systems in which the human general is aided by AI modules of growing capabilities. Competitive game playing on an open Internet RTS game server is therefore likely to improve AI performance and ergonomic GUI design.
- **Applying planning and machine learning techniques to RTS games.** Classic game AI methods – such as alpha-

beta search – cannot be applied directly in this domain due to the large number of game objects, imperfect information, and real-time demands. Search space abstraction, hierarchical planning, and machine learning are approaches to overcome these problems.

Our initial software release implemented a basic RTS programming infrastructure. Recently, we added **terrain features** (water, ground, plateaus, and ramps), **view obstruction** by terrain, and a **split-view GUI** which allows to watch different parts of the playing field simultaneously. In order to maintain the high simulation speed, terrain and vision are based on tiles (Fig. 2). Soon, the complete source code of our RTS programming environment will be released. It is being used in two thesis projects dealing with machine learning of low-level unit behavior and abstraction, planning, and heuristic search in RTS games. We hope to report first results in the RTS game AI domain later this year and to spark enough interest to be able to organize man-machine and machine-machine RTS game tournaments in the near future. For this it will be necessary to define standard game setups including unit properties (size, speed, sight and attack ranges,...), technology tree, and resources. In contrast to commercial products, the focus here will be on simplicity to ease the initial AI development.

3 Related Work

The growing literature on AI in commercial games surely indicates the demand of smarter game AI. Many articles on planning are relevant to this project. Due to space limitation we only briefly mention J. Laird's Soar architecture and its application to first-person shooter games [Laird, 2001] and M. Atkin's work on the GRASP system that is applied to a capture-the-flag game [Atkin, 1998].

Computer soccer pursues similar goals and has become quite popular [Stone, 2002]. This domain can be regarded as a simplified RTS game with only a few objects, no economy, unremarkable terrain features, and more or less complete in-

formation. Another big difference is that in computer soccer a small number of agents are required to compute their actions autonomously whereas in RTS games a large number of objects have to be orchestrated.

The other body of literature relevant to this project is on military analyses and applications. Research in this area spans from mathematical combat models [Ilachinski, 1996] over computer generated forces – which are used in simulation and training – to decision-support systems that aid commanders and troops on the battle-field or even control entire weapon systems autonomously. Our project tries to bring both research communities together.

References

- [Atkin, 1998] M.S. Atkin. AFS and HAC: Domain-general agent simulation and control. In *AAAI Workshop on Software Tools for Developing Agents*, 1998.
- [Buro, 2002] M. Buro. ORTS: A hack-free RTS game environment. In *Proceedings of the Third International Conference on Computers and Games*, pages 156–161, 2002.
- [Forbus et al., 2002] K.D. Forbus, J.V. Mahoney, and K. Dill. How qualitative spatial reasoning can improve strategy game AIs. *IEEE Intelligent Systems*, 17(4):25–30, July 2002.
- [Herz and Macedonia, 2002] J.C. Herz and M.R. Macedonia. Computer games and the military: Two views. *Defense Horizons, Center for Technology and National Security Policy, National Defense University*, 11, April 2002.
- [Ilachinski, 1996] A. Ilachinski. Land warfare and complexity. CRM 96-68, Center for Naval Analysis Research, 1996.
- [Laird, 2001] J. Laird. Using a computer game to develop advanced AI. *Computer*, 34(7):70–75, July 2001.
- [Stone, 2002] P. Stone. Multiagent competitions and research: Lessons from RoboCup and TAC. *RoboCup 2002 International Symposium*, June 2002.
- [von der Lippe et al., 1999] S. von der Lippe et al. A robotic army: The future is CGF. In *10th Conference on Computer Generated Forces and Behavioral Representation*, Florida, USA, 1999.

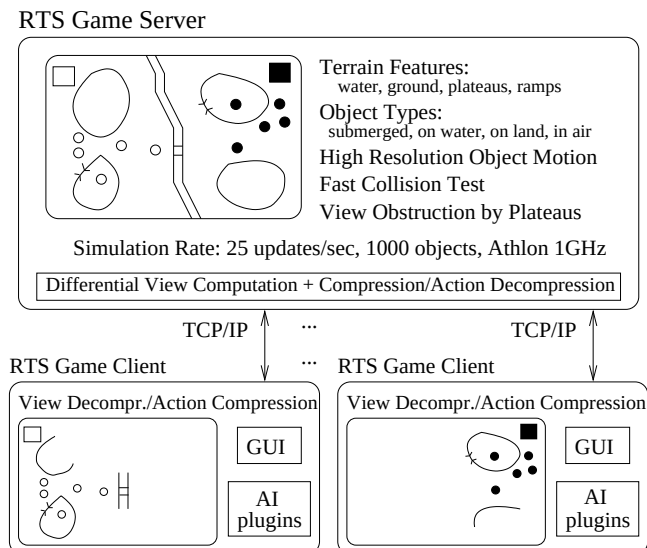


Figure 1: Server-client RTS game architecture. Clients connect to a central server which sends player views, receives actions for all objects, and updates the state of the world.

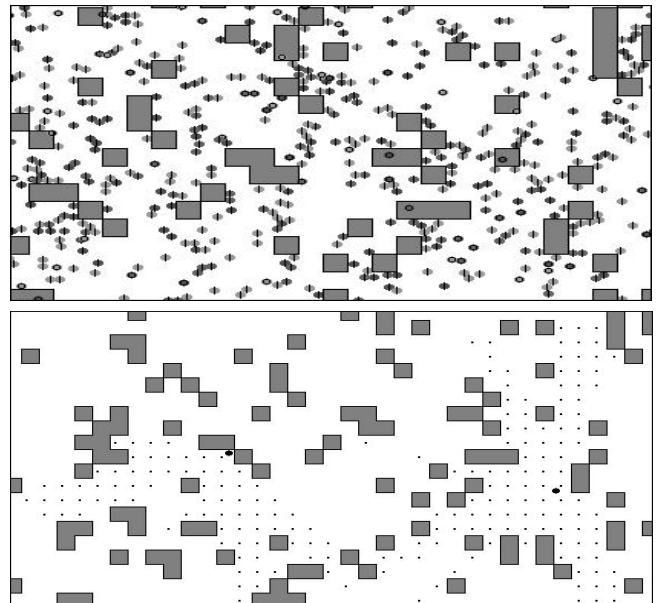


Figure 2: a) Motion/collision stress test: hundreds of moving ground and air units on hilly terrain. All units are visible to all players. b) Tile-based vision: views are obstructed by hills.

Multiple Agents Moving Target Search

Mark Goldenberg, Alexander Kovarsky, Xiaomeng Wu, Jonathan Schaeffer

Department of Computing Science, University of Alberta,
Edmonton, Alberta, Canada T6G 2E8
{goldenbe,kovarsky,xiaomeng,jonathan}@cs.ualberta.ca

Abstract

Traditional single-agent search algorithms usually make simplifying assumptions (single search agent, stationary target, complete knowledge of the state, and sufficient time). There are algorithms for relaxing one or two of these constraints; in this paper we want to relax all four. The application domain is to have multiple search agents cooperate to pursue and capture a moving target. Agents are allowed to communicate with each other. For solving Multiple Agents Moving Target (MAMT) applications, we present a framework for specifying a family of suitable search algorithms. This paper investigates several effective approaches for solving problem instances in this domain.

1 Introduction

In the 2002 Steven Spielberg movie *Minority Report*, John Anderton (played by Tom Cruise) is on the run. In one sequence, Anderton is hiding in a building, and his pursuers unleash a team of mini-robots to flush him out. The robot team separates, each covering a different part of the building. Anderton, realizing the danger, stops fleeing and comes up with a unique solution – he submerges himself in a bathtub of water so as to avoid the robotic detectors. Sadly, he can only hold his breath for so long, before he has to emerge and is found by the robots.

The classic algorithms (such as A^*) are effective for solving search problems that satisfy the properties: one search agent, the agent has perfect information about the environment, the environment and goal state do not change, and enough time is given to make an optimal decision. Relaxing even one of the assumptions gives rise to new algorithms (e.g., moving target search [Ishida and Korf, 1995]), real-time search [Korf, 1990], and D^* [Stentz, 1995]). Many real-world problems do not satisfy all of these properties. In this paper we use an application domain that breaks all of them.

Consider the task of multiple agents having to pursue and capture a moving target; for example a squad of policemen chasing a villain. We want to design a test environment that is as realistic as possible. We assume a grid with obstacles. Agents can only “see” what is directly visible to them. Agents

are allowed to communicate with any agent that they can see. As the target flees, it may become obscured from sight.

The agent’s knowledge of the locations of the target and other agents may be fuzzy (since some of them may be hidden from sight). Given whatever knowledge of the target and other agents is available, an agent must decide how to pursue the target. The target’s possible locations provide information of where to search; the other agents’ possible locations provide information that can be used to coordinate the search effort. The challenge is to have the agents act autonomously to catch the target as quickly as possible.

This paper makes the following contributions: MAMT – a challenging application domain for exploring issues related to Multiple Agents pursuing a Moving Target, a framework for expressing real-time search algorithms for the MAMT domain, and several solutions that allow an agent to act autonomously (using information about the possible positions of the target and other agents in the decision-making process).

More details are available at www.cs.ualberta.ca/~jonathan/Papers/ai.2003.html.

2 Literature

There are a family of A^* algorithms that relax solution optimality by requiring the agent to make the best decision possible given limited search resources (e.g., time) [Korf, 1990]. The *Minimin Lookahead Search* algorithm uses a fixed-depth search (d moves), keeping track of the moves that lead to the (heuristically) best d -move outcome. *Real-Time A^** (RTA^*) is an A^* variant that uses the results produced by the minimin lookahead search as heuristic values in order to guide the search towards achieving the goal [Korf, 1990]. *Moving Target Search* (MTS) is an $LRTA^*$ variant that allows a moving target [Ishida and Korf, 1995]. An assumption in most MTS papers is that the target moves slower than the agent. Without this requirement, the target can stay ahead of the agent and possibly elude capture. There are many real-time search variants (e.g., LPA* [Koenig and Likhachev, 2003]). For the most part, these are orthogonal to our work. The difficult part for an agent is deciding on the search goal; once achieved, then any of several different search algorithms can be used.

Having multiple agents participating in a search is a topic of recent interest. RoboCup is an example, but that work has more limited scope, and agents (players) generally have global knowledge.

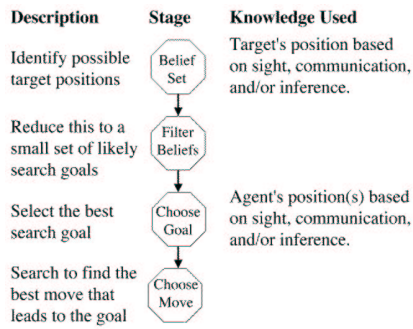


Figure 1: Framework for MAMT solutions

3 Problem Description

The MAMT domain has the following properties. **Agents and target:** multiple agents pursuing a single moving target. **Grid:** $m \times n$ in size, with randomly-placed obstacles. All agents and the target have complete knowledge of the grid topology. **Moving:** all moves are horizontal or vertical and are made simultaneously. **Starting position:** The target always starts in the middle of the grid. The agents are all placed in the lower left corner of the grid. The target is visible to at least one agent. **Vision:** can “see” anything that is in an unobstructed direct line. **Communication:** between moves, agents communicate with any agent that is visible to them. The agents exchange information as to where they believe the target and other agents are located. **Objective:** catch the target in the fewest number of moves.

4 Multiple Agent Moving Target Search

The intent of this work is not to build a new search algorithm. Rather, we want to plug standard search algorithms into a framework that, given a goal selection, will find the “best” way to reach the objective.

When an agent does not know the exact position of the target (from vision or communication), it must maintain a *belief set* of where the target might be. The belief set can take into account the topology of the grid, knowledge of the opponent, and the time since the last known target location. As the time increases since the last sighting, the knowledge of where the target is gets fuzzier. An agent should choose its search area based on its beliefs about the target and other agents.

Figure 1 shows the four-step method that is the framework used for specifying our solution algorithms. There is no “right” way of solving any of these steps. In the following we detail several algorithm alternatives.

4.1 Belief Set

Whenever an agent knows the exact location of the target, that agent’s belief set contains only one location, otherwise it can grow. Since this is a real-time search application, and real agents have limited memory, the belief set size is limited. Before a move, each agent sends its belief set information about the target and other agents to any agent it can see (who may, in turn forward it to agents that they see).

We implemented three strategies for maintaining the belief set. **All-scenarios.** Expand the current belief set to include all possible locations that can be reached in one more move.

Region belief set. This set has the same update as the all-scenarios belief set. After the search goal is chosen, the agent *commits* to only consider beliefs that are connected to the goal location. **Single-location:** The agent maintains a single belief (one grid square). The belief is updated by choosing a random direction and moving the belief in that direction until an obstacle is encountered or new target information is available.

We use a simple greedy algorithm to approximate the four “corners” of the variable shaped belief set. This subset of the belief set is called the *filtered belief set*.

4.2 Goal Selection

Each agent selects a goal from their filtered belief set. Randomly selecting a location from this set is an obvious control strategy to implement. However, a more intelligent strategy is needed – one that considers information about other agents.

The *difference metric* is used to identify a goal that ideally is (a) closest to the agent and yet (b) farthest from the closest other agent. For each location in the filtered belief set, we compute two metrics: the distance from the agent to the belief, and the minimum distance from the belief to the last known agent positions. These values can be determined by search (expensive) or by heuristics (inaccurate).

For each location in the filtered belief set, the agent computes the difference of the above two values, and chooses the one with the minimum difference. The idea is that the agent should assist the other agents by covering the possible escapes of the target that are hard for the other agents to reach.

4.3 Search

Given a goal, each agent performs a search to find a “best” move that progresses towards that goal. Since this has to be a real-time algorithm, the search algorithm is allocated a fixed number of search nodes (approximating a fixed amount of time per decision). The manhattan distance is used as the search evaluation function.

We experimented with two search algorithms. **Single-agent minimin search** [Korf, 1990]: Here the agent uses single-agent search to find the shortest path to the goal location. The search has the effect of selecting the move that tries to chase the target. **Adversarial search:** The search can alternate between moves for the agent (move towards the target) and target (move away from the agent). This becomes an alpha-beta search, where the agent tries to minimize the minimax value of the search (the distance from the target).

4.4 Comments

For non-trivial belief sets, as long as information is known about other agent’s positions the agents will separate to avoid redundancy in the search. Each agent wanders about the grid trying to maximize their coverage, as a function of what they know (about the target and other agents). In many cases (especially for large mazes) an agent may go a long time without getting an update on the target’s position, effectively negating the effectiveness of the belief set.

5 Experiments

There are a variety of target strategies that can be investigated. The strategy used is a weighted combination of four

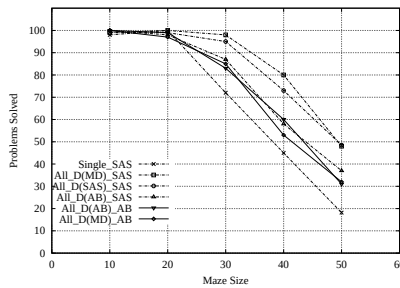


Figure 2: Comparing solutions

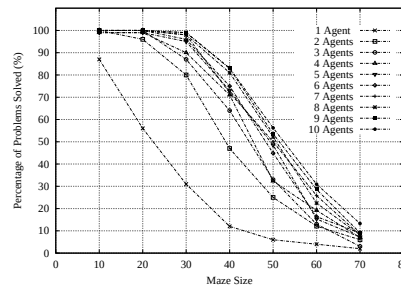


Figure 3: Varying size and # of agents

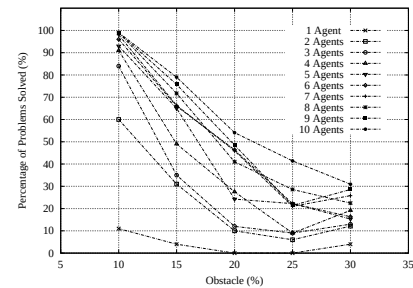


Figure 4: Varying the obstacle density

sub-strategies: distance, mobility, visibility, and random. The maximum scoring move is selected. The weights were hand-tuned based on the perceived realism of the target's behavior.

Solutions to MAMT instances were tested using grids of sizes from 10×10 to 70×70 . Grids had obstacles randomly placed, occupying 10% to 30%, in increments of 5%, of the space. The experiments used 1 to 10 agents. Each pursuer was allocated 50,000 search nodes to make its decision. If search was used to compute the difference metric, then half the search nodes were allocated to this task, and the other half to move selection. Agents were allowed a maximum belief set size of 100. An experiment ended when one of the pursuers caught the target, or when a maximum number of moves was reached. For an $n \times n$ grid, the maximum was set to $15 \times n$. With few exceptions, the target was caught in less than $8 \times n$ moves, or not at all.

Figure 2 compares several different solutions. The belief set was maintained using one of: all-scenarios (ALL), region, and single-location (Single). Except for a single-location belief set, the choice of goal was done using the difference metric based on either manhattan distance (no search) – D(MD), single-agent search – D(SAS), or alpha-beta – D(AB). Having chosen a goal, single-agent search (SAS) or alpha-beta (AB) was used to select the best move. The graph shows the percentage of problems where the target was caught (100 trials with different random seeds per data point) as a function of the maze size.

The control experiment is randomly selecting a goal (single-location), and using single-agent search (SAS) to decide on the move choice. Not surprisingly, this gets poor performance. Region was expected to perform quite well, but instead its results were mixed (not shown).

All_D(MD)_SAS and All_D(SAS)_SAS performed best in all our experiments, with a preference for the manhattan difference metric. That the difference heuristic gets the best performance is gratifying, since it is better informed by using beliefs about the other agents. Using a simple heuristic appears to be as good as or better than using search for determining the “best” search goal. The simplest way of computing the difference metric – using static manhattan distance instead of more accurate search – also leads to shorter solution lengths (up to 10% on average). This shows that it is more beneficial to invest search effort in move selection than in goal selection.

Alpha-beta is out-performed by single-agent search. Since alpha-beta takes into account the target's moves, it cannot reach the search depths that single-agent search can and,

hence, usually yields a lesser-quality solution.

In cases where the target eluded capture, a familiar pattern emerged. The target would stay hidden in a small area, and the agent's knowledge of where the target was became obsolete and effectively useless. The agents would independently wander about, hoping to find the target. In the real world, if such a scenario arises, the agents should wait until more help arrives and then begin the search anew, going through the entire grid systematically.

Figure 3 shows that more agents are better than fewer (using All_D(MD)_SAS). Note that as the number of agents increases, the number of nodes per agent per search in the goal selection gets smaller (recall the *total* search size is limited). Even so, adding more agents is beneficial despite less resources available.

Figure 4 shows that as the mazes become more congested with obstacles, it gets harder for the agents to find the target. Essentially, a higher percent of obstacles gives the target more opportunities to hide. However, at 30% of obstacles, our target starts having problems with avoiding the dead ends and is sometimes caught more easily.

Several control experiments were also done, and they gave predictable results. Simplistic targets (e.g., random, or avoid) were much easier to catch.

6 Future Work and Conclusions

This problem domain is rich in possibilities, and can be extended to increase the “realism” of the simulations. Some examples include: multiple moving targets, more realistic communication, creating “human-like” target behavior, and opponent modeling. The framework used of this research has many opportunities for interesting extensions.

7 Acknowledgments

This research was supported by NSERC, iCORE, and IRIS.

References

- [Ishida and Korf, 1995] T. Ishida and R. Korf. Moving target search: A real-time search for changing goals. *IEEE PAMI*, 17(6):609–619, 1995.
- [Koenig and Likhachev, 2003] S. Koenig and M. Likhachev. D* lite. *AAAI*, pages 476–483, 2003.
- [Korf, 1990] R. Korf. Real-time heuristic search. *Artificial Intelligence*, 42(2–3):189–211, 1990.
- [Stentz, 1995] A. Stentz. The focussed D* algorithm for real-time replanning. In *IJCAI*, pages 1652–1659, 1995.

Delayed Duplicate Detection: Extended Abstract

Richard E. Korf

Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095
korf@cs.ucla.edu

Abstract

Best-first search is limited by the memory needed to store nodes in order to detect duplicates. Disks can greatly expand the amount of storage available, but randomly accessing a disk is impractical. Rather than checking newly-generated nodes as soon as they are generated, we append them to a disk file, then sort the file, and finally scan the sorted file in one pass to detect and remove duplicate nodes. This also speeds up such searches that fit entirely in memory, by improving cache performance. We implement this idea for breadth-first search, performing the first complete searches of the 2×7 sliding-tile puzzle, and the 18-disk, 4-peg Towers of Hanoi puzzle.

1 Introduction: The Problem

Best-first search algorithms, such as breadth-first search (BFS), Dijkstra's algorithm [Dijkstra, 1959], and A* [Hart, Nilsson, & Raphael, 1968], store every node that is generated, in either the Open list or the Closed list. One reason for doing this is to detect duplicate nodes, and avoid expanding a state more than once. As a result, these algorithms are limited by the available memory.

In some problems, this memory limitation can be avoided by depth-first searches (DFS) such as depth-first iterative-deepening (DFID) or iterative-deepening-A* (IDA*) [Korf, 1985], but DFS can generate exponentially more nodes than BFS. For example, in a rectangular-grid problem space, BFS will generate $O(r^2)$ nodes within a radius of r , while DFID will generate $O(3^r)$ nodes. While there are ways of detecting some duplicate nodes in a DFS [Taylor & Korf, 1993], they do not apply to all problem spaces.

2 Frontier Search

An algorithm called *frontier search* [Korf, 1999; Korf & Zhang, 2000] saves only the Open list of nodes at the frontier of the search, and not the Closed list of nodes that have been expanded, thus saving some memory.

With each node, it stores a *used-operator bit* for each operator, to indicate whether the neighboring state reached by that operator has already been generated. For example, the sliding-tile puzzles require four such bits for each state, one for each direction in which a tile could move. When a parent node is expanded, only children that are reached via unused operators are generated, and the parent node is deleted from memory. In each child node, the operator that generates the parent is marked as used. When duplicate states are found in the Open list, only one copy is kept, and any operator that is marked as used in any copy is marked as used in the retained copy. Since the closed list is not stored, reconstructing the solution path requires additional work. See [Korf, 1999; Korf & Zhang, 2000] for two ways to do this.

The memory required by frontier search is proportional to the maximum size of the Open list, or the *width* of the problem space, rather than the size of the space. For example, in the grid space mentioned above, the width of the space grows only linearly with the search radius, while the entire space grows quadratically. As another example, the width of the n -disk, 3-peg Towers of Hanoi space is only 2^n states, while the entire space contains 3^n states. Frontier search is still limited by the memory required to store the Open list, however.

3 Delayed Duplicate Detection (DDD)

Hard disks with hundreds of gigabytes of storage are available for less than a dollar per gigabyte, which is over a hundred times cheaper than memory. Because of high latency, however, a disk behaves more like a sequential device, such as a magnetic-tape drive, with large capacity and high bandwidth, but only if it is accessed sequentially. To quickly detect duplicate states in a search algorithm, however, nodes are usually stored in a hash table, which is designed to be accessed randomly.

Our solution to this problem is rather than checking each newly-generated node for duplicates as soon as it is generated, we append each node to a disk file containing previously generated nodes. At some point later we sort the file, thereby bringing together nodes representing the same state. Then we scan the sorted list of nodes in one pass, merging any duplicate nodes.

3.1 Breadth-First Frontier Search

As a simple example, we describe breadth-first frontier search with delayed duplicate detection. BFS is normally implemented with a FIFO queue, which we implement with two disks files, an input file and an output file. Initially, the input file contains the initial state. As we read each node in the input file, we expand it, and write its children to the output file, with no duplicate checking. When the input file is exhausted, we delete it. At that point, the output file contains all the nodes at the next depth, including any duplicate nodes. We then sort the nodes in the output file by their state representations, which brings together any duplicate nodes representing the same state. Next, we linearly scan the output file, merging duplicate nodes and ORing their used operator bits, and write one copy of each state to a new input file, deleting the output file when we're done. This completes one level of the breadth-first search. The algorithm continues until expanding the nodes in the input file doesn't generate any more nodes in the output file.

3.2 Sorting the Disk Files

Algorithms for sorting disk files are well-known. See [Garcia-Molina, Ullman, & Widom, 2000], pp. 42-48. The basic algorithm is to read as much of the unsorted file as will fit into memory, sort it in memory using quicksort, for example, and write the sorted portion of the file to a new subfile. Continue until the entire original file has been read and written into a set of sorted subfiles. Then, all the sorted subfiles are merged in one pass, storing the head of each file in memory, and writing the lowest record to a final sorted output file.

3.3 DDD in Memory

Surprisingly, delayed duplicate detection is useful even when all nodes fit in memory, resulting in reduced running time due to improved cache performance. In the standard implementation of breadth-first search in memory, the Open list is stored in a hash table. As each new node is generated, it is looked up in the hash table, which often results in a cache miss, since the hash function is designed to randomly scatter the nodes. A DDD implementation doesn't use a hash table, but a single FIFO queue in memory, reading nodes off the head of the queue, and appending them to the tail. Once a level of the search is completed, the queue is sorted in memory using an algorithm such as quicksort, and the sorted queue is scanned, merging duplicate nodes. The advantage of this approach is that the queue is only accessed at the head and tail, or at two points in between during quicksort, and hence most memory references will reside in cache, reducing the running time.

4 Experiments

We implemented a breadth-first search on sliding-tile puzzles, and the 4-Peg Towers of Hanoi problem.

4.1 Sliding-Tile Puzzles

[Schofield, 1967] published a complete breadth-first search of the 3×3 Eight puzzle. We completed a BFS for all sliding-tile puzzles up to the 2×7 Thirteen Puzzle. Table 1 below shows the results. The first column gives the x and y dimensions of the puzzle, and the second column gives the number of moves needed to reach all solvable states, starting with the blank in a corner position. This is also the worst-case optimal solution length, for a goal with the blank in a corner. The third column gives the number of solvable states, which is $(xy)!/2$, and the fourth column gives the width of the problem space, which is the maximum number of nodes at any depth.

Size	Moves	Total States	Max Width
2×2	6	12	2
2×3	21	360	44
2×4	37	20,160	1,999
3×3	31	181,440	24,047
2×5	55	1,814,400	133,107
2×6	80	239,500,800	13,002,649
3×4	53	239,500,800	21,841,159
2×7	108	43,589,145,600	1,862,320,864

Table 1: Sliding-Tile Puzzle Results

The 3×4 and 2×6 Eleven Puzzles were the largest that we could solve in memory. We implemented both a standard BFS algorithm, using one bit of memory per state, and also breadth-first frontier search with delayed duplicate detection. The standard BFS required 17.5 minutes, while the frontier search with DDD required only 9.6 minutes, on a 440 Megahertz Sun Ultra 10 workstation. This demonstrates that DDD frontier search is useful even for problems that fit in memory.

The 2×7 Fourteen Puzzle was the largest we could search exhaustively with our 120 gigabyte disk. At 8 bytes per state, this problem required about 15 gigabytes of disk storage, and ran for 51 hours, 13 minutes. The ratio of the problem size to the problem width is about 23.4, illustrating the advantage of frontier search.

4.2 Towers of Hanoi

For the standard 3-peg Towers of Hanoi problem, there is a simple algorithm that guarantees the shortest path between any pair of states. The 4-peg Towers of Hanoi puzzle, known as Reve's puzzle, is much more interesting. There exists an algorithm for finding a solution, and a conjecture that it generates optimal solutions, but the conjecture remains unproven. [van de Liefvoort, 1992] contains a good bibliography for this problem. [Szegedy, 1999] gives bounds for the k -peg version, but they include an unspecified constant in the exponent.

For the sliding-tile puzzles, all paths between any pair of states have the same even-odd parity, and the algorithm described in Section 3.1 works correctly. For the Towers of Hanoi, however, two states can have both even and odd length paths between them. In that case, frontier search with delayed duplicate detection can fail by

leaking into the interior of the search. One solution to this problem is to store two complete levels of the search at a time. See our full paper for the details [Korf, 2003].

Using this algorithm, we were able to exhaustively search all problems through 18 disks, starting with all disks on one peg. Table 2 shows the results. The first column gives the number of disks, and the second column shows the minimum number of moves required to move all the disks from one peg to another. The third column gives the total number of states of the problem, which is 4^n , where n is the number of disks. The fourth column shows the maximum width of the graph, starting from a state with all disks on one peg. In all cases, the optimal number of moves equals the conjectured minimal number of moves. The 18-disk problem took almost 6 days to run, and at 8 bytes per state required about 30 gigabytes of disk space. Note that the ratio of the total number of states to the problem width for the 18-disk problem is almost 34.5.

Disks	Moves	Total States	Max Width
1	1	4	3
2	3	16	6
3	5	64	30
4	9	256	72
5	13	1,024	282
6	17	4,096	918
7	25	16,284	2,568
8	33	65,536	9,060
9	41	262,144	31,638
10	49	1,048,576	109,890
11	65	4,194,304	335,292
12	81	16,777,216	1,174,230
13	97	67,108,864	4,145,196
14	113	268,435,456	14,368,482
15	129	1,073,741,824	48,286,104
16	161	4,294,967,296	162,989,898
17	193	17,179,869,184	572,584,122
18	225	68,719,476,736	1,994,549,634

Table 2: 4-Peg Towers of Hanoi Results

5 Conclusions and Further Work

We showed that delaying the detection of duplicate nodes in a breadth-first search can effectively make use of very large capacity disk-storage systems. We also showed how to combine this idea with frontier search. In addition, we demonstrated that these ideas can also speed up a search that occurs entirely in memory. Using these algorithms, we completed breadth-first searches of sliding-tile puzzles with up to 43 billion nodes, and 4-peg Towers of Hanoi problems with up to 68 billion nodes. For the 4-peg Towers of Hanoi, we verified a conjecture regarding optimal solution lengths to 18 disks.

Current work is focussed on implementing these techniques for more complex best-first search algorithms such as Dijkstra's algorithm [Dijkstra, 1959] and A* [Hart, Nilsson, & Raphael, 1968].

6 Acknowledgments

Larry Taylor brought the 4-Peg Towers of Hanoi problem to my attention, and also used disk storage in a breadth-first search to find duplicate operator strings [Taylor & Korf, 1993]. Thanks to Jianming He for verifying the Towers of Hanoi results. This research was supported by NSF under grant No. EIA-0113313, by NASA and JPL under contract No. 1229784, and by the State of California MICRO grant No. 01-044.

References

- [Dijkstra, 1959] Dijkstra, E. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik* 1:269–271.
- [Garcia-Molina, Ullman, & Widom, 2000] Garcia-Molina, H.; Ullman, J. D.; and Widom, J. 2000. *Database System Implementation*. Upper Saddle River, N.J.: Prentice-Hall.
- [Hart, Nilsson, & Raphael, 1968] Hart, P.; Nilsson, N.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Systems Science and Cybernetics* SSC-4(2):100–107.
- [Korf & Zhang, 2000] Korf, R., and Zhang, W. 2000. Divide-and-conquer frontier search applied to optimal sequence alignment. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-2000)*, 910–916.
- [Korf, 1985] Korf, R. 1985. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence* 27(1):97–109.
- [Korf, 1999] Korf, R. 1999. Divide-and-conquer bidirectional search: First results. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, 1184–1189.
- [Korf, 2003] Korf, R. 2003. Delayed duplicate detection: initial results. In *Proceedings of the IJCAI-03 Workshop on Model Checking and Artificial Intelligence*.
- [Schofield, 1967] Schofield, P. 1967. Complete solution of the eight puzzle. In Meltzer, B., and Michie, D., eds., *Machine Intelligence 3*. New York: American Elsevier. 125–133.
- [Szegedy, 1999] Szegedy, M. 1999. In how many steps the k peg version of the towers of hanoi game can be solved? In *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS '99)*, LNCS 1563. Trier, Germany: Springer-Verlag.
- [Taylor & Korf, 1993] Taylor, L., and Korf, R. 1993. Pruning duplicate nodes in depth-first search. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-93)*, 756–761.
- [van de Liefvoort, 1992] van de Liefvoort, A. 1992. An iterative algorithm for the reve's puzzle. *The Computer Journal* 35(1):91–92.

A Portfolio Approach to Algorithm Selection

Kevin Leyton-Brown and Eugene Nudelman and Galen Andrew and Jim McFadden and Yoav Shoham
{kevinlb;eugnud;galand;jmcf;shoham}@cs.stanford.edu, Stanford University, Stanford CA 94305

1 Introduction

Although some algorithms are better than others on average, there is rarely a best algorithm for a given problem. Instead, it is often the case that different algorithms perform well on different problem instances. Not surprisingly, this phenomenon is most pronounced among algorithms for solving $\mathcal{NP}$ -Hard problems, because runtimes for these algorithms are often highly variable from instance to instance. When algorithms exhibit high runtime variance, one is faced with the problem of deciding which algorithm to use; in 1976 Rice dubbed this the “algorithm selection problem” [8]. In the nearly three decades that have followed, the issue of algorithm selection has failed to receive widespread attention, though of course some excellent work does exist. By far, the most common approach to algorithm selection has been to measure different algorithms’ performance on a given problem distribution, and then to use only the algorithm having the lowest average runtime. This “winner-take-all” approach has driven recent advances in algorithm design and refinement, but has resulted in the neglect of many algorithms that, while uncompetitive on average, may offer excellent performance on particular problem instances. Our consideration of the algorithm selection literature, and our dissatisfaction with the winner-take-all approach, has led us to ask the following two questions. First, what general techniques can we use to perform per-instance (rather than per-distribution) algorithm selection? Second, once we have rejected the notion of winner-take-all algorithm evaluation, how ought novel algorithms to be evaluated? We offer the following answers:

1. Algorithms with high average running times can be combined to form an algorithm portfolio with low average running time when the algorithms’ easy inputs are sufficiently uncorrelated.
2. New algorithm design should focus on problems on which the current algorithm portfolio performs poorly.

Readers familiar with the boosting paradigm in machine learning [9] will recognize that boosting uses similar ideas: combining weak classifiers into a much stronger ensemble by iteratively training new classifiers on data on which the ensemble performs poorly.

2 Algorithm Portfolios

In our previous work [6] we demonstrated that statistical regression can be used to learn surprisingly accurate algorithm-specific models of the empirical hardness of given distributions of problem instances. In short, the method proposed in that work is:

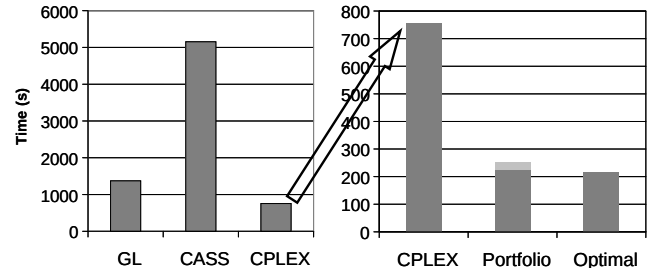


Figure 1: Algorithm and Portfolio Runtimes

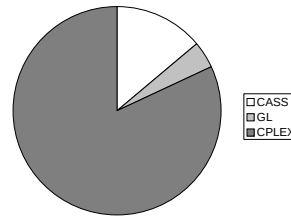


Figure 2: Optimal

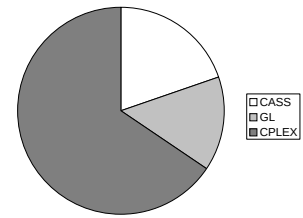


Figure 3: Selected

1. Use domain knowledge to select features of problem instances that might be indicative of runtime.
2. Generate a set of problem instances from the given distribution, and collect runtime data for the algorithm on each instance.
3. Use regression to learn a real-valued function of the features that predicts runtime.

Given this existing technique for predicting runtime, we now propose building portfolios of multiple algorithms as follows:

1. Train a model for each algorithm, as described above.
2. Given an instance:
 - (a) Compute feature values
 - (b) Predict each algorithm’s running time
 - (c) Run the algorithm predicted to be fastest

2.1 WDP Case Study: Past Work

Our case study is based on the data collected in our previous work [6]. In that work we have constructed models for predicting the running time of the CPLEX solver on the winner determination problem (WDP), which is an $\mathcal{NP}$ -Complete combinatorial optimization problem formally equivalent to weighted set-packing. We have also created models for two other WDP algorithms: GL (Gonen-Lehmann) [3], a simple

branch-and-bound algorithm with CPLEX’s LP solver as its heuristic, and CASS [1], a more complex branch-and-bound algorithm with a non-LP heuristic. The data set consists of 4500 instances of a fixed size (256 goods and 1000 non-dominated bids), sampled uniformly from CATS [7] instance generator. Since our methodology relies on machine learning, we split the data into training, validation, and test sets. We report our portfolio runtimes only on the test set that was never used to train or evaluate models. Runtime data was collected on 550 MHz Pentium Xeons, running Linux 2.2.

2.2 WDP Case Study: Portfolios

We now describe new results. Fig. 1 compares the average runtimes of our three algorithms to that of the portfolio. Note that CPLEX would be chosen under winner-take-all algorithm selection. The “optimal” bar shows the performance of an ideal portfolio where algorithm selection is performed perfectly and with no overhead. The portfolio bar shows the time taken to compute features (light portion) and the time taken to run the selected algorithm (dark portion). Despite the fact that CASS and GL are much slower than CPLEX on average, the portfolio outperforms CPLEX by more than a factor of 3. Further, neglecting the cost of computing features, our portfolio’s selections take only 5% longer to run than the optimal selections. Figs. 2 and 3 show the frequency with which each algorithm is selected in the ideal portfolio and in our portfolio. They illustrate the quality of our algorithm selection and the relative value of the three algorithms. While our portfolio does not always make the right choice, most of the mistakes occur when algorithms have very similar running times. These mistakes are not very costly, which explains why our portfolio’s choices have a running time so close to optimal. These results show that our portfolio methodology can work very well even with a small number of algorithms, and even when one algorithm has superior average performance.

3 Focused Algorithm Design

Once it has been decided to select algorithms from a portfolio on a per-instance basis, it is necessary to reexamine the way we design and evaluate algorithms. Since the purpose of designing new algorithms is to reduce the time that it will take to solve problems, designers should aim to produce new algorithms that complement an existing portfolio given a distribution D reflecting problems that will be encountered in practice. The instances with the greatest potential for improvement will be hard for the portfolio, common in D , or both. The full version of this paper describes a technique for using rejection sampling to automatically generate such instances. In Figures 4 and 5 we show how our techniques are able to automatically skew two of the easiest CATS instance distributions towards harder regions. In fact, for these two distributions we generated instances that were (respectively) 100 and 50 times harder than anything we had previously seen! Moreover, the *average* runtime for the new distributions was greater than the observed *maximum* running time on the original distribution.

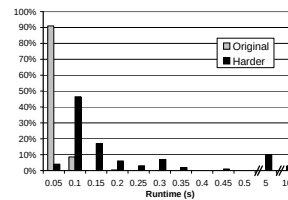


Figure 4: Matching

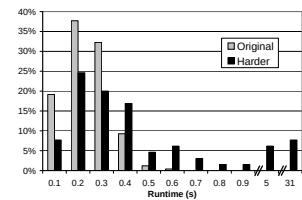


Figure 5: Scheduling

4 Conclusions

Learned runtime models may be used to create algorithm portfolios that outperform their constituent algorithms. These models can also be used to induce harder benchmark distributions for use in the development and evaluation of new algorithms. Our case study on combinatorial auctions demonstrates that a portfolio composed of CPLEX and two older—and generally *much* slower—algorithms outperforms CPLEX alone by about a factor of 3. In the full version of this paper we describe our methodology in more detail, and also:

- Show how to reduce the time spent computing features without substantially degrading portfolio performance;
- Demonstrate ways of using response variable transformations to focus portfolios on metrics other than average running time;
- Explain how to induce distributions with characteristics other than hardness (e.g. realism);
- Compare our approach to previous work that executes algorithms in parallel [2]; uses classification instead of regression [4]; or considers the problem as an MDP [5].

References

- [1] Y. Fujishima, K. Leyton-Brown, and Y. Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *IJCAI*, 1999.
- [2] C. Gomes and B. Selman. Algorithm portfolios. *Artificial Intelligence*, 126(1-2):43–62, 2001.
- [3] R. Gonen and D. Lehmann. Linear programming helps solving large multi-unit combinatorial auctions. Technical Report TR-2001-8, Leibniz Center for Research in Computer Science, April 2001.
- [4] E. Horvitz, Y. Ruan, C. Gomes, H. Kautz, B. Selman, and M. Chickering. A Bayesian approach to tackling hard computational problems. In *UAI*, 2001.
- [5] M. Lagoudakis and M. Littman. Algorithm selection using reinforcement learning. In *ICML*, 2000.
- [6] K. Leyton-Brown, E. Nudelman, and Y. Shoham. Learning the empirical hardness of optimization problems: The case of combinatorial auctions. In *CP*, 2002.
- [7] K. Leyton-Brown, M. Pearson, and Y. Shoham. Towards a universal test suite for combinatorial auction algorithms. In *ACM EC*, 2000.
- [8] J. R. Rice. The algorithm selection problem. *Advances in Computers*, 15:65–118, 1976.
- [9] R. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

A New Node Centroid Algorithm for Bandwidth Minimization

Andrew Lim

Department of Industrial
Engineering and Engineering
Management, Hong Kong University
of Science and Technology
Clear Water Bay, Hong Kong
iealim@ust.hk

Brian Rodrigues

School of Business
Singapore Management
University, 469 Bukit
Timah Road
Singapore 259756
br@smu.edu.sg

Fei Xiao

School of Computing
National University
of Singapore
3 Science Drive 2
Singapore 117543
xiaofei@comp.nus.edu.sg

Abstract

We propose a Node Centroid method with Hill-Climbing to solve the well-known matrix bandwidth minimization problem, which is to permute rows and columns of the matrix to minimize its bandwidth. Many heuristics have been developed for this NP-complete problem including the Cuthill-McKee (CM) and the Gibbs, Poole and Stockmeyer (GPS) algorithms. Recently, heuristics such as Simulated Annealing, Tabu Search and GRASP have been used, where Tabu Search and the GRASP with Path Relinking have achieved significantly better solution quality than the CM and GPS algorithms. Experimentation shows that the Node Centroid method achieves the best solution quality when compared with these while being much faster than the newly-developed algorithms.

1 Introduction

For a symmetric matrix $A = \{a_{ij}\}$, the matrix bandwidth minimization problem is to find a permutation of rows and columns of the matrix A so as to bring all the non-zero elements of A to reside in a band that is as close as possible to the main diagonal, that is to $\text{Minimize}\{\max\{|i - j| : a_{ij} \neq 0\}\}$. The bandwidth minimization problem also can be stated in the context of graph as: Let $G(V, E)$ be a graph on n vertices. Label all vertices with different labels from the set $1, 2, \dots, n$ so that $f(v)$ is the label for vertex v . Then, with the bandwidth of G defined to be $B_f(G) = \text{Max}\{|f(u) - f(v)| : \forall u, \forall v, (u, v) \in E(G)\}$, the bandwidth minimization problem is to find a labeling, f , which minimizes $B_f(G)$. Note, that we transform a graph bandwidth problem into a matrix bandwidth problem by using its incidence matrix. The bandwidth minimization problem was proved to be NP-complete by Papadimitriou [Papadimitriou, 1976].

Many heuristic algorithms have been proposed in solving the bandwidth minimization problem, considering its importance. In 1969, the classical CM [Cuthill *et al.*, 1969] algorithm appeared, which used Breadth-First

Search to construct a level structure of the graph. The GPS algorithm [Gibbs *et al.*, 1976], which is also based on the level structure, can obtain almost the same result as the CM algorithm, but about eight times faster than the CM algorithm. Esposito *et al.* [Esposito *et al.*, 1998] proposed a new WBRA (Wonder Bandwidth Reduction Algorithm), which can achieve better result than the classical CM and GPS algorithm. Marti *et al.* [Marti *et al.*, 2001] proposed a new Tabu Search method in which candidate list strategy was used to accelerate the selection of move in a neighborhood. Extensive Experimentation showed that their Tabu Search outperformed best-known algorithms in terms of solution quality. Recently, Pinana *et al.* [Pinana *et al.*, 2001] used a GRASP (Greedy Randomized Adaptive Search Procedure) with Path Relinking method for the problem. Computational Results showed that the GRASP with Path Relinking achieved the best solution in quality, though it is slower than the Tabu Search.

We have proposed a new NCHC algorithm (Node Centroid method with Hill-Climbing) to solve the bandwidth minimization problem. Experimentation shows that our NCHC algorithm has outperformed other heuristic algorithm on the solution quality. Meanwhile the fast version of our new NCHC algorithm is comparable with CM and GPS algorithm on speed, which is about 100 times faster than the newly developed Tabu Search and GRASP with Path Relinking. In next Section we present the general framework of the algorithm. Computational Results are reported in Section 3. Finally, we draw our conclusion and provide some directions for future research.

2 The Node Centroid with Hill-Climbing algorithm

The Node Centroid method with Hill Climbing (NCHC) employs the strategy of using the Node Centroid method for global search with Hill-Climbing in local search. An initial labeling is generated by performing Breadth-First Search (BFS) on the given graph representation of the matrix with random start vertex. We then use the Node Centroid method to adjust vertices to a central (centroid) position among its neighbors. From this, a new labeling is created on which we perform Hill Climbing

to obtain local optima. The Node Centroid method and Hill Climbing iterate a number of times, following which a new initial labeling is generated by BFS. The entire process is repeated several times within the NCHC algorithm which is described in Algorithm 1 given below:

Algorithm 1 NCHC

```

for  $i = 1$  to restart_Times do
  Initialize(labeling)
  for  $j = 1$  to NC_Times do
    NC(labeling)
    if  $j \bmod 2 \equiv 1$  then
      HC(labeling)
    end if
  end for
end for

```

In the algorithm, the NC component comprises of Node Centroid labeling adjustments and HC denotes Hill Climbing. These will be described in more detail in the following sections. The HC procedure is invoked only every other time we perform the NC since it is the bottleneck for the speed of the algorithm and since experimentation has shown that this frequency proportion works well.

3 Computational Results

We have compared our new NCHC algorithm with algorithms developed by other researchers on three sets of test cases from the *Harwell-Boeing Sparse Matrix Collection* (<http://math.nist.gov/MatrixMarket/data/Harwell-Boeing/>) of standard test matrices, which represent a large spectrum of scientific and engineering applications. We have also developed a fast version of our NCHC algorithm, denoted FNCHC.

The first two test sets have also been used in [Pinana *et al.*, 2001]. Experimental results are shown in Table 1

Table 1: 113 instances from Harwell-Boeing

	GPS	TS(Marti)	GRASP_PR	NCHC	FNCHC
33 instances					
$B_f(G)$	31.42	23.33	22.52	22.39	22.85
Deviation	40.33%	4.20%	0.58%	0.00%	2.05%
CPU seconds	0.003	2.36	4.21	3.37	0.32
CPU	K7 1.2G	P4 1.6G	P4 1.6G	P4 1.6G	P4 1.6G
80 instances					
$B_f(G)$	156.38	100.78	99.43	97.99	106.09
Deviation	59.59%	2.85%	1.47%	0.00%	8.27%
CPU seconds	0.11	121.66	323.19	40.20	0.76
CPU	K7 1.2G	P4 1.6G	P4 1.6G	P4 1.6G	P4 1.6G

As shown in table 1, the best solution in quality has been obtained by our NCHC, which is also much faster than the newly developed GRASP_PR. Our FNCHC has achieved very good solution in a short time, which is more than 100 times faster than the TS and GRASP_PR. Though our FNCHC is slower than the GPS algorithm

but it obtained solution 37% and 47% better than the classical GPS algorithm. We also compare our approach with the Esposito's WBRA and TS [Esposito *et al.*, 1998] on the DWT test set from the Harwell-Boeing Sparse Matrix Collection, where our new NCHC also get the best results, and our FNCHC obtain better result than the WBRA and TS in a short time.

4 Conclusions

We have proposed a Node Centroid adjustment method with Hill Climbing for the well-known matrix bandwidth minimization problem. Experimentation has shown that the Node Centroid global search works well with Hill Climbing for this problem. Best solutions in quality are achieved by the new NCHC algorithm, while the FNCHC provides good solution quality at fast speeds and is comparable in speed to the fast GPS algorithm. It also indicates that we can apply the new Node Centroid procedure to other similar combinatorial optimization problems, such as matrix profile reduction and Minimum Linear Arrangement Problem in the future.

References

- [Cuthill *et al.*, 1969] Cuthill, E., McKee, J., 1969. Reducing the bandwidth of sparse symmetric matrices. In: Proceedings of the ACM National Conference, Association for Computing Machinery, New York, pp. 157-172
- [Esposito *et al.*, 1998] Esposito, A., Catalano, M., S., Malucelli F., Tarricone, L. 1998. Sparse Matrix Bandwidth Reduction: Algorithms, applications and real industrial cases in electromagnetics, 1998. High Performance Algorithms for Structured Matrix Problems, Advances in the theory of Computation and Computational Mathematics Volume 2, pp. 27-45.
- [Marti *et al.*, 2001] Marti, R., Laguna, M., Glover, F. and Campos, V., 2001. Reducing the Bandwidth of a Sparse Matrix with Tabu Search, European Journal of Operational Research, 135(2), pp. 211-220.
- [Gibbs *et al.*, 1976] Gibbs, N.E., Poole, W.G., Stockmeyer, P.K., 1976. An algorithm for reducing the bandwidth and profile of sparse matrix. SIAM Journal on Numerical Analysis 13 (2), pp. 236-250.
- [Papadimitriou, 1976] Papadimitriou, C.H., 1976. The NP-completeness of the bandwidth minimization problem. Computing, 16:263-270
- [Pinana *et al.*, 2001] Pinana, E., Plana, I., Campos, V. Marti, R., 2002, GRASP and Path Relinking for the Matrix Bandwidth Minimization. In print, European Journal of Operational Research. <http://www.uv.es/~rmarti/>.

Combining Two Local Search Approaches to Hypergraph Partitioning

Arathi Ramani and Igor Markov

Department of Electrical Engineering and Computer Science,
University of Michigan, Ann Arbor 48109-2122
{ramania,imarkov}@eecs.umich.edu

Abstract

We study leading-edge local search heuristics for balanced hypergraph partitioning and Boolean satisfiability, intending the generalization of such heuristics beyond their original domains. We adapt the Fiduccia Mattheyses (FM) hypergraph partitioning heuristic to Boolean Satisfiability (SAT), and the WalkSAT SAT solver to hypergraph partitioning. Focusing on balanced hypergraph partitioning, we propose a combination of the classical FM heuristic and our “cross-over” heuristic WalkPart, and empirically show that it is more powerful than each component alone. Empirically, we show a 15% improvement in net cut and a 7% improvement in runtime over a leading-edge implementation of the FM heuristic.

1 Introduction

The focus of our work is on search heuristics for balanced min-cut hypergraph partitioning, a well-known combinatorial optimization problem. In general terms, the problem seeks to assign every vertex of a hypergraph to one of two subsets to minimize the number of connections between the two subsets, subject to having approximately the same number of vertices in each subset. Hypergraph partitioning is important in circuit layout because circuits are often represented by hypergraphs, e.g. in placement by recursive bisection. Many efficient hypergraph partitioning algorithms have been developed [3, 4]. Here, we discuss the combination of two local search heuristics: the Fiduccia-Mattheyses (FM) hypergraph partitioning heuristic [3], and the biased random walk heuristic, adapted from the WalkSAT SAT solver [5].

Our work is motivated by attempts to generalize well-known local search procedures [5, 3] beyond their original domains. We developed a SAT solver based on the FM algorithm [3], and a hypergraph partitioner, WalkPart, based on the WalkSAT algorithm from [5]. The WalkPart algorithm is described in Section 3. A detailed description of the SAT solver is beyond the scope of this paper. Our results show that WalkPart outperforms a highly tuned implementation of FM on several circuit benchmarks. The algorithms also produce very different solutions and appear to have different local minima, suggesting that a hybrid strategy might be effective. Empirical results show that the hybrid approach outperforms both algorithms, in terms of runtime and solution quality. Our main contributions are summarized as follows.

- We describe the balanced min-cut hypergraph partitioning algorithm, WalkPart, based on the WalkSAT satisfiability solver [5]
- We propose the combination of the FM and WalkPart heuristics to form a “hybrid” heuristic, and perform two experiments
- Empirically, we show that a hybrid strategy achieves significantly better performance than either heuristic alone

2 Background and Previous Work

2.1 Hypergraph Partitioning Problem:

A hypergraph $H(V, E)$ is defined as a set, V , of vertices, and a set, E , of hyperedges, where each edge $e \in E$ is a set of two or more vertices from V . The k -way hypergraph partitioning problem is stated as follows: Let $H(V, E)$ be a hypergraph with weighted vertices and edges. The problem asks to find a minimum cost partition P^k of the vertices of H to k disjoint subsets. The cost function $c(P^k)$ typically used is *edge cut*, the sum of the weights of the hyperedges *cut* by the partition (this is called min-cut partitioning). A hyperedge is cut exactly when all of its vertices are in more than one subset. Constraints may be imposed on a partition e.g. limiting the total vertex weight in each subset (*balance constraints*). The balance-constrained weighted hypergraph partitioning problem is NP -hard. The algorithms in this work deal with the balanced min-cut hypergraph bi-partitioning problem, i.e., we consider only the case where $k = 2$. In circuit layout, hypergraphs represent circuits by mapping vertices to signal nodes, and hyperedges to signal nets. In this context, we use the words “node” and “net” to mean “vertex” and “hyperedge” when referring to hypergraphs in this document.

2.2 The Fiduccia-Mattheyses Heuristic

The Fiduccia-Mattheyses (FM) [3] heuristic for bi-partitioning circuit hypergraphs is an iterative improvement algorithm. FM starts with an arbitrary initial solution and changes the solution by a sequence of moves, organized as passes. At the beginning of a pass, all vertices are free to move (unlocked). Each possible move is labeled with the immediate change in total cost it would cause (called “gain”). Iteratively, a move with highest gain is selected and executed, which may result in a lower, higher or unchanged solution cost¹. The moved vertex is locked, i.e., not allowed to move again during that pass. Since moving a vertex can change gains of adjacent vertices, all affected gains are updated. Selection and execution of a best-gain move, followed by gain update, are repeated until every vertex is locked. The best solution seen during the pass is adopted as the starting solution of the next pass. The algorithm terminates when a pass fails to improve solution quality.

The FM heuristic is now used as a component in multi-level partitioning algorithms [4] which create a series of clustered hypergraphs, partition the most clustered with “flat” FM and then refine unclustered versions of the produced partition by applying FM in incremental mode. A highly optimized, leading edge implementation of FM in this context is available in [2]. This work only deals with “flat” partitioning heuristics in the hope that improved flat heuristics will lead to improved multi-level heuristics. This assumption will be verified in our future work.

¹In a problem with balance constraints, moves should also not violate the legality constraint

Table 1: Results for 10 passes of a Hybrid Algorithm: FM is run once to completion, followed by 100K moves of WalkPart alternated with single passes of FM.

Benchmark Name	Nodes	Nets	WalkPart			FMPart		Hybrid		Cut Ratio	Time Ratio
			Moves	Time	Cut	Time	Cut	Time	Cut		
ibm01+10	12752	14111	2M	9.16	610.67	1.54	477.00	2.0	284.00	0.59	1.29
ibm01+4	12752	14111	500K	2.56	506.00	1.39	464.33	2.667	413.33	0.89	1.91
ibm02+10	19601	19584	500K	3.61	329.67	2.15	402.00	3.8	290.67	0.72	1.76
ibm02+4	19601	19584	500K	3.61	654.33	2.14	386.00	3.7	523.33	1.35	1.72
ibm03+10	23136	27401	2M	14.15	1603.33	3.84	1487.67	5.0	1157.33	0.77	1.30
ibm03+4	23136	27401	500K	3.84	1175.00	3.14	2488.00	5.86	1093.33	0.43	1.86
ibm04+10	27507	31970	500K	4.50	1589.00	5.30	718.67	9.0	606.66	0.84	1.69
ibm04+4	27507	31970	2M	14.94	2121.00	3.52	2505.00	5.0	1089.33	0.43	1.41
ibm05+10	29347	28446	2M	17.34	4804.67	9.54	2793.33	14.75	1869.66	0.66	1.54
ibm06+10	32498	34826	2M	12.73	750.00	5.63	1683.00	10.5	1117.00	0.66	1.86
ibm07+10	45926	48117	2M	18.07	2483.33	9.63	1347.33	13.98	1190.67	0.88	1.45
ibm08+10	51309	50513	2M	24.92	1902.67	11.76	3076.00	25.5	1574.66	0.51	2.16
ibm09+10	53395	60902	2M	19.60	2069.00	14.38	1936.00	29.0	1365.33	0.70	2.01
ibm10+10	69429	75196	2M	19.79	3414.67	15.36	2200.33	30.22	1440.33	0.65	1.96
ibm11+10	70558	81454	2M	21.61	2433.33	19.78	4203.33	34.13	2992.00	0.71	1.72
ibm12+10	71076	77240	2M	22.99	3803.67	23.03	4200.67	36	4525.33	1.07	1.56
Average										0.7	1.7

2.3 The WalkSAT Satisfiability Solver

The WalkSAT algorithm [5] is a stochastic local search heuristic for Boolean Satisfiability. WalkSAT is an improvement on GSAT (Selman et al., 1992), a greedy algorithm that starts with a random assignment and flips variables one at a time. The variable chosen for flipping is *always* the one that would cause the greatest immediate reduction in the number of violated clauses. WalkSAT [5] improves on GSAT by using a biased random walk to avoid getting stuck in local minima.

To select the next variable to flip, WalkSAT picks a clause at random from the unsatisfied clauses. With probability p , it flips a variable picked randomly from this clause. With probability $1 - p$ it flips the variable that would cause the *least number of clauses* to be violated when flipped.

3 The WalkPart Algorithm

The WalkPart algorithm is a stochastic local search algorithm for balance-constrained hypergraph bisection. It performs a sequence of moves, until a preset maximum number of moves is reached. A “move” moves a vertex from one partition to another, if balance constraints are not violated. Moves are chosen using criteria similar to WalkSAT: with probability p , a vertex is picked at random, and with probability $1 - p$, a vertex is chosen to minimize some cost function. The selected vertex is then moved to the other partition (if balance constraints allow). After a move, the costs of vertices adjacent to the moved vertex will change, and these are updated.

The move selection procedure is explained in more detail as follows. A net from among the cut nets is chosen at random. With probability p , a vertex is chosen from that net at random, and with probability $1 - p$, the vertex that minimizes the cost function under the decision heuristic in use is chosen from the net. If a chosen move is illegal, the selection process is repeated until a legal move is found.

Due to space constraints, we describe only the decision heuristic that performed best on the benchmarks in this paper, the *normalized node score heuristic*.

Normalized Score Heuristic: The FM heuristic is concerned with the “gain” of a node, determined by the number of nets that can be “cut” or “uncut” by moving the node. Therefore, it only considers “critical” nets that can be cut or uncut in a single move. It is mentioned in [4] that a more thorough measure would also estimate the effort required to uncut the net. We propose a new measure, the “score” of a node, defined as the number of moves required to uncut the net if the node in question was moved. In practice, scores are normalized to avoid bias due to net sizes. We

can quickly approximate the score for a given node for a particular net as follows. Consider a node, n_i , presently in partition j ($j = 0/1$). Let e_k be a hyperedge containing n_i , d_k be the degree of e_k , and $m_{k,j}$ be the number of nodes from edge e_k in partition j , including n_i . Then the normalized score of n_i due to e_k is $\frac{m_{k,j}-1}{e_k}$. The total node score is the sum of the scores due to all nets in which the node occurs. The idea of using a score comes from the work in [6], which discusses an adaptation of WalkSAT to formulas with pseudo-boolean constraints. In WalkPart, with probability $1 - p$ the algorithm picks the node from the chosen net with the minimum score. For the benchmarks in this paper, our empirical results are strongest when this heuristic is used, and when $p = 0.1$.

4 Hybrid Partitioner

Through empirical evaluation, we observe that WalkPart often outperforms a leading-edge implementation of FM [2], on standard industry circuit benchmarks [1]. Also, WalkPart frequently does well on benchmarks where FM does not, implying that both algorithms do not find similar solutions or search in the same areas of the search space. These observations suggest that a hybrid algorithm might be able to exploit the strengths of both.

The hybrid approach alternates runs of FMPart (the FM implementation available in [2]) and WalkPart. It is known that FMPart reduces the net cut greatly in the first few passes, and then spends time performing more passes for relatively small improvements. The hybrid algorithm starts by performing one or more passes of FMPart, and then performs moves of WalkPart to improve FMPart’s solution. Control then switches back to FMPart, and so on. A combined FMPart-WalkPart pass constitutes one pass of the hybrid algorithm. A preset number of hybrid passes are run before the algorithm terminates. The use of a WalkSAT-like randomized component makes it difficult to find obvious convergence criteria. We hope to address this issue in future works. In practice, we observe that the tendency of FM to converge rapidly causes considerable reductions in net cut in a fairly small number of passes. This can be seen from our results in Section 5. Many stochastic local search approaches use such preset limits because of the unpredictable nature of the search.

5 Results

In this section, we discuss the empirical performance of our algorithms. We cannot include the results of all our experiments with different tuning and parameter settings due to limited space. We present results of two relatively simple experiments that illustrate the utility of the hybrid technique. The benchmarks used are standard IBM circuit partitioning benchmarks [1]. Although

Table 2: Results for 10 passes of our Hybrid Algorithm. At each pass, two passes of FM are followed by 50K moves of WalkPart.

Benchmark Name	Nodes	Nets	WalkPart			FMPart		Hybrid		Cut Ratio	Time Ratio
			Moves	Time	Cut	Time	Cut	Time	Cut		
ibm01+10	12752	14111	2M	9.16	610.67	1.54	477.00	1.42	391.66	0.82	0.92
ibm01+4	12752	14111	500K	2.56	506.00	1.39	464.33	1.53	439.67	0.94	1.1
ibm02+10	19601	19584	500K	3.61	329.67	2.15	402.00	1.9	306.33	0.76	0.88
ibm02+4	19601	19584	500K	3.61	654.33	2.14	386.00	2.0	344.67	0.89	0.93
ibm03+10	23136	27401	2M	14.15	1603.33	3.84	1487.67	2.62	1244.33	0.83	0.68
ibm03+4	23136	27401	500K	3.84	1175.00	3.14	2488.00	3.25	1351.33	0.54	1.03
ibm04+10	27507	31970	500K	4.50	1589.00	5.30	718.67	2.9	1131.33	1.57	0.54
ibm04+4	27507	31970	2M	14.94	2121.00	3.52	2505.00	2.76	1071	0.42	0.78
ibm05+10	29347	28446	2M	17.34	4804.67	9.54	2793.33	7.6	2612.33	0.93	0.79
ibm06+10	32498	34826	2M	12.73	750.00	5.63	1683.00	4.1	1160.33	0.68	0.72
ibm07+10	45926	48117	2M	18.07	2483.33	9.63	1347.33	9.3	1687	1.25	0.96
ibm08+10	51309	50513	2M	24.92	1902.67	11.76	3076.00	10.1	1501.33	0.48	0.85
ibm09+10	53395	60902	2M	19.60	2069.00	14.38	1936.00	11.98	1676.33	0.86	0.83
ibm10+10	69429	75196	2M	19.79	3414.67	15.36	2200.33	21.33	1520.66	0.69	1.38
ibm11+10	70558	81454	2M	21.61	2433.33	19.78	4203.33	26.67	4072.33	0.96	1.34
ibm12+10	71076	77240	2M	22.99	3803.67	23.03	4200.67	30.01	3781	0.90	1.30
Average										0.84	0.93

the benchmarks are chosen from the VLSI domain, all the algorithms discussed here are general and can be used on instances from any application. In the experiments, WalkPart uses the normalized score heuristic, and randomness quotient $p = 0.1$.² We use the FM partitioning package by Caldwell et. al., available at [2], for the FM component. WalkPart is written in C++.

In the first experiment, we run 10 hybrid passes after first running FM once to completion. Each hybrid pass performs 100,000 WalkPart moves followed by a single FMPart pass. The solution after a pass is used as the starting solution for the next pass. Results for this experiment are shown in Table 1.

The hybrid algorithm is compared against FMPart and WalkPart, run individually. WalkPart uses the normalized score heuristic and $p = 0.1$, with two different move limits: 500,000 and 2,000,000. Because some moves are picked randomly, increasing the move limit does not always produce better solutions. We show results for the move limit where net cut was lower. For each benchmark we report average solution quality and runtime over three independent starts on a 1.2 GHz AMD Athlon processor with 1-2GB, running Linux.

In the table, columns 1-3 describe benchmark statistics. The benchmarks are named *ibmno+t%*, where *ibm* indicates an IBM benchmark, *no* is the serial number, *t* indicates the balance constraint. Tolerances specify the maximum legal disparity between subset weights as a percentage of total vertex weight. The next three columns give WalkPart’s number of moves, runtime and resulting net cut. The next four columns show runtimes and net cuts for FM and the hybrid algorithm. The lowest net cut out of the 3 algorithms is boldfaced. The last two columns show the ratio of net cut and runtime of the hybrid algorithm to net cut and runtime of FMPart. Averages are shown at the bottom.

We observe that the hybrid algorithm achieves smaller cuts than both FMPart and WalkPart on most benchmarks. This is to be expected, since hybrid passes are run on a solution produced by FMPart. Nevertheless, we can make some important observations. First, FMPart frequently converges when a much better solution is achievable. Second, it is possible to achieve a better solution by introducing a randomized component with a different decision heuristic (normalized score). WalkPart can be used to lead FMPart out of local minima by performing random moves, after which FMPart can be run again to find a better solution. However, this experiment has a runtime cost. On average, this strategy improves run time by 30% over FMPart, but is 1.7 times slower.

Our second experiment improves runtime by avoiding the ini-

²Parameter settings were not chosen arbitrarily. After testing, we chose the values that gave best overall performance.

tial run of FMPart. This experiment also uses 10 hybrid passes. Each pass consists of 2 passes of FMPart followed by 50,000 moves of WalkPart. Results for this experiment are in Table 3.

It is clear that the second hybrid configuration is competitive with FMPart and WalkPart in terms of runtime, and frequently produces better solutions. Also, while either FMPart or WalkPart may achieve lower net cut than the hybrid algorithm, it is very rare that both of them do so. The hybrid approach is usually close to the best solution achieved by either. On average, the second hybrid configuration improves net cut by 15% and is about 7% faster than FMPart alone.

6 Conclusions and Future Work

In this work, we introduce our “crossover” hypergraph partitioning algorithm, WalkPart. We combine WalkPart with the well-known Fiduccia-Mattheyses (FM) heuristic for hypergraph partitioning to produce a hybrid heuristic that draws on the strengths of both techniques. We achieve substantial improvement over FMPart with a very simple approach that requires almost no tuning.

Our first experiment shows that net cuts achieved by FM, a widely used algorithm for this application, can be significantly improved (by 30%), albeit with a runtime cost. However, our second experiment reduces net cuts by an average of 15%, and runtime by 7% over FM. Our results show that combining these two heuristics results in a search procedure that is more powerful than either heuristic used alone. Strong empirical results for the proposed hybrid algorithm motivate further research. We are incorporating our flat partitioning algorithms into the multilevel framework [4, 2] and plan to make it available for use in CAD tools. We are also continuing our investigation of effective search procedures for partitioning and looking for new techniques that can be hybridized with existing ones.

References

- [1] C. J. Alpert, “The ISPD98 Circuit Benchmark Suite”, *Proc. ISPD 1997*, pp. 80-85
- [2] A. E. Caldwell, A. B. Kahng and I. L. Markov, UCLA Physical Design Tools Release: <http://nexus6.cs.ucla.edu/software/PDtools/tar.gz/>
- [3] C. M. Fiduccia and R. M. Mattheyses, “A linear time heuristic for improving network partitions”, *Proc. ACM/IEEE Design Automation Conference*, 1982, pp. 175–181.
- [4] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, “Multilevel Hypergraph Partitioning: Applications in VLSI Design”, in *Proc. ACM/IEEE Design Automation Conference*, 1997, pp. 526-529.
- [5] B. Selman, H. Kautz, and B. Cohen, “Noise strategies for improving local search”, *AAAI ’94*, pp. 337–343.
- [6] J. P. Walser, “Solving Linear Pseudo-Boolean Constraint Problems with Local Search”, *AAAI ’97*.

POSTER PAPERS

VISION AND ROBOTICS

A New Content Based Image Retrieval Method Based on a Sketch-Driven Interpretation of Line Segments

Marco Anelli² and Alessandro Micarelli^{1,2} and Enver Sanginet¹

1. Centro di Ricerca in Matematica Pura e Applicata (CRMPA), Sezione “Roma Tre”.
Via della Vasca Navale 79, 00146 Roma, Italia. sanginet@dia.uniroma3.it
2. Dipartimento di Informatica e Automazione, AI Lab, Università degli Studi “Roma Tre”.
Via della Vasca Navale 79, 00146 Roma, Italia. {anelli,micarel}@dia.uniroma3.it

Abstract

We present a new method for image retrieval by shape similarity able to deal with real images with not uniform background and possible touching/occluding objects. First of all we perform a sketch-driven segmentation of the scene by means of a Deformation Tolerant version of the Generalized Hough Transform (DTGHT). Using the DTGHT we select in the image some candidate segments to be matched with the user sketch. The candidate segments are then matched with the sketch checking the consistency of the corresponding shapes.

1 Motivations and Goals

In this paper we present a new method for Content Based Image Retrieval (CBIR) based on the analysis of the object shapes. In [Anelli *et al.*, 2002] we have proposed a CBIR technique based on a Deformation Tolerant version of the well-known Generalized Hough Transform (DTGHT). In this paper we start from that work and we go further with the objective to augment the system precision. We introduce some segmentation rules corresponding to Gestalt principles (see, for example, [Ullman, 1996] and [Koffka, 1935]) exploiting the continuity properties of the edge segments and a more accurate matching method which compares the searched shape with the image lines. We have obtained an acceptance/rejection rule able to classify a data base image as *relevant* or *not relevant* with respect to the user’s query.

The novelties of our approach with respect to similar systems is its capability to deal with every kind of real images, without caring of light conditions, with not uniform backgrounds and possible occluding objects.

2 Pre-Processing and Sketch Localization

Each image of the system data base is off-line pre-processed in order to extract an edge map and to reduce the noise. We perform a standard edge extraction and thinning process using the Canny edge detector with Sobel 3×3 masks [Shapiro and Stockman, 2001]. After this standard pre-processing, we use two *saliency filters* to erase those thick textured areas which deteriorate the retrieval process (for details, see [Anelli *et al.*,

2002]). From now on we call I the image’s edge map after these pre-processing phases.

We are now interested in merging the pixels in I with their neighbors in order to build line segments. We define a *segment* s as an ordered list of points belonging to I : $s = \langle p_1, p_2, \dots, p_{n_s} \rangle$ such that each p_i ($i = 1, \dots, n_s$) is adjacent to p_{i-1} and p_{i+1} in a 8-connected interpretation of pixel adjacency. Moreover, p_1 and p_{n_s} (the segment end-points) can alternatively be such that: (1) they have only one adjacent point; or (2) they are a junction with another segment, i. e., they have more than 2 adjacent points. All the other points p_i ($i = 2, \dots, n_s - 1$) have exactly 2 adjacent points. This segmentation process is performed because pixels that are part of the same continuous segment are more likely to belong to the same object.

On-line, the system accepts a user drawn query representing the shape she/he is interested to find in the data base. The sketch S is processed in order to extract the R-Table T [Ballard, 1981]. We use the following algorithm:

Sketch Representation Construction(S)

1 Compute the centroid p_r of S .

2 For each point p_k ($k = 1, \dots, m$) of S , set:

$$T[k] := p_r - p_k$$

where $m = |S|$ (m is the cardinality of S).

Once the R-Table is computed, we compare such sketch representation with the previously processed data base images with the aim to localize a region in each image which maximizes the likelihood of the presence of a shape similar to the sketch. This phase is important because we do not manually select in the images the interesting objects from their background like the most working CBIR systems presently do. We adopt a Deformation Tolerant version of the Generalized Hough Transform (DTGHT) [Anelli *et al.*, 2002].

The main difference with the Generalized Hough Transform (GHT)[Ballard, 1981] concerns the voting phase. Here we increment all those accumulator’s positions belonging to a bi-dimensional squared range centered in p_r (called the *voting window* $W(p_r)$), for each edge point p of I and each vector $v = p_r - p$ in T . This is efficiently achieved using a dynamic programming technique and splitting the voting process in two sequential phases: the *exact voting phase* and the *spreading phase* (for details, see [Anelli *et al.*, 2002]). If M is the point with the highest score in the final Hough accumulator (A) after the spreading phase, then we take it as the

most likely position for the center of mass of a possible shape similar to S in I . In other words, if such a shape exists in I , M is its most likely centroid. Experimental results presented in [Anelli *et al.*, 2002] confirm this assumption. Nevertheless, differently from the original GHT, in which the score of the accumulator in its maximum is *equivalent to the number of points of S in I* , with a DTGHT approach, due to the augmented voting area, it is sometimes possible that a thick and random distribution of points and segments in I can produce a high value $A[M]$ not corresponding to a shape really similar to S . In order to reject such cases, in the next section we perform some tests on the segments distribution around M .

3 Verification Phase

In the following we assume that M is the point in I found as described in Section 2, $m = |S|$ and W is the voting window. The aim of the verification phase we are going to explain in this section is to verify if the disposition of the segments in I around M is really perceptually similar to the S' shape. In order to avoid false positives, first of all we check the *portion* of S which can be matched with segments in I . This is intuitively done by projecting S on I (let us call S' such projection, with M its centroid), looking for segments on I in the neighborhood of S' and marking the portion of S *justified* by these segments. Furthermore, we need to take into account *inhibitory* segments intersecting the neighborhood of S' but not really following the shape S . In the following we give the operational definitions for justifying and inhibitory segments.

A point p belonging to I is called a *valid* point if:

$$\exists i \in [1, m] : p + T[i] \in W(M) \wedge |\phi_I(p) - \phi_S(i)| \leq \alpha, \quad (1)$$

where α is a suitable threshold and $\phi_I(p)$, $\phi_S(i)$ are, respectively, the edge orientation of the point p in I and the edge orientation of the i -th point of S . We call i a *valid hypothesis* for p . A segment s_i is a *justifying* segment if:

$$|V_i| \geq k_1 |s_i|, \quad (2)$$

where k_1 is a fixed threshold (presently $k_1 = 0.7$) and V_i is the set of all the valid points of the segment s_i . Let J be the subset of I 's segments which are justifying segments.

Finally, a segment s_i is called an *inhibitory* segment if s_i is not a justifying segment and:

$$k_3 |s_i| \geq |F_i| \geq k_2 |s_i|, \quad (3)$$

where k_3 and k_2 are two fixed thresholds (presently $k_2 = 0.2$, $k_3 = 0.4$) and F_i is the set of all the points p in s_i which satisfy:

$$\exists i \in [1, m] : p + T[i] \in W(M). \quad (4)$$

Let H be the subset of all the inhibitory segments of I . The verification test is given by the logical and of the justification test and the inhibitory test.

Justification Test(J)

- 1 For $k \in [1, m]$, set the local boolean vector $B[k] := false$.
- 2 For each $s_i \in J$ do:
- 3 For each point $p \in V_i$ do:
- 4 For each valid hypothesis k , set $B[k] := true$.
- 5 Return: $|\{B[k] : B[k] = true\}| \geq k_4 m$

Table 1: Precision and recall scores for the three users. The columns show, for each user, the precision and recall values obtained for each query type.

Query	User 1		User 2		User 3		Average	
Car	0.94	0.65	0.81	1	0.91	0.81	0.88	0.82
Pistol	1	0.70	0.93	0.70	0.86	0.60	0.93	0.74
Racket	0.90	0.50	0.73	0.85	1	0.95	0.86	0.76
Watch	0.28	0.10	0.66	0.30	0.75	0.15	0.54	0.18
Bottle	0.60	0.45	0.75	0.60	0.93	0.7	0.76	0.58
Guitar	0.93	0.75	0.94	0.80	0.66	0.50	0.85	0.68

In the above algorithm k_4 is a threshold, currently fixed to 0.7. We check in this way that at least a portion k_4 of S is present in I in the right position with respect to M .

Finally, the following is the inhibitory test:

$$\text{Inhibitory Test}(H, J) := \sum_{s_i \in H} |s_i| < \sum_{s_i \in J} |s_i|.$$

4 Experimental Results

We have tested our method with a data base composed of 283 images randomly taken from the Web. They show a great variety of subjects: 26 cars, 22 horses, 20 guitars, 20 pistols, 20 tennis rackets, 20 saxophones, 20 bottles, 20 watches, 15 crucifixes, 10 vases. Moreover, the data base contains different other subjects such as: animals, faces, landscapes, airplanes, boats, fruits, mushrooms, cups and so on.

No simplifying assumption has been done about images: they usually have not uniform backgrounds and often the retrieved objects are occluded by other objects. No manual segmentation has been performed on the images.

We paid attention to choose as users people not aware of the data base images' shapes. Indeed, only one of the three users in Table 1 has had the possibility to observe the repository images before his drawings (User 3). Each user has performed six queries. The drawn subjects are: a car, a pistol, a tennis racket, a watch, a bottle and a guitar. We report our results in Table 1. Average precision and recall for all the three users and all the queries are, respectively: 0.80 and 0.62.

References

- [Anelli *et al.*, 2002] M. Anelli, A. Micarelli, and E. Sanginetto. A deformation tolerant version of the generalized hough transform for image retrieval. In *Fifteenth European Conference on Artificial Intelligence (ECAI 2002)*, Lyon, France, 2002.
- [Ballard, 1981] D. H. Ballard. Generalizing the Hough Transform to detect arbitrary shapes. *Pattern Recognition*, 13, No. 2:111–122, 1981.
- [Koffka, 1935] K. Koffka. *Principles of Gestalt Psychology*. New York: Harcourt, Brace and World, 1935.
- [Shapiro and Stockman, 2001] L. Shapiro and G. Stockman. *Computer Vision*. Prentice hall, 2001.
- [Ullman, 1996] S. Ullman. *High-level Vision. Object Recognition and Visual Cognition*. A Bradford Book. The MIT Press Cambridge, Massachusetts, 1996.

Towards Pervasive Robotics

Artur M. Arsenio

Artificial Intelligence Lab - Massachusetts Institute of Technology
545 Technology Square, Room NE43-936, MA 02139
arsenio@ai.mit.edu

Abstract

Pervasive robotics will require, in a near future, small, light and cheap robots that exhibit complex behaviors. These demands led to the development of the M2-M4 Macaco project - a robotic active vision head. Macaco is a portable system, capable of emulating the head of different creatures both aesthetically and functionally. It integrates mechanisms for social interactions, autonomous navigation and object analysis.

1 Motivation

One approach of AI is the development of robots whose embodiment and situatedness in the world evoke behaviors that obviate constant human supervision [Brooks, 1999]. With this in mind, we developed the M2-M4 Macaco project, which is described in this paper. M2-M4 Macaco is a robotic active, modular and compact system. This creature was designed to fit different mobile robot platforms or act as a stand-alone system. Another design goal was the portability of both the mechanical and electronic devices and its brain. Macaco characteristics make it a portable, fully operational robotic head whenever not assembled to a mobile platform, able to act as a social agent. A simple communications interface enables operation onboard mobile platforms, turning the robot into an autonomous, sociable machine.

Research robots are enclosed most of the time at lab facilities in which they are developed, most often operating just for demonstration goals. We expect in a near future to have both the robotic head and its brain physically present at exhibitions/seminars, interacting socially. This new approach with complex, portable research robots will lead to commercial applications and increasing synergy among roboticists. Eventually, pervasive robotics - robots present everywhere to perform a variety of tasks - will be possible as smaller, lighter and cheaper robots become available.

1.1 The Robotic creature

This robotic mechanism was designed to resemble a biological creature, exploiting several features of an evolutionary design, but adding others (such as a thermal camera for human detection and night vision) for improved performance.



Figure 1: (Left) The M4 Macaco robot, designed to resemble a dog's head. (Right) The M4-Macaco robotic head and processing hardware assembled to a *Magellan* mobile platform.

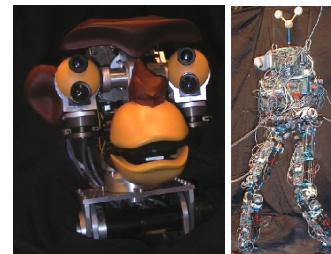


Figure 2: (Left) M2-Macaco, a biological inspired robotic head, designed to resemble a chimpanzee head. (Right) M2 robotic body built by the MIT LegLab.

The replacement of a few M2-M4 Macaco's aesthetic components allows for the metamorphosis of a dog-like (M4 - see Figure 1) into a chimpanzee-like (M2 - see Figure 2) robot. The weight of the head, including motors, gears and gyro, is $\sim 1.6Kg$. The hardware consists of nine small CPU boards with Pentium III at 800MHz, all modules connected by an Ethernet network. Four cameras and a total of nine framegrabbers are used for video acquisition.

2 Implementation

Security is one possible operational scenario for this active head. For this class of applications, Macaco robot was equipped with a behavioral system capable of searching for people or faces, and to further recognize them. In addition, human gaze direction might reveal security threats, and thus a head gaze detection algorithm was developed. Probable targets for such gazings are other people and mostly important, explosives and/or guns. Therefore, salient objects situated in

the world are processed for 3D information extraction and texture/color analysis. Current work is also underway for object and scene recognition from contextual cues.

Another scenario includes search and rescue missions by a mobile robot, which requires additional navigation capabilities in rough terrain. Finally, real world applications are often characterized by strong light variations or the absence of light. This is taken into account through thermal image processing for people detection and for night navigation.

2.1 Visual Pre-Attentive System

Although the real world does not avail precise or singularly adequate perceptual information, unique interpretations of the world can be constructed using an attentional mechanism. A logpolar attentional system was developed to select relevant information from the cameras output, and to combine it in a saliency map (see Figure 3). This map is segmented into three regions of stimuli saliency - the attentional focus.

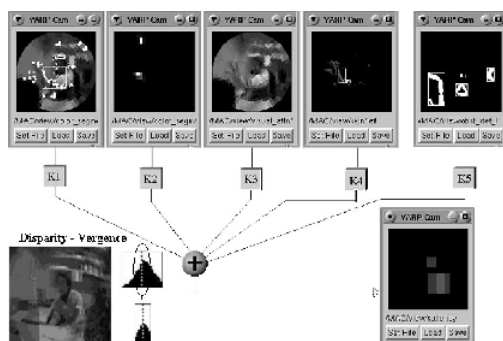


Figure 3: The basic feature maps (Color Processing, Skin Detection, Optical Flow and Edge Detection) are weighted and summed, yielding the saliency map. The focus of attention is obtained from segmenting this map.

Macaco's robotic eyes are equipped with a set of basic movements made by frontal eyed, foveal animals: Ballistic movements are executed without visual feedback; Vergence to a target is maintained by a disparity signal from stereo images; and Vestibulo-ocular-reflex (VOR) stabilizes the cameras using data from the inertial sensor.

2.2 Post-Attentional Vision

The brain for the M2-M4 Macaco robotic head consists of a flexible, modular and highly interconnected architecture that integrates social interaction, object analysis and functional navigation modules, as shown in Figure 4 - the post-attentional modules are described in detail in [Arsenio, 2003],

- Object Analysis: Texture and Color Segmentation algorithms run in parallel, and are integrated with 3D object reconstruction to obtain a rendered object model.
- Social Mechanisms: For the robot to achieve a convincing social role, the vision system is equipped with face detection and recognition modules, together with an algorithm for the detection of human gaze direction.
- Navigation: Although 3D information is lost with low visibility, the platform is still able to operate thanks to

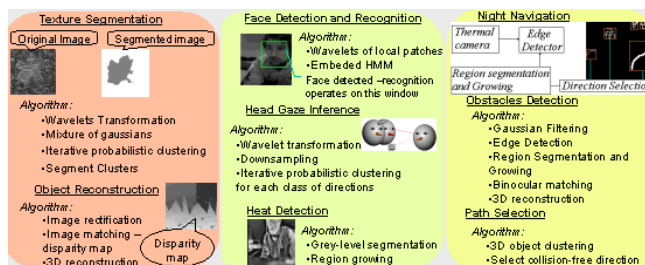


Figure 4: (Left) Object Analysis. (Center) Social and (Right) Navigation Mechanisms.

a thermal camera. A navigation algorithm based on monocular cues runs at frame-rate, for night navigation.

2.3 Architecture

The software architecture includes, besides the Visual Attention system, releasers from body sensors and motivational drives that modulate attentional gains. Action is determined by competing behaviors, which also share resources to achieve multi-behavior tasking.

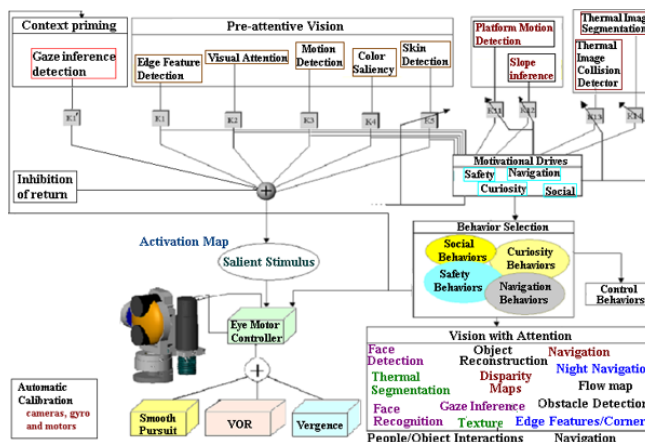


Figure 5: M2-M4 Macaco software architecture.

3 Conclusions

We presented the project of a portable robotic head, modular at the mechanical, hardware and software levels. Although equipped with a complex cognitive system, its small weight and compact size allows it to be incorporated into multiple mobile platforms, and also to be used as a portable autonomous sociable robotic creature.

Acknowledgments

Work funded by DARPA project "Natural Tasking of Robots Based on Human Interaction Clues", contract DABT 63-00-C-10102. Author supported by Portuguese grant PRAXIS XXI BD/15851/98.

References

- [Brooks, 1999] Rodney A. Brooks. Cambrian Intelligence. MIT Press, 1999.
- [Arsenio, 2003] Artur M. Arsenio. A Robot in a Box. Accepted to IEEE 2003 Int. Conf. on Advanced Robotics.

A Visual-Sensor Model for Mobile Robot Localisation

Matthias Fichtner Axel Großmann

Department of Computer Science
Technische Universität Dresden
Dresden, Germany

1 Introduction

Due to recent advances in robot hardware, there is a great demand for vision-based robot localisation techniques [DeSouza and Kak, 2002]. We present a probabilistic sensor model for camera-pose estimation in hallways and other known structured environments. Given a 3D geometrical map of the environment, we want to find an approximate measure of the probability that a given camera image has been obtained at a certain place in the robot's operating environment. Our sensor model is based on feature matching techniques that are simpler than state-of-the-art photogrammetric approaches. This allows the model to be used in probabilistic robot localisation methods, such as Monte Carlo localisation (MCL) [Dellaert *et al.*, 1999]. We have combined photogrammetric techniques for feature projection with the flexibility and robustness of MCL. Moreover, our approach is sufficiently fast to allow for sensor fusion. That is, by using distance measurements from sonars and laser in addition to the visual input, we may be able to improve localisation accuracy. We have used our sensor model with MCL to track the position of a Pioneer 2 robot navigating in a hallway. Possibly, our approach can be used also for localisation in cluttered environments and for shape-based object detection.

In the next section, we briefly describe the components of the visual-sensor model. We conclude with a discussion of experimental results.

2 Sensor Model

A visual-sensor model describes the probability of obtaining a particular camera image given the camera's pose and a geometrical map of the environment. Rather than comparing camera image and the expected view at the pixel level, we gained improved robustness using image features. We decided to use line segments because they can be detected comparatively reliably under changing illumination conditions. As world model, we use a wire-frame model of the operating environment, represented in VRML. The individual processing steps are depicted in Figure 1.

Representation. Each straight-line feature is represented as a single point (ρ, θ) in the 2D Hough space given by $\rho = x \cos \theta + y \sin \theta$, where end-points are neglected. In this representation, truncated or split lines have similar coordinates.

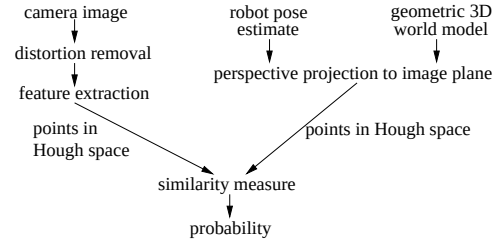


Figure 1: Processing steps of the visual-sensor model.

Similarity measures. The correspondence of image and model features is evaluated by a similarity measure. In general, such a measure may take into account the differences in orientation between corresponding line segments in image and model, or their distance and difference in length. In the following, we present two simple and efficient similarity measures. The first is solely based on the distance of line segments in the Hough space. We consider only those image features as possible matches that lie within a rectangular cell in the Hough space centred at the model feature. The matches are counted and the resulting sum is normalised. The mapping from the expectation (model features) to the measurement (image features) accounts for the fact that the measure should be invariant with respect to objects not modelled in the provided map or unexpected changes in the operating environment. Invariance of the number of visible features is obtained by normalisation. Specifically, this *centred match count* (CMC) measure s_{CMC} is defined as:

$$s_{CMC} = (1/|H_e|) \#_{i=1}^{|H_e|} \min \left(1, \#_{j=1}^{|H_m|} m(h_{e_i}, h_{m_j}) \right)$$

$$m(a, b) \equiv |\rho_a - \rho_b| < t_\rho \wedge \|\theta_a - \theta_b\| < t_\theta$$

where the predicate m defines a valid match using the distance parameters (t_ρ, t_θ) and the operator $\#$ counts the number of matches. Generally speaking, this similarity measure computes the proportion of expected model Hough points $h_{e_i} \in H_e$ that are confirmed by at least one measured image Hough point $h_{m_j} \in H_m$ falling within tolerance (t_ρ, t_θ) . Note that neither endpoint coordinates nor lengths are considered here. The second measure is based on a comparison of the total length values of groups of lines, which we name *grid length match* (GLM). Split lines in the image are grouped together using a uniform discretisation of the Hough space. This method is similar to the Hough transform for straight

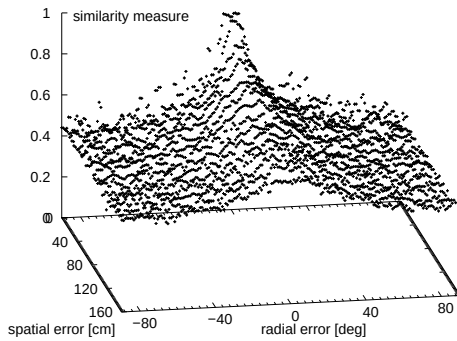


Figure 2: Performance of CMC on artificially created images.

lines. The same is performed for line segments obtained from the 3D model. Again, the mapping is directional, i.e., the world model is used as reference, in order to obtain invariance of noise, clutter, and dynamic objects.

Obtaining probabilities. Ideally, we want a monotonically decreasing similarity measure as the pose estimate of the hypothesis departs from the actual camera pose. By averaging over a large number of different, independent situations, an abstraction from specific poses and environmental conditions is gained. To obtain commensurable values, we express the model in terms of relative robot coordinates instead of absolute world coordinates. The probability returned by the visual-sensor model is obtained from the similarity value by normalisation within the range of the pose deviation.

3 Experimental Results

We have evaluated the proposed sensor model and similarity measures in a series of experiments.

Using artificially generated images. First, we simulated the feature extraction steps from real images by duplicating the right-hand branch of Figure 1. By introducing a pose deviation Δl systematically, we investigated its influence on the similarity values. For visualisation purposes, Δx and Δy were combined into a spatial deviation Δt . For the measures CMC and GLM, over 15 million random camera poses were coupled with a random pose deviation. As shown in Figure 2, the simulation of CMC shows a unique, distinctive peak at zero deviation with monotonically decreasing similarity values as the deviation increases. For GLM, we obtained a peak that is considerably more distinctive. This conforms to our expectations since taking the length of image and model lines into account penalises incidental false matches.

Toward more realistic conditions. In order to learn the effect of distorted and noisy image data on our sensor model, we repeated the experiments above but applied distortions to the synthetically generated features before matching. The error model comprised random splitting, duplication and shifting, and addition of lines yielding typical, noisy feature maps. The results obtained for the two measures did not differ significantly from the first set of experiments.

Using real images from the hallway. As the results obtained in simulation might be questionable with respect to real-world conditions, we conducted another set of experi-

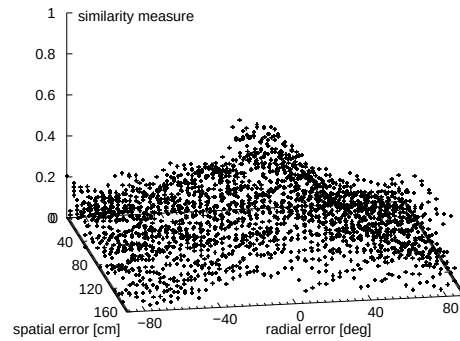


Figure 3: Performance of GLM on real images.

ments using real images. We gathered images with a Pioneer 2 robot in the hallway off-line and recorded the line features. More than 3200 images were taken from 64 different (x, y) locations. Likewise as above, the sensor model was computed for pairs of poses. Results referring to the same discretized pose deviation were averaged. The result of employing the GLM measure is given in Figure 3. Again, GLM yielded a more distinctive peak compared to the curve obtained using CMC.

Application to Monte Carlo localisation. The generic interface of the sensor model allows it to be used in the correction step of Bayesian localisation methods, for example, the standard version of the MCL algorithm. In this experiment, the robot had to follow a pre-programmed route in the shape of a double loop in the corridor. On its way, it had to stop at pre-defined positions, turn to a nearby corner or open view, take an image, turn back and proceed. Upon each image capture, the weights of all samples were computed according to the visual-sensor model using CMC. The preliminary results are very promising. The best hypothesis for the robot's pose was approximately at the correct pose most of the time. We noticed that rotational error has a strong influence on the degree of coincidental feature pairs. This effect corresponds to the results above, where the figures exhibit a much higher gradient along the axis of rotational deviation than along that of translational deviation. This finding can be explained by the effect of motion on features in the Hough space. Hence, the strength of our model lays at detecting rotational disagreement. This property makes it especially suitable for two-wheel driven robots like our Pioneer bearing a much higher rotational odometry error than translational error.

References

- [Dellaert *et al.*, 1999] F. Dellaert, W. Burgard, et al. Using the condensation algorithm for robust, vision-based mobile robot localisation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 1999.
- [DeSouza and Kak, 2002] G. N. DeSouza and A. C. Kak. Vision for mobile robot navigation: A survey. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(2):237–267, 2002.

Improving Speech Recognition on a Mobile Robot Platform through the use of Top-Down Visual Queues

Robert J. Ross, R.P.S. O'Donoghue and G.M.P. O'Hare

Abstract

In many real-world environments, Automatic Speech Recognition (ASR) technologies fail to provide adequate performance for applications such as human robot dialog. Despite substantial evidence that speech recognition in humans is performed in a top-down as well as bottom-up manner, ASR systems typically fail to capitalize on this, instead relying on a purely statistical, bottom up methodology. In this paper we advocate the use of a knowledge based approach to improving ASR in domains such as mobile robotics. A simple implementation is presented, which uses the visual recognition of objects in a robot's environment to increase the probability that words and sentences related to these objects will be recognized.

1 Motivation & Proposal

Through the mapping of an acoustic signal to a string of words ASR systems are a key tool in the control of mobile devices such as robots and PDAs, particularly in cases where manual control is not appropriate or feasible. However despite the substantial improvements in ASR reliability which have been made in the last ten years, results can still be very poor in noisy environments. Most ASR systems are built around a statistical, data driven architecture which fails to capitalize on sources of information other than the input audio signal and static vocabulary. Many hybrid architectures utilizing lip-reading algorithms have emerged in recent years [Chibelushi *et al.*, 2002], but these are dependent on a user directly facing a communication interface.

ASR is becoming an increasingly more important tool in the development of service and mobile robots. This can generally be accredited to the ease of use a natural language interface should bring to human-machine interactions. However in practice the speech recognition systems available fail to produce sufficient accuracy for natural interactions. These failings result from a wide range of environmental distortions and noise, including a) interference from the robot's drive systems, b) reverberations c) multiple user interference (the so-called cocktail party effect).

Users of ASR in the robotics community have generally taken two approaches to counteracting sources of interfer-

ence. On a practical level the simplistic use of hand-held or head-mounted microphones can dramatically improve performance. Another approach uses dialog systems based on finite state grammar models or frame systems to anticipate user utterances from dialog constructs [Matsui, 1999; Bischoff, 2001]. Although these approaches do offer a substantial improvement on a purely black-box approach to ASR, they impose tight constraints on the usability and flexibility of the system as a whole.

Naturally the partial failure of ASR technology in a noisy environment does not only pose problems for the mobile robotics community. Problems with mono-modal speech recognition systems have been recognized for many years, and many attempts have been made to improve recognition quality through the use of low-level visual information. In [Chibelushi *et al.*, 2002] Chibelushi, Deravi and Mason present a comprehensive review of these approaches. Algorithms providing sources of visual information center mostly on those which perform lip-reading or the analysis of facial gesture. The techniques discussed in the review paper can be characterized as being low-level or data driven in nature, with a division being drawn on whether information should be combined before integration into a model, or whether two separate low-level models should first be formed before a general abstraction is made.

The trend towards integrating low level visual and audio information reflects clear evidence of this phenomenon in humans. The McGurk effect is perhaps the best known example of this, where the audio presentation of the sound 'BA' along with the visual lip movements of the sound 'GA' typically results in a listener perceiving the sound 'DA' [McGurk and MacDonald, 1976]. However there is also a growing body of evidence that indicates that humans use high level context and semantic effects to improve our speech recognition performance [Tanenhaus *et al.*, 1995; Simpson, 1994]. Further more it is commonly observed that in conversation we will often reference particular themes, and discuss objects in his or her local environment.

Inspired by this evidence of context and high level semantic priming of speech recognition in humans, we propose the improvement of ASR systems on mobile robots through the use of a top down context priming model, rather than relying on workarounds based on strict dialog systems or user held microphones. The initial model described below is based on

the premise that users commonly discuss objects in their local environment. Specifically our model proposes that upon the visual recognition of an object in the environment, the probability of recognition of words associated with that object should be increased. This is achieved through direct communication between software agents responsible for visual processing and speech recognition. Although such a model is simplistic it acts as a stepping stone towards the further improvement of ASR by context effects.

2 Implementation

The research presented here has been conducted as part of the SAID (Speaking Autonomous Intelligent Devices) Project in University College Dublin [Ross *et al.*, 2002]. The focus of this project is to examine how the two very different computational paradigms of speech recognition and autonomous agents can benefit from each other in the production of intelligent speech enabled devices. The model developed here has therefore been constructed for a complete mobile robot platform, rather than having been implemented as a stand-alone algorithm.

Although the model presented has been developed for a mobile robot with a vision source, the basic principle is not hard-wired to the mobile robot domain. The principle can be applied anywhere where there is a source of high level information which can be used to dynamically prime a speech recognition systems. Specifically the Speech Priming agent discussed below can be applied on any mobile device which has access to spatial knowledge of the outside world (whether acquired through passive visual detection or not).

2.1 Platform

Experiments are carried out using a team of Nomadic Scout II robots, refitted with on-board computers, vision, and sound systems. The control system for the robot is provided through MARC, an experimental Multi-Agent System based architecture [Ross, 2002]. This architecture like many social robot architectures views individual robots as intelligent agents in a social community. However MARC is original in that all aspects of any one robot's control, from high level planning and user modeling to low level movement and reactive control are encapsulated through a community of intentional deliberative agents (See [Wooldridge, 2000] for a good introduction to the theoretical models, and [Collier, 2001] for details of a concrete development and runtime environment for these agents.). Although the robot control architecture contains a large number of diverse agents, discussion here will be limited to those agents specifically connected with speech recognition with visual priming.

2.2 The Agents

The speech recognition and visual priming system consists of a number of agents which provide a) basic speech recognition, b) passive visual object recognition and c) speech recognition priming through the use of a simple semantic model. Despite the fact that these individual agents are internally highly data driven, coupling between the agents is loose with all inter-agent communication via the agent communication

language Teanga [Rooney, 2001]. The internal design of the three main agent types will now be outlined, followed by a discussion of their typical interactions and usage.

Low Level Speech Recognition Agents - These agents work at the level of traditional ASR systems to convert audio signal from the outside world to simple strings of text or n-grams. A number of low level speech agents have been built using a range of speech recognition toolkits. Toolkits employed included both well-known systems like Sphinx and ViaVoice, as well as a less popular but potentially more powerful and open hybrid systems [Carson-Berndsen and Walsh, 2000].

A problem encountered in using the more popular speech recognition toolkits, is that most operate using either a BNF grammar of predicted sentences, or in a completely free-form dictation mode. A middle ground where the adjustment of probabilities on words and sentences was not achieved easily.

Visual Object Recognition Agent - Each robot has on board color CCD camera. This camera is controlled by low level software and drivers to provide video footage of the outside world. A Visual object recognition agent was built which employs color segmentation and edge-based feature detection algorithms to scan for key objects in the robot's field of view. The visual recognition agent passively scans the environment for pre-defined objects such as chairs, colored balls, and large office features. Upon detection of any such object this visual detection agent communicates its observation to any agents which have previously requested to be kept informed of its findings.

Speech Recognition Priming Agent - The third agent type acts as the key filtering mechanism between the visual recognition of objects and the adjustment of word recognition probabilities in the lower-level speech recognition agents.

The agent employs a semantic network type structure to produce strings and groups of probable words for recognition based on the priority of identified external objects. Specifically, the network has been prepared with details of objects commonly found in an office environment, along with associated verbs, prepositions and pseudonyms. When informed that there is a *package* in the robots proximity, this agent can provide a list of words and phrases which are likely to be spoken in connection with such a package e.g. deliver, box. The model can be altered to produce recommendations for varying periods and priorities based on relative importance of objects and the robots current actions respectively.

A typical usage scenario involves the Speech Recognition Priming Agent making an initial request with the Visual Object Recognizer for reports on any key objects recognized in the environment. If and when provided with reports of objects in the robots vicinity, the Speech Recognition Priming Agent will build up a revised list of words which it might expect a user to utter. The Speech Recognition Priming Agent will then attempt to inform the speech recognition agent of this revised list. The relevant speech recognition agent can then choose to take note of these values in speech recognition, or ignore them as it sees fit.

Based on the advice from the Speech Priming Agent, the

Speech Recognition Agents can then interpret the incoming sound signals as appropriate, providing strings of recognized text to any agents which have expressed an interest in its findings e.g. social communication agents, command interpreter agents.

2.3 Test Scenarios

Experiments conducted are based on a user and robot situated in a room which has been furnished with a number of objects, some of which the robot is capable of visually recognizing. The user then issues a number of commands to the robot. This set of commands is composed of instructions and questions about objects in the room. The command set includes both well formed commands and a number of garbled commands which are phonetically very similar to the well formed command. These incorrectly formed commands are typical of slight mispronunciations, or environmental distortions. Badly pronounced utterances range from slight mispronunciations of nouns and verbs, to the replacement of one word with a different but proper word as with

Where is the mall

rather than

Where is the ball

In initial tests the utilization of the context priming model clearly produces more reliable results than those produced when the vision system is disjoint from the speech recognition components.

3 Related Research

The integration of audio and vision is becoming more popular in recent years and is being approached from a number of angles. In addition to the low-level integration of audio and visual information as discussed earlier [Chibelushi *et al.*, 2002], related research on the integration of audio and visual information includes Deb Roy's work on the learning of words from sights and sounds [Roy, 1999], and the generation of natural language from a visual representation [Herzog and Wazinski, 1994].

4 Initial Conclusions & Future Work

Although speech recognition systems can often produce inaccurate results in real-world environments, accuracy in the mobile robot domain can be improved through a knowledge based priming of speech systems. The model presented here was based on the premise that users often discuss objects in their local environment. Through the visual recognition of objects ASR systems were 'primed' for specific topics of conversation. Such a technique is novel and should be contrasted with low level bi-modal speech recognition systems.

Although the model presented here is simple, the principle of using high level data sources to improve ASR performance can easily be extended to other data sources and platforms. Taking advantage of user and task domain modeling is an obvious next step. With this in mind, immediate future work includes the expansion of the semantic model, and the addition of general task domains.

Acknowledgements

We gratefully acknowledge the support of Enterprise Ireland through grant No. IF/2001/02, SAID.

References

- [Bischoff, 2001] Rainer Bischoff. Hermes - a humanoid experimental robot for mobile manipulation and exploration services. IEEE International Conference on Robotics and Automation, apr 2001. Video Abstract.
- [Carson-Berndsen and Walsh, 2000] Julie Carson-Berndsen and Michael Walsh. Interpreting multilinear representations of speech. In *Proceedings of the 8th Australian Conference on Speech Science and Technology*, 2000.
- [Chibelushi *et al.*, 2002] C.C. Chibelushi, F. Deravi, and J.S.D. Mason. A review of speech-based bimodal recognition. *IEEE Transactions on Multimedia*, 4(1):23–37, mar 2002.
- [Collier, 2001] Rem W. Collier. *Agent Factory: A Framework for hte Engineering of Agent Oriented Applications*. PhD thesis, University College Dublin, 2001.
- [Herzog and Wazinski, 1994] G. Herzog and P. Wazinski. VIsual TRANslator: linking perceptions and natural language descriptions. *Artificial Intelligence Review*, 8(2-3):175–187, 1994.
- [Matsui, 1999] Toshihiro Matsui. Integrated natural spoken dialogue system of jijo-2 mobile robot for office services. In *AAAI/IAAI*, pages 621–627, 1999.
- [McGurk and MacDonald, 1976] Harry McGurk and John MacDonald. Hearing lips and seeing voices. *Nature*, 264:746–748, dec 1976.
- [Rooney, 2001] C.F.B. Rooney. A formal syntax & semantics for teanga. Technical report, University College Dublin, 2001.
- [Ross *et al.*, 2002] Robert J. Ross, Bryan McEleney, Robert Kelly, Tarek Abu-Amer, Michael Walsh, Julie Carson-Berndsen, and Gregory M.P. O'Hare. Speaking autonomous intelligent devices. In *In Proc. 13th Irish Conference on Artificial Intelligence & Cognitive Science (AICS 02)*, 2002.
- [Ross, 2002] Robert Ross. Marc - the multi-agent robot control architecture. Technical Report, Oct 2002.
- [Roy, 1999] Deb Kumar Roy. *Learning Words from Sights and Sounds: A Computational Model*. PhD thesis, MIT, 1999.
- [Simpson, 1994] Greg B. Simpson. Context and the processing of ambiguous words. In *Handbook of Psycholinguistics.*, chapter 10, pages 359–374. Academic Press, 1994.
- [Tanenhaus *et al.*, 1995] M.K. Tanenhaus, M.J. Spivey-Knowlton, K.M. Eberhard, and J.E. Sedivy. Integration of visual and linguistic information in spoken language comprehension. *Science*, 268:1632–1634, 1995.
- [Wooldridge, 2000] Michael Wooldridge. *Reasoning about Rational Agents*. The MIT Press: Cambridge, MA, USA, 2000.

Comparing image-based localization methods

Robert Sim and Gregory Dudek

Centre for Intelligent Machines, McGill University

{simra,dudek}@cim.mcgill.ca

Abstract

This paper compares alternative approaches to pose estimation using visual cues from the environment. We examine approaches that derive pose estimates from global image properties, such as *principal components analysis* (PCA) versus from local image properties, commonly referred to as *landmarks*. We also consider the failure-modes of the different methods. Our work is validated with experimental results.

1 Introduction

A considerable amount of research has been conducted on the problem of using vision to localize a robot in a known environment. Two basic approaches have emerged; the first correlates global properties of the image, such as a principal components subspace projection; whereas the second correlates a set of local properties, or landmarks. Each approach has its own strengths and weaknesses, and is based on its own assumptions about the environment. It is often argued that local feature-based representations are more robust to scene dynamics and illumination variation than globally-derived representations. It is also widely assumed that feature-based approaches require less online computation time than global approaches.

The goal of this paper is to critically examine a selection of image-based pose estimation methods and compare them for their performance under a variety of conditions. We consider accuracy and running time as the two major indicators of performance.

2 Previous Work

Pose estimation from monocular vision data is non-intrusive and conceptually appealing as a research direction. Position estimation entails a combination of estimating local displacement with recognizing familiar locations. It has been approached as both a coarse place-recognition problem (e.g. based on color histograms for familiar locations [Ulrich and Nourbakhsh, 2000]) and also as a problem of recognizing a set of local features associated with a specific location (for example [Se *et al.*, 2002]). In several approaches, authors have suggested interpolating between familiar locations to produce

a position estimate more accurate than the density of training data, using either global image features ([Nayar *et al.*, 1994; Pourraz and Crowley, 1999]) or sets of local image features ([Sim and Dudek, 2001]). In this paper we compare the performance of some of these alternatives..

3 Localization Methods

Framework We assume the same framework for all of the methods presented below. First, a set of training images has been collected from known poses in the environment. A model, or models, are constructed by building an interpolant over the set of training poses. We employ bilinear interpolation of the observation space over the Delaunay triangulation of the pose space. The cost of constructing the triangulation and any other preprocessing of the training images (such as tracking features) is referred to as the *offline cost*.

When a pose estimate is required, a probability distribution is computed using a multi-resolution grid-based discretization of the pose space. At each grid position a predicted observation is generated and compared with the actual observation. The probability is computed using Bayes' Rule, whereby, assuming a uniform prior, the probability of a pose q is proportional to the probability of the observation z given the pose:

$$p(q|z) \propto p(z|q)$$

Once a cell with maximal probability is determined, a higher resolution discretization is computed in the neighborhood of that cell, and the process recurses until a desired level of resolution is achieved. By setting the maximum discretization to a fixed value, we fix the number of times the likelihood function is evaluated, thus allowing us to benchmark the performance of the method. This running time is referred to as the *online cost*.

3.1 Global Methods

Edge Density Our first globally-based method computes a map of the local edge density of the image. The density is approximated by convolving the image edge map with a wide (33 pixels) Gaussian kernel. The edge map is relatively resistant to illumination changes, and measuring edge density propagates edge information locally. Depending on the width of the kernel, a linear combination of neighboring training edge densities can loosely approximate the motion of edges as the camera moves.

Principal Components PCA operates by computing a low-dimensional subspace of the space spanned by the training images. For this work, we employ the first twenty principal components. When pose likelihoods are computed online, only the low-dimensional projections of the training inputs are interpolated. The input observation must be projected into the subspace; an operation that is performed once; and subsequent comparisons are between low-dimensional vectors.

3.2 Local Methods

The alternative to computing global image features is to extract a set of local image features, or landmarks, and model their behavior as a function of position. Local features must be *selected* and *tracked*, and subsequently they are *modeled*. Finally, when online they must be *matched* to features in the input observation. Once matching is complete, the individual estimates from different landmarks must be combined in a robust manner.

We use the landmark-based method proposed by Sim and Dudek [Sim and Dudek, 2001]. That model selects landmarks using saliency, and, once they are tracked across the training space, models the landmark behaviors as a function of pose. The models compute a linear interpolant of the landmark attributes over the triangulation of the pose space.

We employ two versions of the landmark model; the first, referred to as **DynamicLM** models the appearance of the landmarks as a function of pose, and thus matching may require several instantiations of the landmark before it is successfully matched. The second, referred to as **FixedLM**, fixes the landmark appearance, speeding up the matching, but reducing the range of poses that the landmark can model.

4 Experimental Approach

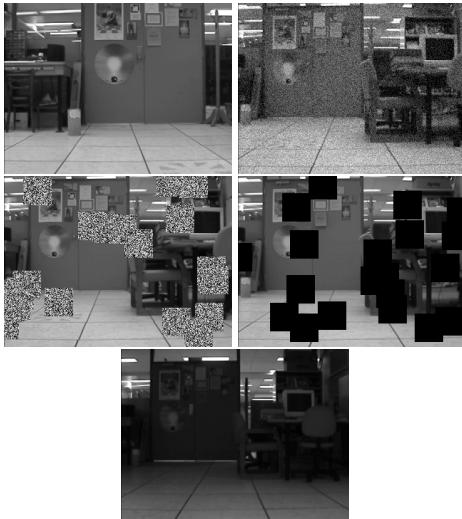


Figure 1: Images from a) the training set (top,left), b) the *Noisy* verification set (top,right), c) the *Occlusion* set with noisy occluders (middle,left), d) the *Occlusion* set with solid occluders (middle,right), and e) the *Illumination* set (bottom).

	Mean Err. (cm)	Max Err. (cm)	Tot. Train'g Time(s)	Online Preproc. (s/img)	Online Loc'zn (s/img)
<i>Global Methods</i>					
Edge Density	8.22	22.4	31.4	0.309	38.1
NN (EdgeDens)	11.1	22.3	n/a	n/a	n/a
PCA	6.06	12.4	2170	0.234	0.208
NN (PCA)	9.7	22.3	n/a	n/a	n/a
<i>Local Methods</i>					
DynamicLM	8.49	23.09	1581	145	19.8
FixedLM	7.61	19.7	1529	10.6	17.9

Table 1: Localization results for *Normal* set.

Training We used a robot to collect 121 images in a grid pattern at 20cm intervals, over a 2m by 2m area (Figure 1a)). Ground truth was obtained by mounting a laser pointer on the robot and measuring by hand the position of the laser point on the floor. As such, the ground truth is accurate to ~0.5cm.

The same set of training images was provided to each localization method for preprocessing and training, and the running times for this phase were recorded.

Verification A set of 29 verification images were collected with the robot from random locations at the same orientation as the training images. Ground truth was measured by hand. The images were collected under the same illumination conditions and observed the same static scene. These images constitute our *Normal* verification set. Gaussian white noise was added to these images to create a *Noisy* verification set (Figure 1a)), and a set of occluders were randomly painted into the images to generate two *Occlusion* sets (Figure 1b) and 1c)). The first set of occluders consists of white noise, and the second consists of solid tiles.

A second set of ten images was collected under low-light conditions from random locations. Ground truth was recorded by hand. These images are referred to as the *Illumination* verification set (Figure 1d)).

Each localization method was applied to each verification set, and the mean error between the maximum-likelihood (ML) estimate and ground truth was compiled on a per-method, per-set basis. The computational cost of preprocessing the input images and computing the ML estimate was also recorded. For each image, the multi-resolution grids were evaluated at a total of 2300 distinct poses.

Measuring performance Consider the expected error using a method that selects the nearest training image with 100% accuracy. This *magic number* can be computed to be 7.6cm for our experiments. A mean error of less than 7.6cm could be considered to be successful at interpolating over the training set. In practice a nearest-neighbor implementation is unlikely to be perfect, and will generate larger errors. We have applied a nearest neighbor (NN) approach to the two global methods on the *Normal* set.

5 Experimental Results

Table 1 depicts the results for the *Normal* set. The mean localization error, maximum outlier, and offline and online run-

	Mean (cm)	Max (cm)	Estimates
<i>Global Methods</i>			
Edge Density	8.4	21.9	29/29
PCA	5.9	11.9	29/29
<i>Local Methods</i>			
DynamicLM	11.8	26.4	29/29
FixedLM	11.0	27.9	6/29

Table 2: Localization results for *Noisy* set.

	Mean (cm)	Max (cm)	Estimates
<i>Global Methods</i>			
Edge Density	60.3	215	29/29
PCA	11.2	21.7	29/29
<i>Local Methods</i>			
DynamicLM	22.0	205	29/29
FixedLM	27.3	180	28/29

Table 3: Localization results for noisy *Occlusion* set.

ning times are depicted. For the two global methods, nearest neighbor results are also tabulated for comparison. Only the PCA and FixedLM methods approach the magic number. The nearest neighbor approaches perform poorly against the magic number. As expected, PCA presents a large offline computational cost, but very low online cost.

Table 2 depicts the results for the *Noisy* set. In the case of the landmark-based approach, we have indicated error results for only those images where at least one landmark was successfully matched. The number of estimates is indicated in the last column, out of the total number of verification images. Both global methods saw an improvement in performance.

Table 3 depicts the results for the *Occlusion* set with noisy occluders. The local methods perform significantly worse, despite improved matching output in the FixedLM case. PCA continues to perform well, but the edge density approach is confounded by the introduction of regions of high edge density.

Table 4 depicts the results for the *Occlusion* set with solid occluders. The local methods demonstrate a degradation in performance, although not as great as for the white noise occluders. There is also a reversal in performance for PCA versus Edge Density.

Table 5 indicates the results under altered illumination conditions. While all of the methods perform poorly, the Edge Density approach suffers the least degradation.

	Mean (cm)	Max (cm)	Estimates
<i>Global Methods</i>			
Edge Density	10.5	30.8	29/29
PCA	96.3	222	29/29
<i>Local Methods</i>			
DynamicLM	12.1	48.6	29/29
FixedLM	15.6	100	28/29

Table 4: Localization results for solid *Occlusion* set.

	Mean (cm)	Max (cm)	Estimates
<i>Global Methods</i>			
Edge Density	30.9	103	10/10
PCA	149	217	10/10
<i>Local Methods</i>			
DynamicLM	105	204	10/10
FixedLM	138	271	9/10

Table 5: Localization results for *Illumination* set.

6 Discussion and Conclusions

Both PCA and the landmark-based methods present significant offline costs, calling into question their practicality for applications such as simultaneous localization and mapping. However, PCA was robust to some types of adverse imaging conditions, and was demonstrably faster in computing online pose estimates.

While the Edge Density method was less accurate than PCA, it took significantly less time to train, at the cost of slower online performance. However, the ongoing increase in CPU speeds makes global methods such as these increasingly practical.

Finally, the landmark-based methods presented a smooth degradation in performance as the imaging conditions became increasingly adverse. In general, these methods performed well, but were significantly more expensive online due to the cost of matching and frequent invocation of the landmark prediction model. Online performance of these approaches depends to a large extent on the nature of the prediction model itself.

We have presented a comparison of several localization methods in a known environment against a variety of imaging conditions. The results challenge some widely held assumptions about the various advantages and disadvantages of local and global methods.

References

- [Nayar *et al.*, 1994] S.K. Nayar, H. Murase, and S.A. Nene. Learning, positioning, and tracking visual appearance. In *Proc. IEEE Conf on Robotics and Automation*, pages 3237–3246, San Diego, CA, May 1994.
- [Pourraz and Crowley, 1999] F. Pourraz and J. L. Crowley. Continuity properties of the appearance manifold for mobile robot position estimation. In *Proceedings of the 2nd IEEE Workshop on Perception for Mobile Agents*, Ft. Collins, CO, June 1999. IEEE Press.
- [Se *et al.*, 2002] S. Se, D. Lowe, and J. Little. Global localization using distinctive visual features. In *Proc. Int. Conf on Intelligent Robots and Systems (IROS)*, pages 226–231, Lausanne, Switzerland, October 2002.
- [Sim and Dudek, 2001] R. Sim and G. Dudek. Learning generative models of scene features. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Lihue, HI, December 2001. IEEE Press.
- [Ulrich and Nourbakhsh, 2000] Iwan Ulrich and Illah Nourbakhsh. Appearance-based place recognition for topological localization. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 1023–1029. IEEE Press, April 2000.

Quantum Computation and Image Processing: New Trends in Artificial Intelligence

S.E. Venegas-Andraca and S. Bose

Centre for Quantum Computation, Department of Physics, University of Oxford

Parks Road, Oxford OX1 3PU, U.K.

{svenegas, s.bose}@qubit.org

http://www.qubit.org

Abstract

The analysis and use of visual information is a first order task for AI researchers. Due to the architecture of classical computers and to the computational complexity of state-of-the-art algorithms, it is required to find better ways to store, process and retrieve information for image processing. One plausible and exciting approach is Quantum Information Processing (QIP). In this poster we present an initial step towards the definition of an emerging field, Quantum Image Processing, by showing how to store an image using a quantum system as well as some of the unique properties of that storage process derived from quantum mechanics laws.

1 Introduction

In 1985 Deutsch [Deutsch, 1985] developed a theoretical machine named Universal Quantum Turing Machine (UQTM), that is, a machine based on quantum theory capable of performing computations, and showed that such machine was a generalization of a universal Turing machine. It is also shown in [Deutsch, 1985] that a quantum computer (QC), that is, a *physical* system capable of performing computations according to the rules of quantum mechanics, can perform certain tasks faster than its classical counterpart. Deutsch and Jozsa [Deutsch and Jozsa, 1992] and Shor [Shor, 1994] showed concrete problems where such speed-up is possible.

Among QC properties, we find:

1. **Superposition of states.** The basic component of a QC is a qubit, that is, a physical entity (such as an electron or a photon) that can be mathematically represented as a vector in a 2D Hilbert space $\mathcal{H}^2$. The general form of a qubit is

$$|\psi\rangle = \alpha|x\rangle + \beta|y\rangle$$

where α, β are complex numbers constrained by $|\alpha|^2 + |\beta|^2 = 1$ and $\{|x\rangle, |y\rangle\}$ is an arbitrary basis of $\mathcal{H}^2$ [Nielsen and Chuang, 2000]. Thus, $|\psi\rangle$ is a superposition of states $|x\rangle$ and $|y\rangle$, and therefore $|\psi\rangle$ can be prepared in an infinite number of ways by varying the values of α and β . See figure 1.a. In contrast, classical computers measure bit values using only one basis, $\{0,1\}$ and the only two possible states are those that correspond to the measurement outcomes, 0 or 1.

2. **Entanglement.** Entanglement is a special correlation among quantum systems that has no paragon in classical systems. Entanglement is seen to be at the heart of QIP unique properties, and an example of it is its role in Quantum Teleportation [Nielsen and Chuang, 2000].

The previous properties may have deep implications in several fields of Computer Science, in particular in Artificial Intelligence (AI) both on theoretical (e.g. faster algorithms and secure transmissions) and technological spheres (current technology is now being built taking into account quantum effects due to component size).

The purpose of this poster is to propose an initial step towards the integration of QIP techniques in image processing and, as a future work, in related fields as Computer Vision and Pattern Recognition.

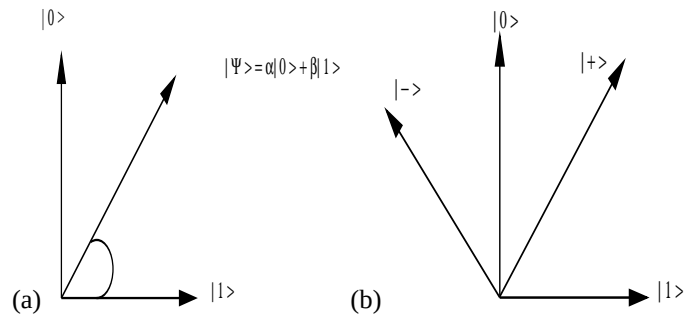


Figure 1: . Mathematical Representation for a qubit

a) The canonical counterpart of classical bits $\{0, 1\}$ in QIP is $\{|0\rangle, |1\rangle\}$. An arbitrary qubit can be written as $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $|\alpha|^2 + |\beta|^2 = 1$.

b) Note that $|\Psi\rangle$ can be written as a linear combination of an infinite number of bases, in particular as a combination of either $B_1 = \{|0\rangle, |1\rangle\}$ or $B_2 = \{|+\rangle, |-\rangle\}$, where $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ and $|-\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$.

2 Storing an image in a qubit array

2.1 Visual Information Storage in a Multi Particle Quantum System

Let $Q = \{|q_1\rangle_1, |q_2\rangle_2, \dots, |q_n\rangle_n\}$ be a set of qubits. Our goal is to store visual information in Q . In order to take advantage of $\mathcal{H}^2$ properties, allowed quantum states used to store visual information in a qubit $|q_i\rangle$ are $M = \{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$.

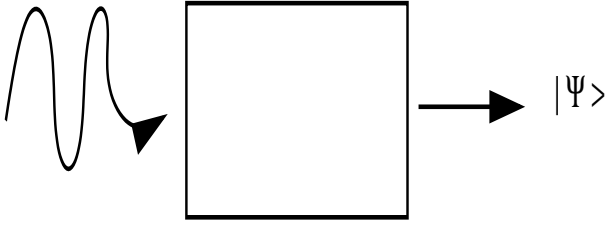


Figure 2: . Frequency to quantum state apparatus

Schematics of an apparatus capable of detecting electromagnetic frequencies and producing a set of quantum states as output. $|\psi\rangle$ being an overall state of that set of qubits

We define a machine B that divides the whole frequency range of the visible spectra into a partition $F_1, F_2, \dots, F_M$ and assigns a qubit array (that is, a set of spatially ordered qubit states) Q_i to each partition subset F_i (see figure 2). Note that B must act as a bijective function between frequency partition subsets F_i and sets of qubit states Q_i , i.e. it is not allowed to have the same qubit state distribution for two different frequency subsets. So, for each pixel from a given image, B produces a particular qubit array from the set $\{Q_i\}$.

Let us suppose that machine B produces only qubit states $|0\rangle$ and $|1\rangle$. Thus, the number of colors that can be represented in an array of n qubits is 2^n , just like in the classical case (for gray scale or primary color values, for instance) but, if we include qubit states $|+\rangle$ and $|-\rangle$, then the number of different colors that can be stored is 2^{2n} .

However, quantum mechanics has a constraint: reading (that is, measuring) non-orthogonal states (as $|0\rangle, |1\rangle, |+\rangle$ and $|-\rangle$ are) is a probabilistic process. Thus, if only one basis is used to measure all states in M , information retrieval will not be accurate. In addition, the postulates of quantum mechanics state that the post-measurement quantum state is, in general, different from the pre-measurement quantum state (they are equal if and only if the pre-measurement quantum state is equal to one of the orthogonal vectors of the measurement basis).

In order to perform accurate measurements, we now introduce per each pixel, a set of control qubits $C = \{|p_1\rangle_1, |p_2\rangle_2, \dots, |p_n\rangle_n\}$. $|0\rangle$ and $|1\rangle$ are the only allowed quantum states for elements in C . The purpose of this new element is to use qubit $|p_i\rangle$ to know which basis has to be used to perform a measurement on $|q_i\rangle$ (a generalization of this scheme is to allow the quantum programmer to choose only one storing basis for each pixel). So, by measuring qubit $|p_i\rangle$ in the canonical basis, it is possible to perform accurate measurements. Formally speaking, the observable used to measure control qubits $|p_i\rangle$ is $\hat{A}_1 = \alpha_1 |0\rangle\langle 0| + \beta_1 |1\rangle\langle 1|$. The observables used to measure qubits $|q_i\rangle$ are, in the case the outcome in measuring $\hat{A}_1$ was α_1 then $\hat{A}_2 = \alpha_2 |0\rangle\langle 0| + \beta_2 |1\rangle\langle 1|$, while in the case the outcome was β_1 , then we measure $\hat{A}_2' = \alpha_2' |+\rangle\langle +| + \beta_2' |-\rangle\langle -|$.

2.2 Quantum secrecy vs Eavesdropping

It is relevant to point out the fact that having no *a priori* knowledge of which measurement basis has to be used to read information in qubits $|q_i\rangle$ introduces two very convenient

properties: secrecy and eavesdropping detection. Indeed, if an eavesdropper tries to read the information in $|q_i\rangle$ without reading $|p_i\rangle$, two main advantages with no counterpart in classical computers are found: a) the eavesdropper has an only 50% chance of reading accurate information (if he/she fails to choose the right measurement basis, he/she will get only a random outcome), and b) the post-measurement state of a measurement performed with a wrong basis allows us to detect eavesdropping [Bouwmeester *et al.*, 2001]. This means that in order to provide secure transmission of information, it is sufficient to send only control qubits $|p_i\rangle$ via secure cryptographic scheme.

3 Future Work

Next steps include the use of entanglement for gray level and color image segmentation. The basic idea is to entangle qubits with similar colors and to develop quantum algorithms for image segmentation.

Acknowledgments

S.E. Venegas-Andraca gratefully acknowledges the receipt of a CONACyT scholarship from the Mexican National Council of Science and Technology.

The authors want to thank N. Paunković for his very stimulating and wise comments about this work.

References

- [Deutsch, 1985] David Deutsch. Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer. In *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, Volume 400, Issue 1818, pages 97–117, July 8, 1985.
- [Deutsch and Jozsa, 1992] David Deutsch and Richard Jozsa. Rapid Solution of Problems by Quantum Computation. In *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, Volume 439, Issue 1907, pages 553–558, December 8, 1992.
- [Shor, 1994] Peter Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Algorithms on a Quantum Computer. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, Santa Fe, NM, Nov. 20–22, 1994. IEEE Computer Society Press.
- [Nielsen and Chuang, 2000] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, United Kingdom, 2000.
- [Gonzalez and Woods, 1993] R. Gonzalez and R. Woods. *Digital Image Processing*. Addison-Wesley Co., 1993.
- [Bouwmeester *et al.*, 2001] Dirk Bouwmeester, Artur Ekert and Anton Zeilinger, editors. *The Physics of Quantum Information*. Springer-Verlag, Berlin, Germany, 2001.