

MULTIAGENT SYSTEMS

MULTIAGENT SYSTEMS

COALITION FORMATION

Complexity of Determining Nonemptiness of the Core*

Vincent Conitzer and Tuomas Sandholm

Carnegie Mellon University
Computer Science Department

5000 Forbes Avenue
Pittsburgh, PA 15213, USA
{conitzer,sandholm}@cs.cmu.edu

Abstract

Coalition formation is a key problem in automated negotiation among self-interested agents, and other multiagent applications. A coalition of agents can sometimes accomplish things that the individual agents cannot, or can do things more efficiently. However, motivating the agents to abide to a solution requires careful analysis: only some of the solutions are stable in the sense that no group of agents is motivated to break off and form a new coalition. This constraint has been studied extensively in cooperative game theory. However, the computational questions around this constraint have received less attention. When it comes to coalition formation among software agents (that represent real-world parties), these questions become increasingly explicit.

In this paper we define a concise general representation for games in characteristic form that relies on superadditivity, and show that it allows for efficient checking of whether a given outcome is in the core. We then show that determining whether the core is nonempty is $\mathcal{NP}$ -complete both with and without transferable utility. We demonstrate that what makes the problem hard in both cases is determining the collaborative possibilities (the set of outcomes possible for the grand coalition), by showing that if these are given, the problem becomes tractable in both cases. However, we then demonstrate that for a hybrid version of the problem, where utility transfer is possible only within the grand coalition, the problem remains $\mathcal{NP}$ -complete even when the collaborative possibilities are given.

1 Introduction

Coalition formation is a key problem in automated negotiation among self-interested agents. A coalition of agents can

sometimes accomplish things that the individual agents cannot, or can do things more efficiently. However, motivating the agents to abide to a solution requires careful analysis: only some of the solutions are *stable* in the sense that no group of agents is motivated to break off and form a new coalition. This constraint has been studied extensively in cooperative game theory. However, the computational questions around this constraint have received less attention. When it comes to coalition formation among software agents (that represent real-world parties), these questions become increasingly explicit.

In general, computational complexity could stem from each potential coalition having some hard optimization problem. For example, when the agents are carrier companies with their own trucks and delivery tasks, they can save costs by forming a coalition (pooling their trucks and tasks), but each potential coalition faces a hard optimization problem: a vehicle routing problem defined by the coalition's trucks and tasks. The effect of such hard optimization problems on coalition formation has been studied by Sandholm and Lesser [Sandholm and Lesser, 1997]. As in the bulk of research on coalition formation, in this paper we do not address that issue. Rather, we assume that such optimization problems have already been solved (at least the pertinent ones), and given this, we characterize the stable feasible outcomes. This has been the focus of most of the work in coalition formation. The contribution of this paper belongs to the relatively new, small set of papers that study the complexity of characterizing such solutions.

The determination of stable solutions has electronic commerce applications beyond automated negotiation as well. For example, consider a large number of companies, some subsets of which could form profitable virtual organizations that can respond to larger or more diverse orders than the individual companies can. Determining stable value divisions allows one to see which potential virtual organizations would be viable in the sense that the companies in the virtual organization would naturally stay together. As another example, consider a future online service that determines how much each employee of a company should be paid so that the company does not collapse as a result of employees being bought away by other companies. The input to this service would be how much subsets of the company's employees would be paid if they left collectively (for instance, a whole department

*The material in this paper is based upon work supported by the National Science Foundation under CAREER Award IRI-9703122, Grant IIS-9800994, ITR IIS-0081246, and ITR IIS-0121678.

could be bought away). This input could come from salary databases or a manager's estimate. The computational problem of determining a stable remuneration would be crucial for such a service. Both of these example problems fit exactly under the model that we study in this paper.

The rest of the paper is organized as follows. In Section 2, we review the required concepts from cooperative game theory. In Section 3, we define a concise general representation for games in characteristic form that relies on superadditivity, and show that it allows for efficient checking of whether a given outcome is in the core. In Section 4, we show that determining whether the core is nonempty is $\mathcal{NP}$ -complete both with and without transferable utility. In Section 5, we demonstrate that what makes the problem hard in both cases is determining the collaborative possibilities (the set of outcomes possible for the grand coalition), by showing that if these are given, the problem becomes tractable in both cases. In Section 6, we show that for a hybrid version of the problem, where utility transfer is possible only within the grand coalition, the problem remains $\mathcal{NP}$ -complete even when the collaborative possibilities are given.

2 Definitions from cooperative game theory

In this section we review standard definitions from cooperative game theory, which we will use throughout the paper. In the definitions, we follow the most prevalent advanced textbook in microeconomics [Mas-Colell *et al.*, 1995].

In general, how well agents in a coalition do may depend on what nonmembers of the coalition do (e.g. [Bernheim *et al.*, 1987; Chatterjee *et al.*, 1993; Evans, 1997; Milgrom and Roberts, 1996; Moreno and Wooders, 1996; Okada, 1996; Ray, 1996]). However, in cooperative game theory, coalition formation is usually studied in the context of *characteristic function games* where the utilities of the coalition members do not depend on the nonmembers' actions [Kahan and Rapoport, 1984; van der Linden and Verbeek, 1985; Zlotkin and Rosenschein, 1994; Charnes and Kortanek, 1966; Shapley, 1967; Wu, 1977]. (One way to interpret this is to consider the coalition members' utilities to be the utilities they can *guarantee* themselves no matter what the nonmembers do [Aumann, 1959; Tohmé and Sandholm, 1999].)

Definition 1 Given a set of players A , a utility possibility vector u^B for $B = \{b_1, \dots, b_{n_B}\} \subseteq A$ is a vector $(u_{b_1}, \dots, u_{b_{n_B}})$ representing utilities that the players in B can guarantee themselves by cooperating with each other. A utility possibility set is a set of utility possibility vectors for a given set B .

Definition 2 A game in characteristic form consists of a set of players A and a utility possibility set $V(B)$ for each $B \subseteq A$.

Sometimes games in characteristic form have *transferable utility*, which means agents in a coalition can transfer utility among themselves.

Definition 3 A game in characteristic form is said to have transferable utility if for every $B \subseteq A$ there is a number $v(B)$ (the value of B) such that $V(B) = \{u^B = (u_{b_1}^B, \dots, u_{b_{n_B}}^B) : \sum_{b \in B} u_b^B \leq v(B)\}$.

It is commonly assumed that the joining of two coalitions does not prevent them from acting as well as they could have acted separately. In other words, the composite coalition can coordinate by choosing not to coordinate. This assumption is known as *superadditivity*.¹ We will assume superadditivity throughout the paper. This actually makes our hardness results *stronger* because even a restricted version of the problem is hard.

Definition 4 A game in characteristic form is said to be superadditive if, for any $B, C \subseteq A$ with B and C disjoint, and for any $u^B \in V(B)$ and $u^C \in V(C)$, we have $(u^B, u^C) \in V(B \cup C)$. (In the case of transferable utility, this is equivalent to saying that for any $B, C \subseteq A$ with B and C disjoint, $v(B \cup C) \geq v(B) + v(C)$.)

We now need a solution concept. In this paper, we study only the best known solution concept, which is called the *core* [Mas-Colell *et al.*, 1995; Kahan and Rapoport, 1984; van der Linden and Verbeek, 1985]. It was first introduced by Gillies [Gillies, 1953].

Definition 5 An outcome $u^A = (u_1^A, \dots, u_n^A) \in V(A)$ is blocked by coalition $B \subseteq A$ if there exists $u^B = (u_{b_1}^B, \dots, u_{b_{n_B}}^B) \in V(B)$ such that for all $b \in B$, $u_b^B > u_b^A$. (In the case of transferable utility, this is equivalent to saying that the outcome is blocked by B if $v(B) > \sum_{b \in B} u_b^A$.) An outcome is in the core if it is blocked by no coalition.

In general, the core can be empty. If the core is empty, the game is inherently unstable because no matter what outcome is chosen, some subset of agents is motivated to pull out and form their own coalition. In other words, requiring that no subset of agents is motivated to break off into a coalition of its own overconstrains the system.

An example of a game with an empty core is the one with players $\{x, y, z\}$, where we have the utility possibility vectors $u^{\{x,y\}} = (2, 1)$, $u^{\{y,z\}} = (2, 1)$, and $u^{\{x,z\}} = (1, 2)$ (and the ones that can be derived from this through superadditivity). The same example with transferable utility also has an empty core.

In the rest of this paper, we will study the question of how complex it is to determine whether the core is nonempty, that is, whether there is a solution or the problem is overconstrained.

3 Representing characteristic form games concisely

In our representation of games in characteristic form, we distinguish between games without transferable utility, where we specify some utility possibility vectors for some coalitions, and games with transferable utility, where we specify the values of some coalitions.

¹When superadditivity holds, it is always best for the grand coalition of all agents to form. On the other hand, without superadditivity, even finding the optimal coalition structure (partition of agents into coalitions) can be hard [Sandholm *et al.*, 1999; Larson and Sandholm, 2000; Shehory and Kraus, 1998; 1996; Ketchpel, 1994].

If the representation of the game specifies $V(B)$ or $v(B)$ explicitly for each coalition $B \subseteq A$, then the length of the representation is exponential in the number of agents. In that case, any algorithm for evaluating nonemptiness of the core (as long as it reads all the input) requires time exponential in the number of agents. However, that run time is polynomial in the size of the input (this can be accomplished, for example, using the algorithms that we introduce in Section 5).

Of course, most characteristic form games that represent real-world settings have some special structure. This usually allows for a game representation that is significantly more concise. The complexity of characterizing the core has already been studied in certain very specific concisely expressible families of games before. For example, Faigle et al. study the complexity of testing membership in the core in minimum cost spanning tree games [Faigle et al., 1994]. Deng and Papadimitriou study games where the players are nodes of a graph with weights on the edges, and the value of a coalition is determined by the total weight of the edges contained in it [Deng and Papadimitriou, 1994]. Deng et al. study an integer programming formulation which captures many games on graphs [Deng et al., 1997]. All of those results depend heavily on concise game representations which are specific to the game families under study. Typically, such a family of games is played on a combinatorial structure. Cooperative games on combinatorial structures have been systematically studied [Bilbao, 2000].

As a point of deviation, we study a natural representation that can capture *any* characteristic form game.² Conciseness in our representation stems only from the fact that in many settings, the synergies among coalitions are sparse. When a coalition introduces no new synergy, its utility possibility vectors can be *derived* using superadditivity. Therefore, the input needs to include only the utility possibility vectors of coalitions that introduce synergy. The following definitions make this precise.

Definition 6 We represent a game in characteristic form without transferable utility by a set of players A , and a set of utility possibility vectors $W = \{(B, u^{B,k})\}$. (Here there may be multiple vectors for the same B , distinguished by different k indices.) The utility possibility set for a given $B \subseteq A$ is then given by $V(B) = \{u^B : u^B = (u^{B_1}, \dots, u^{B_r}), \bigcup_{1 \leq i \leq r} B_i = B, \text{ all the } B_i \text{ are disjoint, and for all the } B_i, (B_i, u^{B_i}) \in W\}$. To avoid senseless cases that have no outcomes, we also require that $(\{a\}, (0)) \in W$ for all $a \in A$.³

Definition 7 We represent a game in characteristic form with transferable utility by a set of players A , and a set of values $W = \{(B, v(B))\}$. The value for a given $B \subseteq A$ is then given by $v(B) = \max\{\sum_{1 \leq i \leq r} v(B_i) : \bigcup_{1 \leq i \leq r} B_i = B, \text{ all the } B_i \text{ are disjoint, and for all the } B_i, (B_i, v(B_i)) \in W\}$. To avoid senseless cases that have no outcomes, we also require

²Our hardness results are not implied by the earlier hardness results for specific game families because it is not possible to concisely represent those games in our input language.

³Setting the utility to 0 in this case is without loss of generality, as we can simply normalize the utility function to obtain this.

that $(\{a\}, 0) \in W$ whenever $\{a\}$ does not receive a value elsewhere in W .

So, we only need to specify a basis of utility possibilities, from which we can then derive the others. This representation integrates rather nicely with real-world problems where determining any coalition's value is complex. For example, in the multiagent vehicle routing problem, we solve the routing problem for every coalition that might introduce new synergies. When it is clear that there is no synergy between two coalitions (for example, if they operate in different cities and each one only has deliveries within its city), there is no need to solve the routing problem of the coalition that would result if the two coalitions were to merge.

The following lemmas indicate that we can also use this representation effectively for checking whether an outcome is in the core, that is, whether it satisfies the strategic constraints.

Lemma 1 Without transferable utility, an outcome $u^A = (u_1^A, \dots, u_n^A) \in V(A)$ is blocked by some coalition if and only if it is blocked by some coalition B through some utility vector u^B , where $(B, u^B) \in W$.

Proof: The "if" part is trivial. For "only if", suppose u^A is blocked by coalition C through some u^C , so that for every $c \in C$, $u_c^C > u_c^A$. We know $u^C = (u^{C_1}, \dots, u^{C_r})$ where $(C_i, u^{C_i}) \in W$. But then, C_1 blocks u^A through u^{C_1} . ■

The proof for the same lemma in the case of transferable utility is only slightly more intricate.

Lemma 2 With transferable utility, an outcome $u^A = (u_1^A, \dots, u_n^A) \in V(A)$ is blocked by some coalition if and only if it is blocked by some coalition B through its value $v(B)$, where $(B, v(B)) \in W$.

Proof: The "if" part is trivial. For "only if", suppose u^A is blocked by coalition C through $v(C)$, so that $v(C) > \sum_{c \in C} u_c^A$.

We know that $v(C) = \sum_{1 \leq i \leq r} v(C_i)$ where $(C_i, v(C_i)) \in W$.

It follows that $\sum_{1 \leq i \leq r} v(C_i) > \sum_{1 \leq i \leq r} \sum_{c \in C_i} u_c^A$, and hence for at least one C_i , we have $v(C_i) > \sum_{c \in C_i} u_c^A$. But then C_i blocks u^A through $v(C_i)$. ■

4 Checking whether the core is nonempty is hard

We now show that with this representation, it is hard to check whether the core is nonempty. This holds both for the non-transferable utility setting and for the transferable utility setting.

Definition 8 (CORE-NONEMPTY) We are given a super-additive game in characteristic form (with or without transferable utility) in our representation language. We are asked whether the core is nonempty.

We will demonstrate $\mathcal{NP}$ -hardness of this problem by reducing from the $\mathcal{NP}$ -complete EXACT-COVER-BY-3-SETS problem [Garey and Johnson, 1979].

Definition 9 (EXACT-COVER-BY-3-SETS) We are given a set S of size $3m$ and a collection of subsets $\{S_i\}_{1 \leq i \leq r}$ of S , each of size 3. We are asked if there is a cover of S consisting of m of the subsets.

We are now ready to state our results.

Theorem 1 *CORE-NONEMPTY without transferable utility is $\mathcal{NP}$ -complete.*

Proof: To show that the problem is in $\mathcal{NP}$, nondeterministically choose a subset of W , and check if the corresponding coalitions constitute a partition of A . If so, check if the outcome corresponding to this partition is blocked by any element of W .

To show $\mathcal{NP}$ -hardness, we reduce an arbitrary EXACT-COVER-BY-3-SETS instance to the following CORE-NONEMPTY instance. Let the set of players be $A = S \cup \{w, x, y, z\}$. For each S_i , let (S_i, u^{S_i}) be an element of W , with $u^{S_i} = (2, 2, 2)$. Also, for each $s \in S$, let $(\{s, w\}, u^{\{s, w\}})$ be an element of W , with $u^{\{s, w\}} = (1, 4)$. Also, let $(\{w, x, y, z\}, u^{\{w, x, y, z\}})$ be an element of W , with $u^{\{w, x, y, z\}} = (3, 3, 3, 3)$. Finally, let $(\{x, y\}, u^{\{x, y\}})$ with $u^{\{x, y\}} = (2, 1)$, $(\{y, z\}, u^{\{y, z\}})$ with $u^{\{y, z\}} = (2, 1)$, $(\{x, z\}, u^{\{x, z\}})$ with $u^{\{x, z\}} = (1, 2)$ be elements of W . The only other elements of W are the required ones giving utility 0 to singleton coalitions. We claim the two instances are equivalent.

First suppose there is an exact cover by 3-sets consisting of $S_{c_1}, \dots, S_{c_m}$. Then the following outcome is possible: $(u^{S_{c_1}}, \dots, u^{S_{c_m}}, u^{\{w, x, y, z\}}) = (2, 2, \dots, 2, 3, 3, 3, 3)$. It is easy to verify that this outcome is not blocked by any coalition. So the core is nonempty.

Now suppose there is no exact cover by 3-sets. Suppose the core is nonempty, that is, it contains some outcome $u^A = (u^{C_1}, \dots, u^{C_r})$ with each (C_i, u^{C_i}) an element of W , and the C_i disjoint. Then one of the C_i must be $\{s, w\}$ for some $s \in S$: for if this were not the case, there must be some $s \in S$ with $u_s^A = 0$, because the C_i that are equal to S_i cannot cover S ; but then $\{s, w\}$ would block the outcome. Thus, none of the C_i can be equal to $\{w, x, y, z\}$. Then one of the C_i must be one of $\{x, y\}, \{y, z\}, \{x, z\}$, or else two of $\{x, y, z\}$ would block the outcome. By symmetry, we can without loss of generality assume it is $\{x, y\}$. But then $\{y, z\}$ will block the outcome. (Contradiction.) So the core is empty. ■

We might hope that the convexity introduced by transferable utility makes the problem tractable through, for example, linear programming. This turns out not to be the case.

Theorem 2 *CORE-NONEMPTY with transferable utility is $\mathcal{NP}$ -complete.*

Proof: To show that the problem is in $\mathcal{NP}$, nondeterministically choose a subset of W , and check if the corresponding coalitions constitute a partition of A . If so, nondeterministically divide the sum of the coalitions' values over the players, and check if this outcome is blocked by any element of W .

To show $\mathcal{NP}$ -hardness, we reduce an arbitrary EXACT-COVER-BY-3-SETS instance to the following CORE-NONEMPTY instance. Let the set of players be

$A = S \cup \{x, y\}$. For each S_i , let $(S_i, 3)$ be an element of W . Additionally, let $(S \cup \{x\}, 6m)$, $(S \cup \{y\}, 6m)$, and $(\{x, y\}, 6m)$ be elements of W . The only other elements of W are the required ones giving value 0 to singleton coalitions. We claim the two instances are equivalent.

First suppose there is an exact cover by 3-sets consisting of $S_{c_1}, \dots, S_{c_m}$. Then the value of coalition S is at least $\sum_{1 \leq i \leq m} v(S_{c_i}) = 3m$. Combining this with the coalition $\{x, y\}$, which has value $6m$, we conclude that the grand coalition A has value at least $9m$. Hence, the outcome $(1, 1, \dots, 1, 3m, 3m)$ is possible. It is easy to verify that this outcome is not blocked by any coalition. So the core is nonempty.

Now suppose there is no exact cover by 3-sets. Then the coalition S has value less than $3m$ (since there are no m disjoint S_i), and as a result the value of the grand coalition is less than $9m$. It follows that in any outcome, the total utility of at least one of $S \cup \{x\}$, $S \cup \{y\}$, and $\{x, y\}$ is less than $6m$. So this coalition will block. So the core is empty. ■

Our results imply that it is computationally hard to make any strategic assessment of a game in characteristic form when it is concisely represented.

5 Specifying redundant information about the grand coalition makes the problem tractable

Our proofs that CORE-NONEMPTY is hard relied on constructing instances where it is difficult to determine what the grand coalition can accomplish. So, in effect, the hardness derived from the fact that even collaborative optimization is hard in these instances. While this is indeed a real difficulty that occurs in the analysis of characteristic form games, we may nevertheless wonder to what extent computational complexity issues are introduced by the purely strategic aspect of the games. To analyze this, we investigate the computational complexity of CORE-NONEMPTY when $V(A)$ (or $v(A)$) is explicitly provided as (possibly redundant) input, so that determining what the grand coalition can accomplish can no longer be the source of any complexity.⁴ It indeed turns out that the problem becomes easy both with and without transferable utility.

Theorem 3 *When $V(A)$ is explicitly provided, CORE-NONEMPTY without transferable utility is in $\mathcal{P}$.*

Proof: The following simple algorithm accomplishes this efficiently. For each element of $V(A)$, check whether it is blocked by any element of W . ■

For the transferable utility case, we make use of linear programming.

Theorem 4 *When $v(A)$ is explicitly provided, CORE-NONEMPTY with transferable utility is in $\mathcal{P}$.*

⁴Bilbao et al. have studied the complexity of the core in characteristic form games with transferable utility when there is an oracle that can provide the value $v(B)$ of any coalition B [Bilbao et al.,]. Our amended input corresponds to asking one such query in addition to obtaining the unamended input.

Proof: We decide how to allocate the $v(A)$ among the agents by solving a linear program. The core is nonempty if and only if the following linear program has a solution:

- $\sum_{1 \leq i \leq n} u_i \leq v(A)$;
- For any $(B, v(B))$ in W , $\sum_{b \in B} u_b \geq v(B)$. ■

The algorithms in the proofs also construct a solution that is in the core, if the core is nonempty.

6 Hybrid games remain hard

Not all complexity issues disappear through having the collaborative optimization problem solution available. It turns out that if we allow for *hybrid* games, where only *some* coalitions can transfer utility among themselves, the hardness returns. In particular, we show hardness in the case where only the grand coalition can transfer utility. This is a natural model for example in settings where there is a market institution that enforces payments, but if a subset of the agents breaks off, the institution collapses so payments cannot be enforced.

We demonstrate $\mathcal{NP}$ -hardness of this problem by reducing from the $\mathcal{NP}$ -complete NODE-COVER problem [Garey and Johnson, 1979].

Definition 10 (NODE-COVER) *We are given a graph $G = (V, E)$, and a number k . We are asked whether there is a subset of V of size k such that each edge has at least one of its endpoints in the subset.*

We are now ready to state our result.

Theorem 5 *When only the grand coalition can transfer utility, CORE-NONEMPTY is $\mathcal{NP}$ -complete, even when $v(A)$ is explicitly provided as input.*

Proof: To show that the problem is in $\mathcal{NP}$, nondeterministically divide $v(A)$ over the players, and check if this outcome is blocked by any element of W .

To show $\mathcal{NP}$ -hardness, we reduce an arbitrary NODE-COVER instance to the following CORE-NONEMPTY instance. Let $A = V \cup \{x, y, z\}$, and let $v(A) = 6|V| + k$. Furthermore, for each edge (v_i, v_j) , let $(\{v_i, v_j\}, u^{\{v_i, v_j\}})$ be an element of W , with $u^{\{v_i, v_j\}} = (1, 1)$. Finally, for any $a, b \in \{x, y, z\}$ ($a \neq b$), let $(\{a, b\}, u^{\{a, b\}})$ be an element of W , with $u^{\{a, b\}} = (3|V|, 2|V|)$. The only other elements of W are the required ones giving utility 0 to singleton coalitions. This game does not violate the superadditivity assumption, since without the explicit specification of $v(A)$, superadditivity can at most imply that $v(A) = 6|V| \leq 6|V| + k$. We claim the two instances are equivalent.

First suppose there is a node cover of size k . Consider the following outcome: all the vertices in the node cover receive utility 1, all the other vertices receive utility 0, and each of x, y , and z receives utility $2|V|$. Using the fact that all the edges are covered, it is easy to verify that this outcome is not blocked by any coalition. So the core is nonempty.

Now suppose there is some outcome u^A in the core. In such an outcome, either each of $\{x, y, z\}$ receives at least $2|V|$, or two of them receive at least $3|V|$ each. (For if not, there is some $a \in \{x, y, z\}$ with $u_a^A < 2|V|$ and some

$b \in \{x, y, z\}$ ($b \neq a$) with $u_b^A < 3|V|$, and the coalition $\{b, a\}$ will block through $u^{\{b, a\}} = (3|V|, 2|V|)$.) It follows that the combined utility of all the elements of V is at most k . Now, for each edge (v_i, v_j) , at least one of its vertices must receive utility at least 1, or this edge would block. So the vertices that receive at least 1 cover the edges. But because the combined utility of all the elements of V is at most k , there can be at most k such vertices. So there is a node cover. ■

Hybrid games, where only some coalitions can transfer utility, are quite likely to appear in real-world multiagent settings, for example because only some of the agents use a currency. Our result shows that for such hybrid games, even when the collaborative optimization problem has already been solved, it can be computationally hard to strategically assess the game.

7 Conclusions and future research

Coalition formation is a key problem in automated negotiation among self-interested agents, and other multiagent applications. A coalition of agents can sometimes accomplish things that the individual agents cannot, or can do things more efficiently. However, motivating the agents to abide to a solution requires careful analysis: only some of the solutions are stable in the sense that no group of agents is motivated to break off and form a new coalition. This constraint has been studied extensively in cooperative game theory. However, the computational questions around this constraint have received less attention. When it comes to coalition formation among software agents (that represent real-world parties), these questions become increasingly explicit.

In this paper we defined a concise general representation for games in characteristic form that relies on superadditivity, and showed that it allows for efficient checking of whether a given outcome is in the core. We then showed that determining whether the core is nonempty is $\mathcal{NP}$ -complete both with and without transferable utility. We demonstrated that what makes the problem hard in both cases is determining the collaborative possibilities (the set of outcomes possible for the grand coalition), by showing that if these are given, the problem becomes tractable in both cases. However, we then demonstrated that for a hybrid version of the problem, where utility transfer is possible only within the grand coalition, the problem remains $\mathcal{NP}$ -complete even when the collaborative possibilities are given.

Future research can take a number of different directions. One such direction is to investigate the complexity of restricted families of games in characteristic form.⁵ Another direction is to evaluate other solution concepts in cooperative game theory from the perspective of computational complexity under our input representation. A long-term goal is to extend our framework for finding a strategically stable solution to take into account issues of computational complexity

⁵One interesting restricted family is that of *convex games*. In a convex game (with transferable utility), for any $B, C \subseteq A$, $v(B) + v(C) \leq v(B \cup C) + v(B \cap C)$, and in such games the core is known to always be nonempty. How complex it is here to *construct* a solution in the core with our representation is an open question.

in determining the synergies among coalitions (for example, when routing problems need to be solved, potentially only approximately, in order to determine the synergies).

References

- [Aumann, 1959] R Aumann. Acceptable points in general cooperative n-person games. volume IV of *Contributions to the Theory of Games*. Princeton University Press, 1959.
- [Bernheim *et al.*, 1987] B Douglas Bernheim, Bezalel Peleg, and Michael D Whinston. Coalition-proof Nash equilibria: I concepts. *J. of Economic Theory*, 42(1):1–12, June 1987.
- [Bilbao *et al.*,] J. M. Bilbao, J. R. Fernandez, and J. J. Lopez. Complexity in cooperative game theory.
- [Bilbao, 2000] J. M. Bilbao. *Cooperative Games on Combinatorial Structures*. Kluwer Academic Publishers, 2000.
- [Charnes and Kortanek, 1966] A Charnes and K O Kortanek. On balanced sets, cores, and linear programming. Technical Report 12, Cornell Univ., Dept. of Industrial Eng. and Operations Res., Ithaca, NY, 1966.
- [Chatterjee *et al.*, 1993] K. Chatterjee, B. Dutta, D. Ray, and K. Sengupta. A noncooperative theory of coalitional bargaining. *Review of Economic Studies*, 60:463–477, 1993.
- [Deng and Papadimitriou, 1994] Xiaotie Deng and Christos H. Papadimitriou. On the complexity of cooperative solution concepts. *Mathematics of Operations Research*, pages 257–266, 1994.
- [Deng *et al.*, 1997] Xiaotie Deng, Toshihide Ibaraki, and Hiroshi Nagamochi. Algorithms and complexity in combinatorial optimization games. SODA 1997.
- [Evans, 1997] Robert Evans. Coalitional bargaining with competition to make offers. *Games and Economic Behavior*, 19:211–220, 1997.
- [Faigle *et al.*, 1994] U. Faigle, S. Fekete, W. Hochstattler, and W. Kern. On the complexity of testing membership in the core of min-cost spanning trees. Technical Report 94.166, Universitat zu Koln, 1994.
- [Garey and Johnson, 1979] Michael R Garey and David S Johnson. *Computers and Intractability*. W. H. Freeman and Company, 1979.
- [Gillies, 1953] D Gillies. *Some theorems on n-person games*. PhD thesis, Princeton, Dept. of Mathematics, 1953.
- [Kahan and Rapoport, 1984] James P Kahan and Amnon Rapoport. *Theories of Coalition Formation*. Lawrence Erlbaum Associates Publishers, 1984.
- [Ketchpel, 1994] Steven Ketchpel. Forming coalitions in the face of uncertain rewards. AAAI, pages 414–419, Seattle, WA, July 1994.
- [Larson and Sandholm, 2000] Kate Larson, Tuomas Sandholm. Anytime coalition structure generation: An average case study. *J. of Experimental and Theoretical AI*, 11:1–20, 2000. Early version: AGENTS, pp. 40–47, 1999.
- [Mas-Colell *et al.*, 1995] Andreu Mas-Colell, Michael Whinston, and Jerry R. Green. *Microeconomic Theory*. Oxford University Press, 1995.
- [Milgrom and Roberts, 1996] Paul Milgrom and John Roberts. Coalition-proofness and correlation with arbitrary communication possibilities. *Games and Economic Behavior*, 17:113–128, 1996.
- [Moreno and Wooders, 1996] Diego Moreno and John Wooders. Coalition-proof equilibrium. *Games and Economic Behavior*, 17:80–112, 1996.
- [Okada, 1996] Akira Okada. A noncooperative coalitional bargaining game with random proposers. *Games and Economic Behavior*, 16:97–108, 1996.
- [Ray, 1996] Indrajit Ray. Coalition-proof correlated equilibrium: A definition. *Games and Economic Behavior*, 17:56–79, 1996.
- [Sandholm and Lesser, 1997] Tuomas Sandholm and Victor R Lesser. Coalitions among computationally bounded agents. *Artificial Intelligence*, 94(1):99–137, 1997. Special issue on Economic Principles of Multiagent Systems. Early version: International Joint Conference on Artificial Intelligence (IJCAI), pages 662–669, 1995.
- [Sandholm *et al.*, 1999] Tuomas Sandholm, Kate Larson, Martin Andersson, Onn Shehory, and Fernando Tohmé. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111(1–2):209–238, 1999. Early version appeared at the National Conference on Artificial Intelligence (AAAI), pages 46–53, 1998.
- [Shapley, 1967] L. S. Shapley. On balanced sets and cores. *Naval Research Logistics Quarterly*, 14:453–460, 1967.
- [Shehory and Kraus, 1996] Onn Shehory and Sarit Kraus. A kernel-oriented model for coalition-formation in general environments: Implementation and results. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 134–140, Portland, OR, August 1996.
- [Shehory and Kraus, 1998] Onn Shehory and Sarit Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1–2):165–200, 1998.
- [Tohmé and Sandholm, 1999] Fernando Tohmé and Tuomas Sandholm. Coalition formation processes with belief revision among bounded rational self-interested agents. *Journal of Logic and Computation*, 9(97–63):1–23, 1999. Early version: IJCAI-97 Workshop on Social Interaction and Communityware, pp. 43–51, Nagoya, Japan.
- [van der Linden and Verbeek, 1985] Wim J van der Linden and Albert Verbeek. Coalition formation: A game-theoretic approach. In Henk A M Wilke, editor, *Coalition Formation*, volume 24 of *Advances in Psychology*. North Holland, 1985.
- [Wu, 1977] L. S. Wu. A dynamic theory for the class of games with nonempty cores. *SIAM Journal of Applied Mathematics*, 32:328–338, 1977.
- [Zlotkin and Rosenschein, 1994] Gilad Zlotkin and Jeffrey S Rosenschein. Coalition, cryptography and stability: Mechanisms for coalition formation in task oriented domains. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 432–437, Seattle, WA, July 1994.

An Integrated Multilevel Learning Approach to Multiagent Coalition Formation

Leen-Kiat Soh and Xin Li

Department of Computer Science and Engineering
University of Nebraska-Lincoln
115 Ferguson Hall, Lincoln, NE 68588-0115 USA
{ lksoh, xinli@cse.unl.edu }

Abstract

In this paper we describe an integrated multilevel learning approach to multiagent coalition formation in a real-time environment. In our domain, agents negotiate to form teams to solve joint problems. The agent that initiates a coalition shoulders the responsibility of overseeing and managing the formation process. A coalition formation process consists of two stages. During the *initialization* stage, the initiating agent identifies the candidates of its coalition, i.e., known neighbors that *could* help. The initiating agent negotiates with these candidates during the *finalization* stage to determine the neighbors that *are willing* to help. Since our domain is dynamic, noisy, and time-constrained, the coalitions are not optimal. However, our approach employs learning mechanisms at several levels to improve the quality of the coalition formation process. At a tactical level, we use reinforcement learning to identify viable candidates based on their potential utility to the coalition, and case-based learning to refine negotiation strategies. At a strategic level, we use distributed, cooperative case-based learning to improve general negotiation strategies. We have implemented the above three learning components and conducted experiments in multisensor target tracking and CPU re-allocation applications.

1 Introduction

Multiagent coalition formation is important for distributed applications ranging from electronic business to mobile and ubiquitous computing where adaptation to changing resources and environments is crucial. It increases the ability of agents to execute tasks and maximize their payoffs. Moreover, coalitions can dynamically disband when they are no longer needed or effective. Thus the automation of coalition formation will not only save considerable labor time, but also may be more effective at finding beneficial coalitions than human in complex settings [Jennings, 2002].

Although considerable research has been conducted either in coalition formation among self-interested agents (e.g., [Tohme and Sandholm, 1999], [Sandholm *et al.*, 1999],

[Sen and Dutta, 2000]), or in coalition formation among cooperative agents (e.g., [Shehory *et al.*, 1997]), little work has been done in coalition formation among both self-interested and cooperative agents. Furthermore, there have been no attempts to study coalition formation among such agents in a dynamic, real-time, uncertain, and noisy environment, which is a typical real-world environment and in which a sub-optimal coalition needs to be formed in a real-time manner.

We propose an integrated multilevel learning approach to multiagent coalition formation. In our approach, agents are assumed to be cautiously cooperative—they are willing to help only when they think they benefit from it—and honest. However, due to the noisy, uncertain, dynamic and real-time nature of our domain, not every agent can be correct in its perceptions and assumptions. Thus, to achieve a coalition, an initiating agent has to negotiate with other agents. Through concurrent, multiple 1-to-1 negotiations, the initiating agent identifies the agents that are willing to help. The formation process is successful if the initiating agent successfully persuades enough agents to join the coalition.

Note that in this paper, we focus on improving the quality of the coalition formation process, and not on the quality of the coalition after it is formed and executed.

Note also that our approach is an example of the “good enough, soon enough” design paradigm. In our domain, an agent has incomplete information about the environment, the task execution is time constrained, and the communication between agents is not reliable, so an optimal coalition formed from the deep learning is impractical. Thus, a sub-optimal yet fast coalition formation process is warranted.

2 Coalition Formation

In our problem domain, when an agent cannot solve a task execution or resource allocation problem by itself or can get more benefits from collaborating with other agents, it initiates a coalition formation process to form a coalition and solve the problem jointly. Figure 1 depicts our coalition formation modules that make up the two stages: initialization and finalization. The feasibility study and the ranking of candidates are the initialization stage whereas the nego-

tiations and their management the finalization stage. This two-stage model [Soh and Tsatsoulis, 2001] allows an agent to form an initial coalition *hastily* and *quickly* to react an event and to *rationalize* to arrive at a working final coalition as time progresses, as a result of our previously described domain nature.

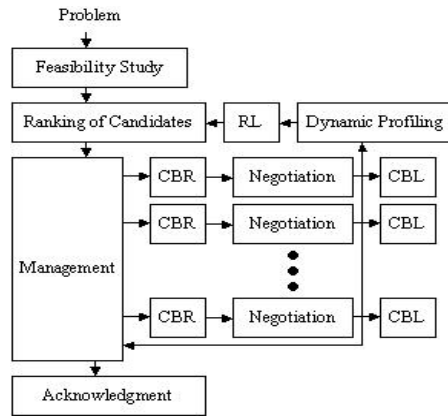


Figure 1. An overview of the coalition formation process

Here we briefly describe each module of the design:

(1) **Dynamic Profiling:** Every agent dynamically profiles each neighbor as a vector in the agent about the negotiation relationship between them, and profiles each negotiation task as a case in the casebase about the negotiation strategy description and negotiation outcome.

(2) **Feasibility Study:** This module analyzes the problem and computes (a) whether the agent has the resources to do something about it, and (b) if yes, the list of agents that the agent thinks could help.

(3) **Ranking of Candidates:** This module scores and ranks each candidate, and proportionately assigns the requested demand to each candidate, based on its *potential utility* (section 3.1).

(4) **Management:** This module initiates negotiations with top-ranked candidates. That is, the module manages multiple, concurrent 1-to-1 negotiations. For each negotiation task, it first finds a negotiation strategy through CBR. Then, it spawns a thread to execute that negotiation task. The module oversees the various negotiation threads and modifies the tasks in real-time. For example, the module will terminate all remaining negotiations once it finds out that it no longer can form a viable coalition. The module will reduce its requests or demands once it has secured agreements from successful negotiations. And so on. In effect, this management simulates a 1-to-many negotiation.

(5) **CBR:** Given the problem description of a task, the CBR module retrieves the best case from the casebase, and adapts the solution of that best case to the current problem. This is based on the work of [Soh and Tsatsoulis, 2001].

(6) **Negotiation:** Our negotiation protocol is argumentative. The initiating agent provides evidence for its request to persuade the responding agent. The responding agent evaluates these evidence pieces and if they are higher than a

dynamic persuasion threshold, then the responding agent will agree to the request. The responding agent also has the ability to counter-offer due to time constraints or poor evidence. This is based on the work of [Soh and Tsatsoulis, 2001].

(7) **Acknowledgment:** Once all negotiations are completed, if a coalition has been formed, the agent confirms the success of the coalition to all agents who have agreed to help. If the agent has failed to form a coalition, it informs the agents who have agreed to help so they can release themselves from the agreements.

In the next section, we will discuss the learning mechanisms, a critical part of our coalition formation approach.

3 Learning

Our learning approach incorporates reinforcement learning and case-based learning at two levels. At a tactical level, we use reinforcement learning to identify viable candidates based on their potential utility to the coalition, and case-based learning to refine specific negotiation strategies. At a strategic level, we use distributed, cooperative case-based learning to improve general negotiation capabilities.

3.1 Reinforcement Learning

Reinforcement learning is evident at the coalition initialization and finalization stages. During initialization, the initiating agent measures the *potential utility* of a candidate based on a weighted sum of (1) its past cooperation relationship with the initiator such as the candidate's helpfulness, friendliness, and the agent's helpfulness and importance to the candidate, (2) its current cooperation relationship with the initiating agent such as whether the two agents have already been negotiating about other problems, and (3) its ability to help towards the current problem. An initiating agent thus will more likely approach the agents that have been helpful before, thus reinforcing the cooperation relationship among them.

During finalization, an initiating agent also appeals to a candidate about how helpful the initiating agent has been in the past. A candidate is more easily persuaded if it realizes that a particular agent has been helpful in the past, and thus once again reinforcing their cooperation relationship. For details, please refer to [Soh and Tsatsoulis, 2001].

3.2 Case-Based Learning

We use CBR to retrieve and adapt negotiation strategies for negotiations during coalition finalization. We also equip our CBR module with both individual and cooperative learning capabilities (Figure 2). Individual learning refers to learning based on an agent's perceptions and actions, without direct communication with other agents. Cooperative learning refers to learning other agents' experience through interaction among agents. When an agent identifies a problematic case in its casebase, it approaches other agents to obtain a possibly better case.

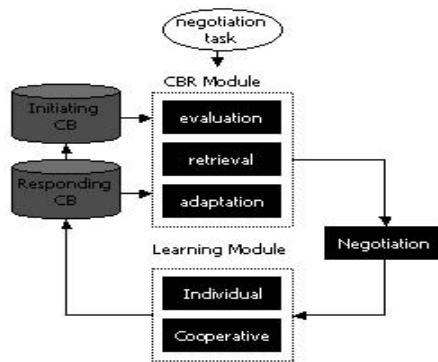


Figure 2. The relationship between case learning and CBR as well as negotiation tasks in an agent

There has been research in distributed and cooperative CBR. [Prasad and Plaza, 1996] proposed treating corporate memories as distributed case libraries. Resource discovery was achieved through (1) negotiated retrieval that dealt with retrieving and assembling case pieces from different resources in a corporate memory to form a good overall case, and (2) federated peer learning that dealt with distributed and collective CBR [Plaza *et al.*, 1997]. [Martin *et al.*, 1999] extended the model using the notion of competent agents. [Martin and Plaza, 1999] employed an auction-based mechanism that focused on agent-mediated systems where the best case was selected from the bid cases.

Our methodology employs a cautious utility-based adaptive mechanism to combine the two learning approaches, an interaction protocol for soliciting and exchanging information, and the idea of a chronological casebase. It emphasizes individual learning and only triggers cooperative learning when necessary. Our cooperative learning also differs from collective CBR in that it does not merge case pieces into one as it considers entire cases. In addition, our research focus here is to define a mechanism that combines individual and cooperative learning.

Note that the communication and coordination overhead of cooperative learning may be too high for cooperative learning to be cost-effective or timely. Moreover, since an agent learns from its own experience and its own view of the world, its solution to a problem may not be applicable for another agent facing the same problem. This injection of *foreign* knowledge may also be risky as it may add to the processing cost without improving the solution quality of an agent [Marsella *et al.*, 1999].

3.2.1 Chronological Casebase and Case Utility

We have utilized the notion of a chronological casebase in which each case is stamped with a time-of-birth (when it was created) and a time-of-membership (when it joined the casebase). All initial cases are given the same time-of-birth and time-of-membership. In addition, we profile each case's usage history (Table 1). An agent evaluates the utility of a case based on its usage history. If the case has a low utility, it may be replaced (or forgotten). If the

case is deemed problematic, then a cooperative learning will be triggered and the case will be replaced. Table 2 shows the heuristics we use in tandem with the chronological casebase. When a negotiation completes, if the *new* case adds to the casebase's diversity, the agent learns it. If the casebase's size has reached a preset limit, then the agent considers replacing one of the existing cases with the new case. For our individual case-based learning, we use heuristics *H1*, *H2*, and *H3*.

Parameters	Description
<i>TU</i>	The number of times the case has been used
<i>TSU</i>	The number of times the case has been used in a successful negotiation
<i>TINC</i>	The number of times the usage of the case has led to a new case getting added to the casebase
<i>TR</i>	The number of times the case has been designated as a problematic case, i.e., with very low utility
<i>TS</i>	The last time that the case was used or the time when the case was added

Table 1. The usage history that an agent profiles of each case

Heuristics	Description
<i>H1 Currency</i>	If a case has not been used in a long time, then this case is more likely to be replaced.
<i>H2 Evolution</i>	With everything else equal, an old case is more likely to be replaced than a young case.
<i>H3 Usefulness</i>	If a case's <i>TSU</i> is significantly small, then the case is more likely to be replaced.
<i>H4 Solution Quality I</i>	If a case has a high <i>TU</i> but a low <i>TSU</i> , then it is a problematic case.
<i>H5 Solution Quality II</i>	If a case has a low <i>TSU</i> , and a high <i>TINC</i> , then the solution of this case is probably not suitable for the problems encountered by the agent and it is a problematic case.
<i>H6 Persistence</i>	The urgency to trigger a cooperative learning process is directly dependent on the case's <i>TR</i> , until a pre-determined threshold is reached.
<i>H7 Ignorance</i>	If a case's <i>TR</i> has reached a pre-determined threshold and it is still not replaced, then the problematic case is removed without replacement.

Table 2. Heuristics that support the chronological casebase

3.2.2 Cooperative Learning

Figure 3 depicts our cooperative learning design. We adhere to a cautious approach to cooperative learning:

(1) The agent evaluates the case to determine whether it is problematic. To designate a case as problematic, we use heuristics *H4* and *H5*: a (frequently used) case is problematic if it has a low success rate (*TSU/TU*) and a high incurrance rate (*TINC/TU*). The profiling module keeps track of the utility of the cases.

(2) The agent only requests help from a selected agent that it thinks is good at a particular problem. We want to approach neighbors who have initiated successful negotiations with the current agent, with the hope that the agent may be able to learn how those neighbors have been able to be successful. This is determined based on the profile of each neighbor that the agent maintains. The exchange protocol is carried out by the case request and case response modules.

- (3) If the foreign case is similar to the problematic case, the agent adapts the foreign case before adopting it into its casebase. At the same time, the usage history parameters of the new case are reset.
- (4) If a problematic case cannot be fixed after K times, it will be removed (Heuristics $H6$ and $H7$).

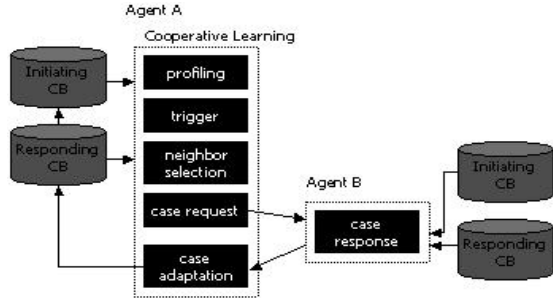


Figure 3. The cooperative learning design

4 Experiments and Results

We have implemented a multiagent system with multiple agents that perform multi-sensor target tracking and adaptive CPU reallocation in a noisy environment (simulated by a JAVA-based program called RADSIM). Each agent has the same capabilities, but is located at a unique position. Each agent controls a sensor and can activate the sensor to search-and-detect the environment. When an agent detects a moving target, it tries to implement a tracking coalition by cooperating with at least two neighbors. And this is when a CPU shortage may arise: the activity may consume more CPU resource. When an agent detects a CPU shortage, it needs to form a CPU coalition to address the crisis.

The multiagent system is implemented in C++. In the current design, each agent has $3+N$ threads. The *core* thread is responsible for making decisions and managing tasks. A *communication* thread is used to interact with the message passing system of the sensor. An *execution* thread actuates the physical sensor: calibration, search-and-detect for a target, etc. Each agent also has N *negotiation* threads to conduct multiple, concurrent negotiations.

We used two simulations for our experiments. We conducted experiments in a simulation called RADSIM where communication may be noisy and unreliable, and one or two targets may appear in the environment. We also designed and implemented our own CPU shortage simulation module. Each task is designated with a CPU usage amount plus a random factor. When an agent detects a CPU shortage, the tasks that it currently performs slow down. Thus, a CPU shortage that goes unresolved will result in failed negotiations since our negotiations are time-constrained.

4.1 Impacts of Learning

We also conducted experiments with four versions of learning: (1) both case-based reasoning and reinforcement learn-

ing (CBRR), (2) only case-based reasoning (NoRL), (3) only reinforcement learning (NoCBR), and (4) no learning at all (NoCBRR). Figure 4 shows the result in terms of the success rates for negotiations and coalition formations.

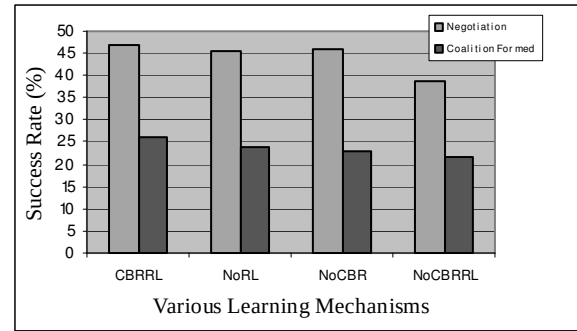


Figure 4. Success rates of negotiations and coalition formations for different learning mechanisms

The agent design with both case-based reasoning/learning and reinforcement learning outperformed others in both negotiation success rate and coalition formation success rate. That means with learning, the agents were able to negotiate more effectively (perhaps more efficiently as well) that led to more coalitions formed. Without either learning (but not both), the negotiation success rates remained about the same but the coalition formation rate tended to deteriorate. This indicates that without one of the learning methods, the agents were still able to negotiate effectively, but may be not efficiently (resulting in less processing time for the initiating agent to post-process an agreement). With no learning, the agents fared noticeably poorly.

4.2 Resource Allocation and System Coherence

We conducted experiments in CPU re-allocation to test the coherence of our system. We refer to the CPU allocation as a sustenance resource since in order for an agent to obtain more CPU, it needs to incur CPU usage while negotiating for the resource. By varying the amount of the initial CPU allocation to each agent, we created mildly-constrained, overly-constrained, and unevenly-constrained scenarios. Tables 3 and 4 compared the agents' behavior in terms of successes in negotiations and coalition formations. In particular, the coalition success rate is the number of successfully formed coalitions over the number of coalitions initiated, where a coalition is successfully formed when the CPU obtained satisfies the agent's need. We observed the following:

- (1) In all experiments, the reduction in CPU shortage of each agent and the whole system was obvious. Gradually, the CPU resource was reallocated more evenly among agents. The possibility of a CPU shortage decreases and each agent's shortage amount decreases. This shows a coherent, cooperative behavior among the agents.

(2) In all experiments, after some period of time, each agent's CPU allocation converged to an average level (14%). After that, each agent fluctuated around that level.

(3) We also observed that the coalition formation was the most successful for the system as a whole when there were roughly the same number of resourceful and resource-starved agents (Experiment #3) and this type of system also required the least number of negotiations and coalitions to converge.

Initiating agent	Initial CPU allocation (%) / Negotiation success rate (%) / # of negotiations											
	Experiment #1			Experiment #2			Experiment #3			Experiment #4		
	Total CPU=28%			Total CPU=56%			Total CPU=56%			Total CPU=56%		
1	7	15	20	7	45.5	11	7	80	5	7	42.9	14
2	7	25	12	7	55.6	9	7	75	8	14	80	5
3	7	35.3	17	7	41.7	12	21	62.5	8	14	44.4	9
4	7	30	10	35	25	4	21	80	5	21	66.7	3
Average	7	25.4	14.75	14	44.4	9	14	73.1	6.5	14	51.6	7.75

Table 3. Comparison between negotiations in experiments

Initiating agent	Initial CPU allocation (%) / Coalition formation rate (%) / Coalition success rate (%) / # of coalitions															
	Experiment #1				Experiment #2				Experiment #3				Experiment #4			
	Total CPU = 28%				Total CPU = 56%				Total CPU = 56%				Total CPU = 56%			
1	7	28.6	14.3	5	7	80	20	5	7	100	100	2	7	66.7	16.7	6
2	7	60	40	5	7	75	50	4	7	100	100	3	14	66.7	66.7	3
3	7	57.1	28.6	7	7	80	40	5	21	100	66.7	3	14	75	25	4
4	7	40	20	5	35	33.3	0	3	21	100	100	2	21	50	50	2
Average	7	45.8	25	6	14	70.6	29.4	4.25	14	100	90	2.5	14	66.7	33.3	3.75

Table 4. Comparison between coalitions in experiments

4.3 Individual & Cooperative Case-Based Learning

For our investigation of individual and cooperative case-based learning, we conducted two sets of experiments, Comprehensive Experiment A (CEA) and Comprehensive Experiment B (CEB). We carried out CEA to study the effects of individual learning in subsequent cooperative learning and the roles of cooperative learning in agents of different initial knowledge. We performed CEB to investigate the effects of the environment on the agents' learning.

4.3.1 Comprehensive Experiment A (CEA)

We conducted four sets of experiments in CEA as shown in Table 5. The goal of these experiment sets was to investigate how learning differed given different casebase sizes, and how learning differed given different types of initial casebases (some had cases collected from different agents from an earlier run, some had only their own cases). Note that for the following experiments we set the limit on the casebase size as 30 where case replacement started to take place. We used two main parameters to evaluate the casebases: utility and diversity. First, we rank the outcome of each case following the utility values of Table 6.

Experiment Set	A1	A2	A3	A4
ES1	16	16	16	16
ES2	2	16	16	16
ES3	16	16	16	28
ES4	2	10	20	28

Table 5. Experiment sets. For example, in ES1, every agent has 16 cases in its casebase; and so on

Outcome	Utility
success	10
channel_jammed	6
Aborted	5
out_of_time	4
out_of_resources	3
rejected	2
others	0

Table 6. Utility of each outcome for a case

The average utility of the case base is the average product of each case's *TU* value and the utility value of its outcome. The diversity measure of a casebase is computed as the average difference between each pair of cases in the casebase. Three slopes, *sizeSlope*, *diffSlope*, and *utilSlope*, were computed as growth rate between the first learning point and the last learning point, for size, diversity, and utility, respectively. Table 7 shows one example of the results on initiating casebases.

			Size Slope	Utility Slope	Diff. Slope	Ave. Utility Gain	Ave. Diff. Gain
Experiment 1 Combinational-first-combinational-later	A1	indi	0.1207	0.0832	0.0225	0.0718	0.0938
		coop	0.1143	0.0752	0.0173	0.2242	0.0299
	A2	indi	0.0469	0.1112	0.0108	0.1035	0.0779
		coop	0.0385	0.119	0.0064	0.2152	-0.0005
	A3	indi	0.1642	0.086	0.0322	0.0932	0.0719
		coop	0.1667	0.0579	0.03	0.0084	0.0569
	A4	indi	0.1507	0.0784	0.0224	0.0788	0.072
		coop	0.1556	0.0697	0.0218	0.1025	0.1022
	ave		0.1197	0.0851	0.0204	0.1122	0.0630
Experiment 2 Individual-first-combinational-later	A1	indi	0.1429	0.107	0.0447	0.0957	0.1391
		coop	0.1351	0.1106	0.0402	0.1899	-0.0007
	A2	indi	0.1569	0.062	0.0364	0.0614	0.1314
		coop	0.1795	0.0683	0.038	0.0884	-0.0071
	A3	indi	0.2	0.1317	0.0575	0.1339	0.2887
		coop	One point			-0.0974	0.1831
	A4	indi	0.1136	0.0735	0.0318	0.0598	0.145
		coop	0.1154	0.0648	0.0333	0.1858	0.1005
	ave		0.1406	0.0810	0.0374	0.1135	0.0847

Table 7. Utility and difference gains for both Sub-Experiments Exp1 and Exp2, after the second stage, for initiating casebases

Looking at all our results, we observed the following:

- (1) Cooperative learning results in more utility and diversity per learning occurrence than individual learning,
- (2) A small casebase learns more effectively in terms of utility and diversity, but not faster since our learning is problem-driven. A large casebase learns in a similar manner as an average casebase except when it is greater than the preset limit that triggers case replacement.

(3) The initial casebase affects the effectiveness of learning. Both types of learning bring more utility and diversity to an initial casebase previously grown within an agent than one that has been influenced by other agents.

4.3.1 Comprehensive Experiment B (CEB)

The objective of CEB was to see how the learning results changed in different environments, as shown in Table 8.

	Combination of CPU and Tracking Coalitions
ES1	CPU coalitions more often than tracking
ES2	CPU and tracking coalitions similarly frequent
ES3	Tracking coalitions more often than CPU

Table 8. Sub-Experiments setup in CEB

A tracking coalition is more taxing since it requires at least three agents to be successfully formed. Moreover, a tracking task is durational such that it takes time to actually carry out the tracking task. However, a CPU re-allocation task is carried out at a point in time. In addition, a tracking task is highly time-constrained. A coalition has to be formed in time to *catch* the target before the target moves out of the sensor coverage area. Thus, negotiations related to tracking are more difficult to manage. For these three sets of sub-experiments, they had a few things in common: (1) all of them began with the same set of initial case bases, and (2) every sub-experiment ran with the both individual and cooperative learning. We observed the following:

(1) Different environments affect agents' learning behavior. Depending on the frequency of a task and its characteristics, an agent may rely more on individual learning or cooperative learning. For example, if a type of tasks (tracking) is time consuming and durational, then increasing its frequency actually weakens the potential benefit of individual learning and encourages more cooperative learning.

(2) The environments impact the two initiating and responding roles differently, especially for negotiations associated with tough requirements (such as at least three members of a tracking coalition). Since an initiating agent has to shoulder the coalition management and decision making, it is able to learn more diverse and useful cases. But, negotiating as a responder, an agent's responsibility is less and thus considers fewer issues; thus the learning is less impressive.

5 Conclusions

We have described an integrated multilevel approach to coalition formation, using case-based learning and reinforcement learning to learn better tactics as the agent solves a problem, and distributed, cooperative case-based learning to learn improve the agent's knowledge base strategically. We have conducted several experiments and the results have been promising in proving the feasibility of our approach. With learning, our agents negotiate and form coalitions better. Our future work will focus on tying the outcome of an executed coalition (already formed) to the planning stage to improve our strategic learning.

Acknowledgments

This work is partially supported by a grant from the DARPA ANTS project, subcontracted from the University of Kansas (Contract number: 26-0511-0026-001). We also thank Juan Luo for her programming and experiments.

References

- [Jennings, 2002] N. R. Jennings. Coalition formation algorithms for virtual organizations. <http://www.iam.ecs.soton.ac.uk/projects/cfvo>.
- [Marsella *et al.*, 1999] S. Marsella, J. Adibi, Y. Al-Onaizan, G. A. Kaminka, I. Muslea, M. Tallis, and M. Tambe. On being a teammate: experiences acquired in the design of RoboCup teams. In *Proc. 3rd Agents'99*, pages 221-227, Seattle, WA, 1999.
- [Martin *et al.*, 1999] F. J. Martin, E. Plaza and J. L. Arcos. Knowledge and experience through communication among competent (peer) agents, *Int. J. Software Engr. & Knowledge Engr.*, 9(3):319-341, 1999.
- [Martin and Plaza, 1999] F. J. Martin and E. Plaza. Auction-based retrieval, *Proc. 2nd Congres Catala d'Intel.ligencia Artificial*, pages 1-9, 1999.
- [Plaza *et al.*, 1997] E. Plaza, J. L. Arcos and F. Martin. Co-operative case-based reasoning, in G Weiss (ed.) *Distributed Artificial Intelligence Meets Machine Learning*, Lecture Notes in Artificial Intelligence, Springer Verlag, pages 1-21, 1997.
- [Prasad and Plaza, 1996] M. V. N. Prasad, and E. Plaza. Corporate memories as distributed case libraries, *Proc. 10th KAW'96*, pages 1-19, 1996.
- [Sandholm *et al.*, 1999] T. W. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohme. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111(1-2): 209-238, 1999.
- [Sen and Dutta, 2000] S. Sen and P. S. Dutta. Searching for optimal coalition structures. In *Proc. 4th Int. Conf. on Multiagent Systems*, pages 286-292, Boston, MA, July 7-12, 2000.
- [Shehory *et al.*, 1997] O. Shehory, K. Sycara, and S. Jha. Multi-agent coordination through coalition formation. In *Intelligent Agents IV: Agent Theories, Architectures and Languages*, Lecture Notes in Artificial Intelligence, number 1365, pages 143-154, Springer, 1997.
- [Soh and Tsatsoulis, 2001] L.-K. Soh and C. Tsatsoulis. Reflective negotiating agents for real-time multisensor target tracking. In *Proc. of IJCAI'01*, pages 1121-1127, Seattle, WA, 2001.
- [Tohme and Sandholm, 1999] F. Tohme and T. Sandholm. Coalition formation processes with belief revision among bounded rational self-interested agents. *Journal of Logic and Computation*, 9(6):793-815, December 1999.

Dynamics of Coalition Formation in Combinatorial Trading

Yiming Ye and Yuhai Tu

IBM T.J. Watson Research Center
Yorktown Heights, NY 10598, USA

yiming@watson.ibm.com and yuhai@us.ibm.com

Abstract

This paper studies the dynamics of agent mediated combinatorial trading at the macroscopic level. The combinatorial marketplace consists of a retailer who wishes to sell bundles of items, and a large number of agents with different purchasing goals. These agents dynamically form coalitions to exploit the benefits of grouping based on their complementary needs. A novel physics based dynamic equation is proposed to capture the essence of the movements of agents among different sized coalitions. Simulation experiments are performed to study the global behavior of the agents and the effectiveness of the agent mediated combinatorial trading.

1 Introduction

The pervasive connectivity of the Internet and the powerful architecture of World Wide Web are changing many market conventions. Tremendous opportunities for conducting business on the Internet are emerging. Intelligent agents will play a crucial role in electronic commerce where dynamic and heterogeneous interactions between thousands of organizations and millions of individuals are involved. So far, we have already witnessed the involvements of e-commerce agents in traditional business settings. Furthermore, this involvement is re-shaping the way in which business is conducted in areas such as comparison shopping, dynamic pricing, negotiation, auction, and brokerages, to name a few. The automation brought by e-commerce agents will dramatically reduce certain types of frictional costs and time incurred in the exchange of commodities.

In e-commerce, the virtual distance separating producers, wholesalers, distributors, retailers, and consumers has collapsed to near zero. All of the parties involved are faced with rich choices, and it is natural for them to utilize situations to their best advantage. As the population of e-commerce agents increases, automated negotiations among them on behalf of their parties will be prevalent. One of the areas in which agent negotiations will be heavily involved is combinatorial markets, where combinations of goods and services are being traded and efficiently allocated. For example, a trading agent can be

constructed to perform real-time procurement of bundles of complementary goods and services on multiple simultaneous Internet auctions and exchanges. Alternatively, a travel service agent can be constructed to provide combinations of hotel and flight arrangements to potential customers. Recently, research on combinatorial auctions has attracted considerable attention (Fujishima et al 1999; Leyton-Brown et al 2000; Klemperer 1999; Sandholm 1999; Lehmann 1999; Rassenti 1982; Rothkopf 1998; Yokoo et al 2001) due to the sheer interconnectedness of the Internet. One way to analyze combinatorial auctions is to use search algorithms such as branch and bound. This approach can produce optimal solutions, however, it is exponential in the worst-case. An alternative approach, which uses approximation algorithms, is typically polynomial but the quality of the solutions cannot be guaranteed.

This paper addresses the issue in a combinatorial market where the retailer prefers to sell bundles of goods. A large number of buyer's agents dynamically form coalitions to exploit the benefits of grouping based on their complementary needs. The transaction is done at the coalition level such that the price for each buyer's agent is cheaper than if buying independently. Coalition formation has been addressed by researchers from both the game theory community and the multi-agent community. Game theory emphasizes the issues of N-person games formation under different settings and the distribution of the benefits among players (Kraus et al 1991). It concentrates on the stability and fairness issues for given coalitions. Multi-agent research emphasizes the special properties of a multi-agent environment and considers the effects of communication costs and limited computation time on the coalition formation process (Sandholm 1999a; Sandholm 1999b).

We use a physics-based approach to study the dynamic behavior of agents in the combinatorial market where coalitions are involved. There are efforts to study large-scale multi-agent systems using a physics-based approach (Shehory et. al; Lerman et. al. 1999). For example, Lerman and Shehory (Lerman et. al. 2000) propose a physics-motivated mechanism for coalition formation in non-combinatorial markets. The problem addressed in our work is different because we study the issue where the retailer prefers to sell complementary goods. In contrast, the formulation of Ler-

man and Shehory deals with the case where the retailer only has one category of goods. Thus, the agents in our study are heterogeneous because they may be interested in different goods. Agents in their study are homogeneous in the sense that they are all interested in one category of goods.

The remainder of this paper is organized as follows. The next section proposes a combinatorial market model and a physics-motivated dynamic equation for structured coalition formation. Section 3 presents simulation experiments to study the global behavior of the system. Section 4 briefly concludes the paper.

2 Combinatorial Market Model

Figure 1 shows the marketplace to be discussed in this paper. Suppose that the retailer has two types of items to sell: item of type a and item of type b . Suppose that there are two categories of agents for buyers in the Marketplace. An agent of category A is only interested in buying one item of type a . An agent of category B is only interested in buying one item of type b . Each agent represents one buyer and receives instructions from the buyer on what item to register and obtain authorizations from the retailer. Then it can interact with other agents in the marketplace and perform purchase related activities. In the rest of this paper, we assume that in the marketplace there are in total m_A of A type agents and m_B of B type agents registered.

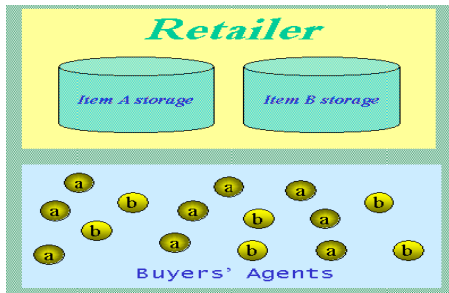


Figure 1: The combinatorial market place discussed in this paper. The small circles represent agents. The character contained in a circle represents the item in which the agent is interested.

There are many ways the retailer may sell its items. It might simply sell an item to a buyer's agent when the agent requests the item. However, retailer can save resources if it can sell items in bundles composed of the two types of items. The retailer can give a cheaper price for each item in the complementary bundle and at the same time still obtain a higher utility value. In order for agents benefit from bundles, they need to form coalitions. We assume that the agents and the marketplace are fully connected. However, agents do not have a global knowledge of the whole situation, and their decisions are based solely on local conditions.

Now we discuss the dynamics of the changes of coalitions in the marketplace. A single agent might join a coalition

in order to obtain a discounted price. Since the agent cannot have a global view of the whole marketplace at a given time instant, the coalition chosen might not be the best one for the agent at that time. An agent may also leave a coalition to look for better opportunities. We assume that the retailer is able to monitor the coalitions of the whole marketplace and to decide when to perform the combinatorial transactions. Our goal is to study the dynamics of the coalition formations in the marketplace and the effectiveness of the marketplace.

We first define the price for a given item associated with a given coalition. We consider a coalition formed by n_A of type A agent and n_B of type B agent. If all of the agents in this coalition voted to perform the coalition transaction (i.e., to buy their items in a bundle), then we use $P_A(n_A, n_B)$ to denote the price the seller prefers to set for an item of type A and $P_B(n_A, n_B)$ to denote the price the seller prefers to set for an item of type B . Suppose that the retailer set the prices as follows:

$$P_A(n_A, n_B) = P_A^{(0)} - \Delta P_A \times n_A + q_A \times |n_A - r_0 n_B|$$

$$P_B(n_A, n_B) = P_B^{(0)} - \Delta P_B \times n_B + q_B \times |n_A - r_0 n_B|$$

Here $P_A^{(0)}$ and $P_B^{(0)}$ are the base prices for item A and item B respectively. The real price for an item is determined by the size of the coalition.

ΔP_A and ΔP_B are the price reduction rates due to grouping for item of type A and for item of type B respectively. In a coalition, as the number of items of a given type increases, the price per item of that type decreases.

The variable r_0 gives the preferred ratio n_A/n_B of the two types of items in a bundle for the retailer. The term q_A gives the price penalty rate for item of type a if the ratio of the items in the coalition does not match the retailer's preferred ratio. Similarly, the term q_B gives the price penalty rate for item of type b . This kind of combinatorial preference is fairly common in traditional commerce as well as in e-commerce. For example, a furniture retailer may prefer to sell a sofa and the accompanying coffee table as a set, rather than sell each piece separately. In this case, the price for a complete bundle (one coffee table and a sofa) may be cheaper than the sum of the prices for the components. The ratio of the number of coffee tables to the number of sofas is important, since the retailer might prefer the ratio to be 1 if there is equal number of table-sofa pairs in the storage. In any case, buyers who are interested in separate pieces can form coalitions to save money. This kind of coalition forming process might be time consuming in traditional commerce, because location and time constraints make it difficult to find complementary buyers. However, it will be totally different in an agent-mediated e-commerce combinatorial marketplace. The sheer interconnectedness of the World Wide Web will enable the e-commerce agents to work at any time from anywhere in cyberspace. Furthermore, agents can negotiate instantly provided that the requirements of the task can be clearly specified, and the interaction protocols can be clearly defined among agents and between agents and the corresponding enterprise system (as is assumed in this paper).

In order to study the dynamics, we use $N(n_A, n_B, \tau)$ to represent the number of coalitions composed by n_A of type A agents and n_B of type B agents at time τ . Different types of coalitions can be represented by a lattice as shown in Figure 2. Where (i, j) represent the value of $N(i, j, \tau)$.

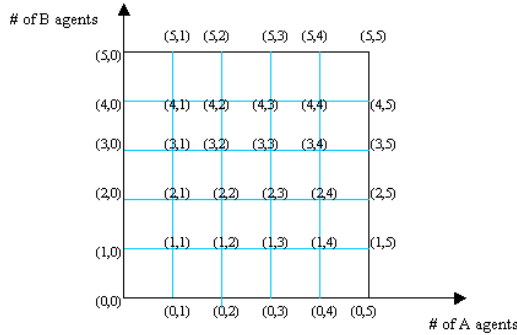


Figure 2. The coalition lattice in a marketplace that contains 5 A type agents and 5 B type agents.

The number of different types of coalitions will change while the system approaches equilibrium. Our goal is to study the value of $N(i, j, \tau)$ and other associated indicators of the effectiveness of the combinatorial markets as time evolves.

We assume that for a given coalition, at most one agent can join or leave at a time. Thus, the only translation in the coalition lattice is between neighbors as illustrated by the following:

$$(1,0) + (n_A, n_B) \Leftrightarrow (n_A + 1, n_B)$$

$$(0,1) + (n_A, n_B) \Leftrightarrow (n_A, n_B + 1)$$

When a category A agent joins a coalition of type (n_A, n_B) , this coalition becomes a coalition of type $(n_A + 1, n_B)$. When a category A agent leaves a coalition of type (n_A, n_B) , this coalition becomes a coalition of type $(n_A - 1, n_B)$. Similar analysis can be applied to type B agent.

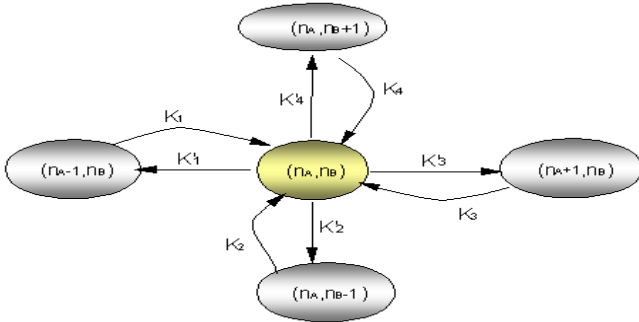


Figure 3. Coalition transitions

Figure 3 shows the transitions that could happen with respect to coalition type (n_A, n_B) when it is within the lattice. Please note that when (n_A, n_B) is at the boundary of the lattice, some of the transitions in Figure 3 will not happen. The dynamics of the transition can be characterized by the following equation (for all the (n_A, n_B) pairs, except $(1,0)$ and $(0,1)$):

$$\begin{aligned} \frac{\partial N(n_A, n_B, \tau)}{\partial \tau} = & K_1 N(1,0, \tau) N(n_A - 1, n_B, \tau) + K_2 N(0,1, \tau) N(n_A, n_B - 1, \tau) \\ & + K_3 N(n_A + 1, n_B, \tau) + K_4 N(n_A, n_B + 1, \tau) \\ & - K_1' N(1,0, \tau) N(n_A, n_B, \tau) - K_2' N(0,1, \tau) N(n_A, n_B, \tau) \\ & - K_3' N(n_A, n_B, \tau) - K_4' N(n_A, n_B, \tau) \end{aligned}$$

Here, K_i and K_i' are rate constants, which depend on the price change for each transition. In the above equation, the total number of agents of the market is conserved during the dynamic process. The single agent pool serves as the source/sink of the system. Please note that $0 \leq n_A \leq m_A$ and $0 \leq n_B \leq m_B$.

The term $K_1 N(1,0, \tau) N(n_A - 1, n_B, \tau)$ gives the rate for a type A agent joins a coalition of category $N(n_A - 1, n_B)$.

The term $K_3 N(n_A + 1, n_B, \tau)$ gives the rate when a category $N(n_A + 1, n_B)$ coalition becomes a category $N(n_A, n_B)$ coalition. In other words, it gives the rate for a type A agent leaves a $N(n_A + 1, n_B)$ coalition.

The term $K_2 N(0,1, \tau) N(n_A, n_B - 1, \tau)$ gives the rate for a type B agent joins a category $N(n_A, n_B - 1)$ coalition. In other words, it gives the rate for a category $N(n_A, n_B - 1)$ coalition becomes a category $N(n_A, n_B)$ coalition.

The term $K_4 N(n_A, n_B + 1, \tau)$ gives the rate for a type B agent leaves a category $N(n_A, n_B + 1)$ coalition. In other words, it gives the rate for a category $N(n_A, n_B + 1)$ coalition becomes a category $N(n_A, n_B)$ coalition.

Other terms in the above equation are related to type B agent and can be similarly explained.

When the value of $N(i, j, \tau)$ is outside the boundary of the lattice, the corresponding term in the above equation will equal to 0. In other words, in the above equation, we set $N(i, j, \tau) = 0$ if $i < 0$, or $j < 0$, or $i > m_A$, or $j > m_B$, or $i = j = 0$.

Now, we need to study the rate constants involved in the dynamic equation.

One simple model would be to let all of the escape rates be the same (i.e., the rate at which an agent leaves a coalition, a.k.a., the opportunist's rates):

$$K_3 = K_4 = K_1' = K_2' = K_{\text{escape}}$$

We assume that the rate an agent joins a coalition is closely related to the changes in price for this coalition. Let the change in price serves as energy in Boltzman distribution.

K_1 is a constant that is related to the transition of a coalition from $(n_A - 1, n_B)$ to (n_A, n_B) . Thus, this constant is only related to type A agent.

At point $(n_A - 1, n_B)$, we have the price for a type A agent:

$$P_A(n_A - 1, n_B) = P_A^{(0)} - \Delta P_A \times (n_A - 1) + q_A | (n_A - 1) - r_0 n_B |$$

At point (n_A, n_B) , we have the price for an A type agent:

$$P_A(n_A, n_B) = P_A^{(0)} - \Delta P_A \times n_A + q_A | n_A - r_0 n_B |$$

The price change is:

$$P_A(n_A, n_B) - P_A(n_A - 1, n_B) = -\Delta P_A + q_A | n_A - r_0 n_B | - q_A | (n_A - 1) - r_0 n_B |$$

Thus, we can set the transition rate as:

$$K_1 = K_0 \times e^{\Delta P_A - q_A (|n_A - r_0 n_B| - |(n_A - 1) - r_0 n_B|)}$$

Here, K_0 is the base rate in the above equations.

Similarly, we can set other rates:

$$K_3' = K_0 \times e^{\Delta P_A - q_A (|(n_A + 1) - r_0 n_B| - |n_A - r_0 n_B|)}$$

$$K_2 = K_0 \times e^{\Delta P_B - q_B (|n_A - r_0 (n_B + 1)| - |n_A - r_0 n_B|)}$$

$$K_4' = K_0 \times e^{\Delta P_B - q_B (|n_A - r_0 (n_B + 1)| - |n_A - r_0 n_B|)}$$

To analysis the dynamics of the above system, we can study, under different values of parameters, the evolution of $N(n_A, n_B, \tau)$ and other indicators of the marketplace as a function of τ and at the equilibrium state. Based on the calculated derivatives, we can update a coalition using the following formula (for all the (n_A, n_B) pairs, except $(0, 1)$ and $(1, 0)$):

$$N(n_A, n_B, \tau + \Delta\tau) \leftarrow N(n_A, n_B, \tau) + \left[\frac{\partial(n_A, n_B, \tau)}{\partial\tau} \right] \Delta\tau$$

During the update process, the time step $\Delta\tau$ should be adaptively selected such that it is small enough and the updated results for any coalitions should not be less than 0.

The updated value of $N(0, 1, \tau + \Delta\tau)$ and $N(1, 0, \tau + \Delta\tau)$ can be obtained from the following equation after all the other $N(n_A, n_B, \tau + \Delta\tau)$ are updated:

$$\sum_{n_A, n_B=0}^{m_A, m_B} n_A N(n_A, n_B, \tau + \Delta\tau) = m_A$$

$$\sum_{n_A, n_B=0}^{m_A, m_B} n_B N(n_A, n_B, \tau + \Delta\tau) = m_B$$

3 Simulation Results

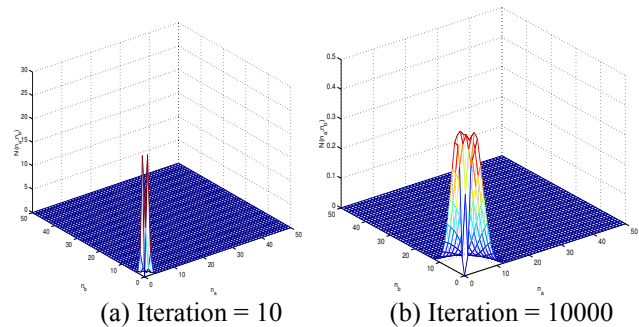
We have conducted various experiments to test the marketplace. At first, we set all the values of $N(n_A, n_B, 0)$ equals to 0, except that $N(1, 0, 0) = m_A$ and $N(0, 1, 0) = m_B$. In our experiments, we set $m_A = m_B = 50$, and $K_0 = 1$. Please note that $P_A^{(0)}$ and $P_B^{(0)}$ have no influence on the derivatives.

In the first set of experiments, we study the evolution of the coalitions in the market as a function of time. Table 1 shows the parameter values that are used.

K_{escape}	r_0	ΔP_A	ΔP_B
0.0005	0.5	0.1	0.1
q_A	q_B	k_0	
0.05	0.05	0.006	

Table 1

Figure 4 (a), (b), (c), and (d) show the evolutions of coalitions as a function of the iterations. Notice that at beginning, most of the agents are in coalitions that contain small number of type A and type B agents. Then, as a result of dynamic movements of agents, the number of coalitions with small size decreases and the number of coalitions with large size increases until equilibrium is reached. Further note that the distribution of A type agents and B type agents is not symmetric. The reason for this is that the ratio $r_0 = 0.5$, rather than $r_0 = 1$. Thus coalitions with twice as many A agents as B agents receive the least price penalty. Although intuitively most of the agents should stay in large coalitions at equilibrium, Figure 5 shows that this is not the case. This result is different from that of (Lerman et al., 2000) in which most agents join the largest coalitions. One reason for this difference is that the price settings in combinatorial markets are quite different from those in non-combinatorial markets. Our situation is much more complex, and the coalition size is only one of the factors that influence the movements of agents. The escape rate is a parameter that has a bigger effect on the size distribution of coalitions. The greater the escape rate, the more the agents escape from the larger coalitions. Compare Figures 6(a), 6(b), and 5(d), we can notice that the number of the small sized coalitions is bigger when the escape rate is larger.



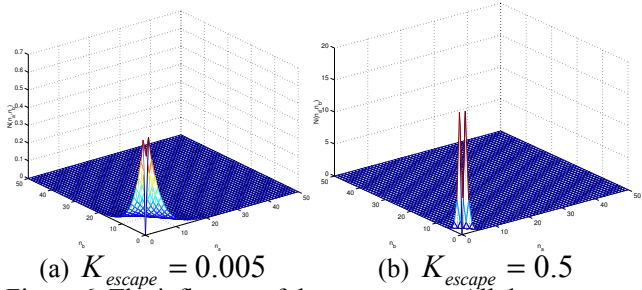
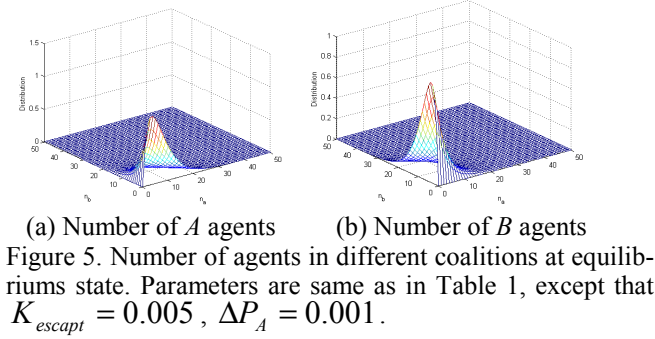
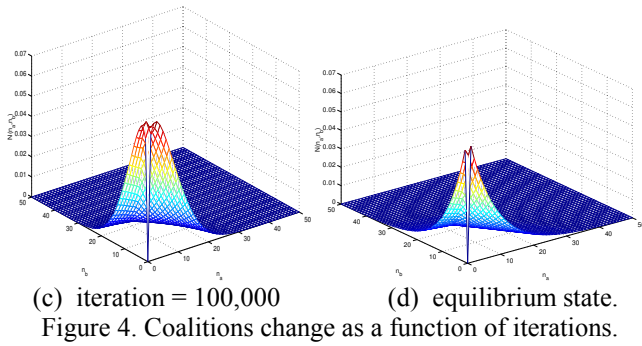


Figure 7 shows the value of entropy as a function of the iterations. We use the same values for parameters as those in Table 1, except that $K_{escape} = 0.005$ and $r_0 = 0.1$. We set the initial prices as $P_A^{(0)} = 2.85$ and $P_B^{(0)} = 5.1$. Please note that these initial prices are set to make sure that all the prices are positive for any coalitions at any given moment.

To calculate the entropy with respect to the number of coalitions of various sizes, we first calculate the probability that a given sized coalition within the population of all the coalitions:

$$p_{N(i,j)} = \frac{N(i,j)}{\sum_{i=0,j=0}^{i=m_A,j=m_B} N(i,j)}$$

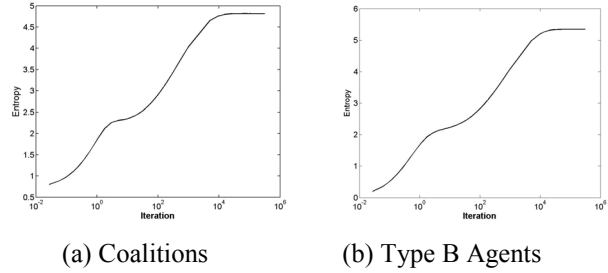
The entropy with respect to coalition sizes is given by:

$$H_N = - \sum_{i=0,j=0}^{i=m_A,j=m_B} p_{N(i,j)} \log(p_{N(i,j)});$$

Similarly, the entropy for type B agent within different coalition sizes can be calculated as:

$$H_B = - \sum_{i=0,j=0}^{i=m_A,j=m_B} \frac{j * N(i,j)}{50} \times \log\left(\frac{j * N(i,j)}{50}\right);$$

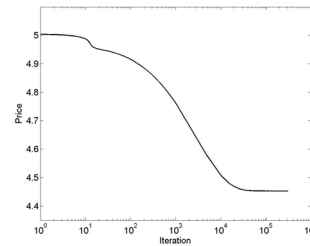
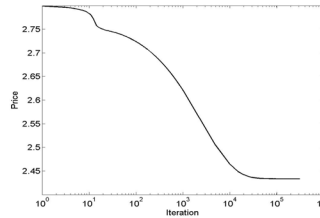
We can notice that the values of both entropies increase with the iteration until the system reaches its equilibrium.



The average price per type A item and the average price per type B item are principal indicators of the effectiveness of the proposed combinatorial marketplace. These average prices can be calculated with the following formulae respectively.

$$\frac{\sum_{i=0,j=0}^{i=m_A,j=m_B} \{P_A(i,j)[i \times N(i,j)]\}}{m_A}; \quad \frac{\sum_{i=0,j=0}^{i=m_A,j=m_B} \{P_B(i,j)[j \times N(i,j)]\}}{m_B}.$$

Figure 8 and Figure 9 show the evolution of the average prices of a type A item and that of a type B item. We can notice that the average prices keep decreasing until the system reaches the equilibrium. This demonstrates the advantages of using agents in combinatorial markets.



4. Conclusions

In this paper, we propose a theoretical framework for coalition formation in combinatorial transactions. A novel physics-based dynamic equation is proposed to capture the essence of the movements of agents among different sized coalitions as driven by the price differences of these coalitions and by the opportunities of joining better coalitions. We perform simulation experiments to evaluate the proposed combinatorial marketplace and the results show that the marketplace can reduce the average prices for the buyer's agents if opportunities are fully explored. Our approach towards combinatorial trading is different from previous work because we address the issue from a macroscopic point of view.

Our framework is generic, although we make many simplifying assumptions in our current proof of concept model so as to make the simulations easier. We are exploring many more issues with our current model as a starting point. In our model, the rate at which an agent joins a coalition depends only on the price reduction for that agent. Any request to join a coalition is accepted. Another way of modeling the market is to emphasize the coalition an agent is about to join. For example, the rate can be determined by considering how the changes in coalition size change the prices for the "old" members of the coalition, or the total price for the "old" members of the coalition etc. In our current model, we simply assume that the escape rates are the same for all the agents. It is interesting to study the situation when the escape rates depend on different types of agents and different sizes and structures of the coalitions. In our current model, the ratio of the two types of agents within a coalition is an important factor in determining the price. It is worth studying the coalition market without perfect ratio of goods. In our current model, the price given by the retailer is fixed. It would be interesting to study the situation where the retailer can perform dynamic pricing according to its knowledge of the coalitions. Our model addresses a very simple combinatorial market where there is only one retailer and two different products. It would be interesting to study the dynamic behaviors in a marketplace where there are a large number of retailer's agents that might form coalitions to offer complementary products and a large number of buyer's agents that might form coalitions to explore the benefits of grouping. It would also be interesting to study the coalition dynamics of open combinatorial markets when agents are allowed to flow in and out of the marketplace. It is important to study the qualitative behaviors of the proposed model throughout in the parameter space. The simulation based on our current simple model already shows some interesting phenomena. We are convinced that the behaviors of the system will be quite rich when more complex models are used.

Acknowledgments

We would like to thank Eric Harley and the IJCAI reviewers for their valuable comments.

References

- [Fujishima et. al 1999] Yuzo Fujishima, Kevin Leyton-Brown, and Yoav Shoham. Taming the computation complexity of combinatorial auctions: Optimal and approximate approaches. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 548–553, 1999.
- [Klemperer 1999] Paul Klemperer. Auction theory: A guide to the literature. *Journal of Economics Surveys*, 13(3):227–286, 1999.
- [Kraus et. al 1991] S. Kraus and J. Wilkenfeld. Negotiation over time in a multi-agent environment: preliminary report. In *Proc. IJCAI'91*, pages 56–61, Australia, 1991.
- [Lehmann et. al 1999] Daniel Lehmann, Liadan ItaO'Callaghan, and Yoav Shoham. Truth revelation in approximately efficient combinatorial auction. In *Proceedings of the First ACM Conference on Electronic Commerce (EC-99)*, pages 96–102, 1999.
- [Lerman et. al 2000] K. Lerman and O. Shehory, Coalition Formation for Large-Scale Electronic Markets, in *Proceedings of International Conference on Multi-agent Systems*, 2000.
- [Leyton-Brown et. al 2000] Kevin Leyton-Brown, Mark Pearson, and Yoav Shoham. Towards a universal test suite for combinatorial auction algorithms. In *Proceedings of the Second ACM Conference on electronic Commerce (EC-00)*, pages 66–76, 2000.
- [Rassenti et. al 1982] Rassenti, S. J.; Smith, V. L.; and Bulfin, R. L. 1982. A combinatorial auction mechanism for airport time slot allocation. *Bell J. of Economics* 13:402–417.
- [Rothkopf et. al 1998] Rothkopf, M. H.; Pekeć, A.; and Harstad, R. M. 1998. Computationally manageable combinatorial auctions. *Management Science* 44(8):1131–1147.
- [Sandholm 1999a] Tuomas Sandholm. An algorithm for optimal winner determination in combinatorial auction. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 542–547, 1999.
- [Sandholm 1999b] T. Sandholm. Negotiation among self-interested computationally limited agents. Ph.D. Thesis, University of Massachusetts, Amherst, MA, USA, 1996.
- [Shehory et. al 1999a] O. Shehory and S. Kraus. Feasible formation of coalitions among autonomous agents in non-super-additive environments. *Computational Intelligence*, 15(3): 218–251, August 1999.
- [Shehory et. al 1999b] O. Shehory, S. Kraus, and O. Yadgar. Emergent cooperative goal satisfaction in large-scale automated-agent systems. *Artificial Intelligence*, 1999.
- [Weiss 1999] G. Weiss eds. *Multiagent Systems – A Modern Approach to Distributed Artificial Intelligence*, 1999
- [Yokoo et. al 2001] Makoto Yokoo, Yuko Sakurai, and Shigeo Matsubara. Bundle Design in Robust Combinatorial Auction Protocol against False-name Bids, 17th International Joint Conference on Artificial Intelligence (IJCAI-2001).

MULTIAGENT SYSTEMS

EMERGENT BEHAVIOR

Biologically-Inspired Self-Assembly of Two-Dimensional Shapes Using Global-to-Local Compilation

Attila Kondacs, MIT Artificial Intelligence Laboratory, kondacs@mit.edu

Abstract

In this paper, we present a programming language approach for the assembly of arbitrary two-dimensional shapes by decentralized, identically-programmed agents. Our system compiles a predetermined global shape into a program that instructs these agents to grow the shape via replication and location-based control mechanisms. In the global-to-local compilation phase, an input shape is decomposed into a network of covering-discs. The disc network parameterizes the agent program, a biologically-inspired framework allowing agents to amorphously produce the shape using replication and local interaction. Our system is robust to random agent failure, and regenerates in the event of region death.

1 Introduction

Biological cells assemble into complex structures with impressive robustness. They exhibit advanced global behaviors without centralized control or strict sequentiality of execution, despite random cell death or malfunction. In contrast, modern artificial systems are highly centralized and sequential, rendering them vulnerable to failure and encouraging the production of more complex, precise components. Our ability to embed millions of tiny chips with sensors [Abelson *et al.*, 1995; McLurkin, 1999], or program biological cells to serve as logic gates [Weiss *et al.*, 1999; 1998], marks a shift in technology to a reliance on cheaper, decentralized parts [Butera, 2001]. Traditional programming techniques are no longer sufficient for engineering systems, such as self-assembling nanostructures, to exhibit a robustness comparable to biological cells. How will we program large numbers of unreliable, locally-interacting parts to engage in coherent behaviors?

In this paper, we present a programming language approach to designing self-assembling systems [Nagpal, 2001; Coore, 1999]. We use morphogenesis and developmental biology [Wolpert, 1998; Bard, 1990] as motivation for organizing robust local behavior. However, unlike current approaches to designing emergent systems, the general principles are formalized as a programming language – with explicit primitives, means of combination, and means of abstraction – thus

providing a framework for the design and analysis of self-organizing, spatially-controlled systems.

2 The Problem

Here, we apply the above approach to the synthesis of arbitrary two-dimensional shapes from tiny distributed computing units *amorphously*. An amorphous computing medium [Abelson *et al.*, 2000; 1999] is a system of irregularly placed, locally interacting, identically-programmed, asynchronous computing elements. We can model these computing units as *cells* with identical DNA. These cells have modest computing power, with limited ability to retain local state, and they may die. The communication radius is large relative to the radius of a cell, yet small compared to the size of the whole structure. Despite all these limitations, we would like the cells to exhibit highly complex collective behaviour: our goal is to compile a predetermined global shape into an amorphous cell program that instructs the cells to grow into the shape. In other words, starting from one cell, the cells continuously replicate into a configuration that geometrically approximates the input shape.

The global-to-local compilation proceeds in two phases: first, the input shape is decomposed into a semi-efficient packing of covering discs; second, adjacent discs are linked into a bidirectional network using a local, relative coordinate system (“reference points”). This choice of representation permits the structure to be produced in the growing phase amorphously by cells whose internal program relies on the recursive execution of only two key primitives: locating the reference points of discs, and growing into a disc. The cells use replication, messaging, and competition – mechanisms inspired by cell differentiation and morphogenesis [Lawrence, 1992; Wolpert, 1969] – to achieve these primitives robustly.

The system compiles and grows any connected two dimensional shape. The size of the compiled cell program is linear in the number of covering discs of the shape. The growth process is robust to random cell-death.

3 The Model

Our objects of discourse are locally-interacting, asynchronous computing units, here referred to as *cells*. Cells have a center location and a fixed radius r . A cell can replicate, and will place a given child between $2r$ and $4r$ away

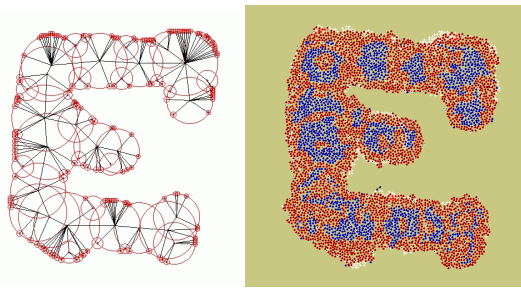


Figure 1: Compiled disc network with the spanning tree and the fully grown versions of a letter E shape.

from its center. Cells are identically-programmed, differing only by a limited amount of local state. They have limited computing power and are vulnerable to random death. They have no a priori knowledge of their positions or orientations. Our model for a cell substrate is analogous to living tissue: cells are tightly packed, cannot move around nor can they overlap.

3.1 Cell Primitives and Communication

A cell can execute five basic *low-level actions* actions. It may:

1. Change its internal state
2. Emit gradient messages
3. Hand messages to immediate neighbors
4. Reproduce
5. Die.

A *gradient message* has a specified amplitude and carries a numeric id. Amplitudes have an upper bound of 30 times the cell radius. Just as chemicals diffuse in fluid, a gradient message's strength decays with distance from the source [Nusslein-Volhard, 1996]. In our model, the decay is linear, and without noise. The amplitude of a given message is known by all cells, thus the strength of a received message will determine the distance from the source. In this way, cells can accomplish differentiation through *triangulation*: after receiving gradients and observing distances from at least three nearby non-collinear cells, a cell can determine its position relative to the source cells.

A *local message* carries a fitness value, a reference point id, and a state bit. Local messages are delivered only to *immediate neighbors*, or "touching" cells: cells within a fixed, short radius. There is always some local message exchange between immediate neighbors. Note that the lack of noise in messaging is a simplifying and unrealistic assumption in amorphous computing.

The low-level actions are combined to form the following *high-level actions*:

1. Grow a disc of cells
2. Activate or deactivate a cell as a competitor for a role
3. Turn reference point role on or off
4. Hand over a role to a neighbor
5. Triangulate another set of potential role holders

High-level actions are always associated with a *reference point role*. Reference points are cells that have been designated as "coordinates" of a local grid; these cells must exude gradient messages that allow nearby cells to triangulate their relative positions. Reference point roles can be *activated* and *deactivated* in any cell. Reference points will be discussed in great detail in sections 4 and 5. When such a role is activated, the cell enters a *local competition* for that role with other activated cells in its local neighborhood in an attempt to determine a unique role-holder for the given reference point. The details of the local competition are discussed in section 5.1.

The execution model of the cell is analogous to a Turing machine. The cell *transfer function* takes the internal state and the messages heard, maps these into a sequence of high-level cell actions, and translates these into a sequence of low-level cell actions. In each *cell-step*, the cell determines the next in the sequence of low level cell-actions, using the output of the transfer function, and then executes this action.

Asynchronous execution is achieved by using 5 threads, and each thread cycles through 70% of the cells in random order. A scheduler ensures that any cell that is more than 8 cell-steps ahead of the average number of cell-steps will not be chosen for execution in the next cycle, and that any cell that is more than 8 steps behind the average will certainly be included. The rest of the 70% of the cells are chosen randomly. The difference in the number of cell executions between any two cells cannot exceed $2 * (5 + 8) = 26$ cell-steps.

4 Compilation

In our system the two-dimensional shape is compiled directly into the cell program. The complexity of the program grows linearly with the complexity of the input shape - that is the number of discs the compiler covers the shape with. This approach differs significantly from models based on cellular automata [Margolus, 1996], in which local rules are constructed empirically and creation of complex shapes would be intractable; and from those based on evolution [Forrest and Mitchell, 1993; Mitchell *et al.*, 1994b; 1994a], in which the relationship between local and global behaviors is not well understood.

The compilation is a regular sequential (non-amorphous) procedure whose input is the pixel map of the 2 dimensional shape, and whose output is the amorphous cell program or transfer function. It operates in six simple stages. In the first stage, we select a *disc-covering* for the input shape. Second, we form a graph using our discs for our vertex set, in which two vertices form an edge if they intersect. A spanning tree is then selected from this graph; pairs of discs will be called *neighboring discs* if they are connected in the spanning tree.

Third, we locate the positions for reference points within each disc. In order to do this, we pick an arbitrary global orientation. Then we find, along with the *disc center*, the north, south, east and west *cardinal reference points* (i.e. points on the circumference of the discs that lie north, south, east and west of the the disc centers) in each disc. We proceed to locate the *intersection points* of neighboring discs in the spanning tree. Two problems can arise at this stage of the compi-

lation. (1) Consider a neighboring pair of spheres (A, B). If the intersection points of A and B are too close to each other the reference points in A and B used for triangulating reference points of B may be almost collinear, which makes the outcome of triangulation procedure ambiguous. To remedy this, if this problem arises we locate an extra reference point called the *temporary disc center*. (2) If the center of disc B is not inside the fully grown disc A , there may be no cells where B 's center is supposed to be. In order to place cells at B 's center location in the growing phase, a sequence of temporary discs are grown, whose centers hand over the temporary disc center role called *line reference point* to a better positioned neighboring cell as soon as there is one. During the compilation phase the first line reference point is located. Sections 5.3 and 5.4 give a more detailed description of how the growth phase proceeds in these two cases.

Fourth, we designate *activating sets* and *deactivating sets* for each reference point. An activating or deactivating set is a set of reference point messages whose combined presence and/or absence can activate or deactivate another reference point role.

Next the (de)activating sets are converted to *boolean statements* that specify exactly when a cell should consider activating or deactivating a reference point role. The boolean language is the heart of the cell transfer function. It contains only one primitive: a message from a reference point can be heard. The language has the usual *And*, *Or* and *Not* constructors, as well as two additional constructors: *AtLeast*(k, set) and *AtMost*(k, set). These express that at least (or at most) k of the statements in *set* must (may) be true. The primitives are combined, using the constructors, to form compound statements, thus forming a concise description of the complex geometric and logic relations existing among reference points. The boolean statements would, for example, compactly describe the following simplified scenario: A cell should consider competing for the north cardinal point role in disc B if: (it does not hear other north cardinal point messages from B) *And* (it hears a message from B 's center) *And* (it hears at least three other cardinal points of disc B). A more complete example boolean statement is listed in section 5.2.

In the last, sixth stage of the compilation, we calculate the ideal message strengths associated with messages in the boolean statements. The latter distances, together with the compound boolean statements, make up the transfer function.

5 Growing Phase

Each cell executes the same cell program. The compilation procedure outputs this program from a given connected two-dimensional shape. The program consists of the transfer function and an interpreter. The cell maps the combination of its internal state and received messages to a sequence of high-level actions. The cell proceeds to translate the high-level actions into a sequence of low-level actions that determine (a) which messages a cell must prepare to send, (b) whether the cell should produce a child, (c) the internal state it must assume during the next cell-step. While over a long period of time all of the cells' average running speeds are the same, the cell-steps are not synchronized, and the expected dura-

tion of a step varies among cells. Collectively, cells execute two major operations recursively: (a) growing a disc via cell replication, and (b) triangulating target cells that will assume reference point roles.

Let's call line messages, and messages from disc centers and temporary centers, *grow messages*. In every cell-step of its existence, the cell listens for grow messages. If it detects one, it will attempt to reproduce, placing a cell in the surrounding empty space. It will suspend replication after 20 unsuccessful trials until there is a new opening in the surrounding area, i.e. until a nearby cell dies. Thus the gap in the shape left by a dead cell is filled by the offspring of the expired cell's neighbors. In case the cell does not hear any grow messages for 50 steps, the cell dies and is removed from the space.

In each cell-step the cell processes the gradient messages it hears, let us call this its *in-message set*. It selects the statements from the set of boolean statements stored in the transfer function that evaluate to true. To evaluate a boolean statement the cell should match its in-messages against the list of messages that are required to be present or missing by the boolean. Each of the true boolean statements asserts that a reference point role should activate or deactivate in this cell. The cell makes the final decision of whether to (de)activate as a competitor for the reference point role based on the proximity of the perceived strengths of the messages (that are required to be heard by the boolean) to the ideal message strengths stored in the transfer function. In the next cell-step the activated cell enters (and the deactivated cell exits) the competition for holding this role with other activated cells in its neighborhood.

5.1 Local Competition for a Reference Point Role

Activated (or competing) cells keep track of 4 things: whether they have settled into a steady *winner* or *loser* state from the *competing* state; their own *fitness value*; the best received fitness value. A competing cell will send local messages to its neighbors with (a) the reference point id for which it is competing, (b) a state bit showing whether it has stabilized, and (c) the best fitness value it observes (possibly its own). The cell's active neighbors will keep doing the same. Thus if its fitness value is better than anyone else's, this value will spread across its competing neighbors and its neighbors' neighbors, and so forth. If a cell has had a better fitness score than any of the fitness values it sees in the local messages for 40 steps, it stabilizes as a winner. This triggers all its competing neighbors to settle (end the competition) as losers, which will induce all their competing neighbors (and in turn their competing neighbors neighbors, and so on) to stabilize as losers. Eventually, all the cells competing for this particular reference point role that can be reached via a sequence of immediate active neighbor relationships from the winner cell stabilize as losers. Once a cell has stabilized as a winner, it becomes the holder of the particular reference point role for which the cells have competed. It instantly sends out the corresponding reference point gradient message. In the event that there is a topologically disconnected pocket of cells still competing for the same role, the presence of the reference point gradient will suppress any further competition

with lower leading score. If the gradient message or the winner disappears the passive loser cells resume competition for the reference point role until a new leader stabilizes. Thus the competition never ends, but reaches equilibrium, and resumes competition as soon as the equilibrium is disturbed. The 40 step wait before a cell stabilizes ensures that the local messages with the maximum score have time to spread through the competing cells in spite of the asynchronous execution of cell-steps.

If various competing cells die, local competition will, with high probability, finish successfully. We may consider the following cases. If a non-leading cell dies its loss does not disturb the outcome of the competition. If a non-stabilized leading cell dies, the neighboring cells will sense the absence of the local message with the winner state bit and will start broadcasting a special local message containing the reference point role. This will spread through the competing cells and restart the local competition for this particular role. It is unlikely that all the cells surrounding the now missing leading cell die before they could trigger a new competition. However, if this does occur, the rest of the competing cells that have stabilized in loser states will realize that the end of the competition for this role has not been followed by a corresponding gradient message from the winner in 50 cell-steps, and will automatically resume competition for this role.

5.2 Triangulating the Reference Points of a Disc

Assume that the fully grown disc A has a neighbor, disc B , that has not been grown yet. Let's denote A 's center reference point $A.C$, cardinal reference points $A.N$, $A.S$, $A.E$, $A.W$, (north, south, east and west respectively) and the intersection points with disc B , $AB.I1$ and $AB.I2$. I will describe a possible sequence of reference point activation path that will lead to a fully grown B disc with its reference points in place.

An outline is as follows: first triangulate B 's center, $B.C$, then grow disc B , proceed to triangulate the cardinal points of B . Once all the cardinal points of B have stabilized, triangulate the intersection points of B with its neighboring discs simultaneously, and recursively do the same with B 's yet un-grown neighbors.

In this section, we will consider the ideal case: when A contains $B.C$. Using A 's cardinal point gradient messages $B.C$ can be located easily. In this case all cardinal points of B that lie inside A are triangulated using reference points of A . We need to consider three cases. (1) Assume that there are at least two such cardinal points of B that lie in A . After these have stabilized, they, along with $B.C$ are used to triangulate the remaining cardinals of B . (2) If there is only one cardinal, say $B.E$, in A then after this one has stabilized, first the opposite direction cardinal, $B.W$ is located: $B.E$ and $B.C$ are used to find the cell in B that is furthest away from $B.E$, but is still in B . Next the cardinal point that is closer to the intersection points $AB.I1$ and $AB.I2$ is triangulated, using $AB.I1$, $AB.I2$, $B.E$, $B.W$ and $B.C$. Finally, the last remaining cardinal point of B is triangulated using the 3 other cardinals and the center of B . (3) If there is no cardinal in A then $AB.I1$, $AB.I2$, and $B.C$ are used to locate the cardinals of B that are closest to $A.C$, then the opposite cardinals using the same procedure as described above.

The scheme described so far is the most preferred method of growth, i.e. the one followed when nothing is damaged. However, in the event that cells die unexpectedly, or if some of the cardinal points are covered by other complete discs, for example, the order may be different. This complex growth order is determined by a sequence of boolean statements in the transfer function of the cell program. Consider the following simplified statement for activating $B.N$:

```
And[Not[B.N];
    B.C;
    Or[At_least 3 (cardinals_in_disc B);
        (opposite_cardinal_point_in_disc North B);
        Or[map
            (function disc →
                And[intersection_points_of_discs B disc]
                (neighboring_discs_of_disc B)]]]]
```

The cell will check if the statement evaluates to true on the presently observed in-message set. It finds the first boolean in the normal form that evaluates to true, which we will call the *satisfied boolean*. Thus, in this example, it will most prefer to use, along with $B.C$, the 3 other cardinals of B to locate $B.N$, then $B.S$, then the two intersection points with A , or some other neighbors. If the first, most preferred boolean is not satisfied (that is, the messages that would satisfy it are not heard), perhaps because some cells died, it will try to use the second most preferred boolean, then the third, and so on. If there is no satisfied boolean, the cell will wait, thus the triangulation is unaffected by asynchronous execution of cell cell-steps.

5.3 Handing down a Reference Point Role

If $B.C$ is outside disc A , the cells in A proceed to triangulate the line reference point $B.L1$ midway between $AB.I1$ and $AB.I2$ on the circumference of A . As soon as $B.L1$ has stabilized, it starts exuding a temporary disc center message with range equal to the distance between $B.L1$ and $AB.I1$. This induces cells in the range to reproduce. Cells around $B.L1$ constantly communicate their fitness score for the role of $B.C$ via local messages to the cell holding $B.L1$. Once a cell with a better fitness than that of $B.L1$ has showed up, $B.L1$ role is replaced by a temporary disc center role, $B.T1$. At the same time the cell holding this role sends a local message to the better scoring neighbor that turns on the role $B.L2$ in it. The latter cycle continues until a cell can find no immediate neighbor that scores better than itself for 50 cell-steps. At this stage the $B.L$ role is turned off and the $B.C$ role stabilizes in its place, the appropriate center gradient message is sent out, which in turn turns off all temporary centers in B . Note that during this passing on of reference point roles the local competition procedure plays no part other than stabilizing $B.L1$. The procedure is still robust because if a cell with a $B.L$ role dies, and as a result, there are no more $B.L$ (or $B.C$) messages observed by the cell, then the temporary center $B.Ti$ that hears no larger id $B.Tj$ temporary center message will turn into a $B.Li$, and the handing down procedure restarts from that point.

The first disc is special as it cannot be grown by any other disc. The latter reference point role handing over procedure

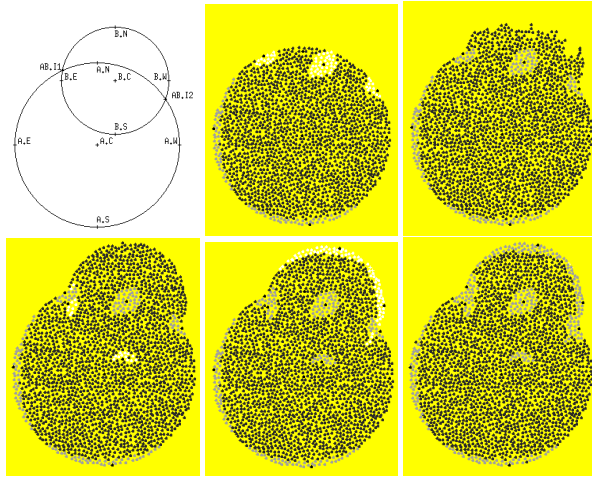


Figure 2: 1: Center, cardinal and intersection reference points of discs A and B. 2-6: Growing phase. Grey cells are stabilized loser competitors, white cells are still competing for a role. Black cells surrounded by white or grey cells are leading competitors or stabilized reference points. Local competition for the intersection points (AB.I1, AB.I2) and B's center (B.C) (2); growing disc B (3); local competition for south (B.S) and east (B.E) (4) and north (B.N) and west (B.W) (5) cardinal reference point roles in disc B; and the grown shape (6).

is used to pick the first, north, and the second, east cardinal points in the first disc. The orientation of the first north cardinal is chosen randomly and the orientation of the first east cardinal point will randomly choose between the east and west directions relative to the already picked north. On the figures these are externally forced to be up and left.

5.4 Temporary Discs

If the distance between $AB.I1$ and $AB.I2$ is too small relative to the radius of A or the cell size, or either of the radii are too small relative to the other one or to the cell size, then triangulation of the cardinals may be ambiguous. In these cases, if simple geometric checks confirm the ambiguity, a temporary helper disc, T , is grown. T 's center, $T.C$ is at the midpoint of the arc between $AB.I1$ and $AB.I2$ on A 's circumference. After $T.C$ has stabilized, if T does not cover B completely, T 's intersections with B , $TB.I1$ and $TB.I2$ are triangulated. The diameter of T is chosen such that the distance between $TB.I1$ and $TB.I2$ is large enough to avoid the original ambiguity. If T contains B , $TB.I1$ and $TB.I2$ are picked on the circumference of T . In either case $TB.I1$ and $TB.I2$ are used in place of $AB.I1$ and $AB.I2$ in all the above discussed procedures. Once at least 2 cardinal points and the center of B have stabilized, $T.C$ is turned off, and any cells that fall outside the ranges of all the presently existing grow messages die in 50 cell-steps.

6 Overall Robustness and Simulation Results

One of the main goals of this research is to explore how the robustness of computing can be improved using spatial control mechanisms. We ask ourselves, to what types of failures, and to what degree of failure, is our system robust? Furthermore, what imperfections can we introduce that will expose vulnerabilities in our system?

Let us consider the robustness of our parallel system in the face of random cell death, that is, if we continuously exterminate cells at random with a certain frequency. Whether the shape can effectively continue to grow depends on the rate at which we cause cells to die. Certainly, the rate of cell death can not be faster than the rate of regeneration.

All high level actions of cells, including local competition, role handing, triangulation, disc creation, can function in spite of a high death rate. The robustness of the competition procedure against cell death was discussed in section 5.1. Let us discuss some of the other various aspects of how our system can accomodate cell death.

If a cell with a cardinal, intersection point, disc center or temporary center reference point role dies, its missing gradient message will trigger a new local competition among all the cells competing for the role, and this will eventually lead to the selection of a new reference point holder cell. If a line-reference point dies, the role-handing down procedure will backtrack to the last still available line-reference point. If the first disc's center dies before the first disc's cardinal reference points stabilize a neighboring cell will resume the first disc center role.

If multiple reference points vanish simultaneously, it is possible that an entire set of cells competing for a reference point role will stop competing for the role. This is very unlikely for two reasons: first, deactivation will take place only after the boolean statement of the satisfying set in the deactivating set has evaluated true for 100 consecutive cell-steps. Therefore, some reference point role holders and thus their gradient messages will be missing for 100 steps. In this many steps, however, they will be replaced by new role holder cells, which will turn the booleans in the deactivating set false before they can take effect. Second, each reference point role has multiple activating sets, each of which must be missing for the role to be deactivated.

In case such a deactivation occurs to a disc center, it will cause the cells only in this disc to die, after a 50 cell-step wait period. Let's call the discs that can regenerate all their neighbors *regenerating discs*. The compilation is designed so that any disc with radius larger than 8 times the cell radius is a regenerating disc. Assume a whole disc of cells has been destroyed because all the cells competing for the center reference point have died. As long as at least one of the regenerating disc neighbors of the missing disc is relatively unharmed, the disc will regrow under normal growth procedures. On the other hand, to make regrowth impossible, all disc centers and at least two of the cardinals of all regenerating disc would have to die, all in a very short amount of time.

Overall, our system is quite robust to random cell death. Consider the following experiment. After there exist at least 50 cells, in each cell-step in which a new cell is created, a

cell is randomly chosen and with 50% probability it dies. For every $2k$ cells that are created, on average k have died, so about k cells are left. That is, the number of live and dead cells will be approximately the same. Several runs of this experiment have shown that, under these circumstances, the cells will still correctly approximate the compiled shape.

The system has been designed so that the shape can recover from two other types of failure. The first type is sudden massive cell death, that is, death to many cells at random locations across the shape at once. The second is regional cell death where a contiguous patch of cells die at once. We are still investigating how well the system can recover from these failures.

7 Future Work

As stated before, lack of noise in messaging is unrealistic; a natural next step would be to deal with this type of failure. Furthermore, to match the robustness of live tissue, we would have to get rid of the assumptions of uniform cell size, immobile cells, linear, reliable and non-changing gradient decay function (the latter for real chemical gradients is exponential with exponent multiplier varying with temperature!). The present approach could prove interesting for the areas of reconfigurable robots and autonomous agents [Butler *et al.*, 2002]. Finally, it would be worthwhile to translate the current 2 dimensional program into a 3D shape growing implementation.

8 Acknowledgements

This research was supported by a National Science Foundation grant on Quantum and Biologically Inspired Computing (QuBIC) from the Division of Experimental and Integrative Activities, contract EIA-0130391. The author would like to thank Jacob Abernethy, Catherine Chang, Hal Abelson, Radhika Nagpal and Gerald Sussman for their help.

References

- [Abelson *et al.*, 1995] H. Abelson, A. Berlin, N. Cohen, L. Fogel, C. Ho, M. Horowitz, J. How, T. Knight, R. Newton, and K. Pister. Distributed information systems for MEMS. *ISAT (Information Science and Technology Study Group) Study*, 1995.
- [Abelson *et al.*, 1999] H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, T. Knight, R. Nagpal, E. Rauch, G. Sussman, and R. Weiss. Amorphous Computing. *White paper*, 1999. <http://www.swiss.ai.mit.edu/projects/amorphous/>.
- [Abelson *et al.*, 2000] H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, T. Knight, R. Nagpal, E. Rauch, G. Sussman, and R. Weiss. Amorphous computing. *Communications of the ACM*, 43(5), May 2000.
- [Bard, 1990] J. Bard. *Morphogenesis*. Cambridge University Press, U.K., 1990.
- [Butera, 2001] W. Butera. *Programming a Paintable Computer*. PhD thesis, MIT Media Lab, 2001.
- [Butler *et al.*, 2002] Zack Butler, Satoshi Murata, and Daniela Rus. Distributed replication algorithms for self-reconfigurable modular robots. *Distributed Autonomous Robotics Systems*, 5, 2002.
- [Coore, 1999] D. Coore. *Botanical Computing: A Developmental Approach to Generating Interconnect Topologies on an Amorphous Computer*. PhD thesis, MIT, Dept of Electrical Eng. and Computer Science, February 1999.
- [Forrest and Mitchell, 1993] S. Forrest and M. Mitchell. What makes a problem hard for a genetic algorithm? *Machine Learning*, 13:285–319, 1993.
- [Lawrence, 1992] P. A. Lawrence. *The Making of a Fly: the Genetics of Animal Design*. Blackwell Science, Oxford, U.K., 1992.
- [Margolus, 1996] N. Margolus. Cam-8: A computer architecture based on cellular automata. *Pattern Formation and Lattice-Gas Automata, American Mathematical Society*, pages 167–187, 1996.
- [McLurkin, 1999] James McLurkin. Algorithms for distributed sensor networks. Master's thesis, University of California, Berkeley, December 1999.
- [Mitchell *et al.*, 1994a] M. Mitchell, J. Crutchfield, and P. Hrabar. Evolving cellular automata to perform computations: Mechanisms and impediments. *Physica D*, 75:361–391, 1994.
- [Mitchell *et al.*, 1994b] M. Mitchell, J. Holland, and S. Forrest. When will a genetic algorithm outperform hill climbing? *Advances in Neural Information Processing Systems*, pages 285–319, 1994.
- [Nagpal, 2001] R. Nagpal. *Programmable Self-Assembly: Constructing Global Shape using Biologically-inspired Local Interactions and Origami Mathematics*. PhD thesis, MIT, Dept of Electrical Engineering and Computer Science, June 2001.
- [Nusslein-Volhard, 1996] C. Nusslein-Volhard. Gradients that organize embryo development. *Scientific American*, August 1996.
- [Weiss *et al.*, 1998] R. Weiss, G. Homsy, and R. Nagpal. Programming biological cells. In *8th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '98), Wild and Crazy Ideas Session*, October 1998.
- [Weiss *et al.*, 1999] R. Weiss, G. Homsy, and T. Knight. Toward in vivo digital circuits. In *Dimacs Workshop on Evolution as Computation*, January 1999.
- [Wolpert, 1969] L. Wolpert. Positional information and the spatial pattern of cellular differentiation. *Journal of Theoretical Biology*, 25:1–47, 1969.
- [Wolpert, 1998] L. Wolpert. *Principles of Development*. Oxford University Press, U.K., 1998.

Emergence of Cooperation in a Pursuit-Evasion Game

Geoff Nitschke

Artificial Intelligence Laboratory
Department of Information Technology
University of Zurich, Winterthustr. 190
Zurich, Switzerland
nitschke@ifi.unizh.ch

Abstract

This research concerns the comparison of three different artificial evolution approaches to the design of cooperative behavior in a group of simulated mobile robots. The first and second approaches, termed: *single pool* and *plasticity*, are characterized by robots that share a single genotype, though the plasticity approach includes a learning mechanism. The third approach, termed: *multiple pools*, is characterized by robots that use different genotypes. The application domain implements a pursuit-evasion game in which teams of robots of various sizes, termed: *predators*, collectively work to capture either one or two others, termed: *prey*. These artificial evolution approaches are also compared with a static rule based cooperative pursuit strategy specified *a priori*. Results indicate that the multiple pools approach is superior comparative to the other approaches in terms of measures defined for prey-capture strategy performance. That is, this approach facilitated specialization of behavioral roles allowing it to be effective for all predator team sizes tested.

1 Introduction

The use of evolutionary computation as a design process for control of robot teams has received much attention in both real world [Quinn, 2000], [Schultz and Grefenstette, 2000], and simulated problem domains [Kitano, 1997]. Such attention is due to the difficulty in specifying *a priori* an effective means for multiple interacting robot controllers, and the inherent adaptability of artificially evolved controllers. The use of evolutionary computation to exploit emergent cooperation remains a relatively unexplored area of research in the pursuit and evasion domain [Benda *et al.* 1985] and related predator-prey systems [Nishimura and Takashi, 1997] using multiple predators and prey. Various approaches have been used to study the pursuit evasion domain, where the task is for multiple predators to capture a prey by surrounding it [Korf, 1992], [Levy and Rosenschein, 1992], though few researchers have investigated emergent cooperation in these systems, with notable exceptions such as Denzinger and Fuchs [1996], Haynes and Sen [1996] and Yong and Miikkulainen [2001].

This paper describes a comparison of three artificial evolution approaches for the synthesis of cooperative behaviour evaluated within teams of simulated *Khepera* robots [Mondada *et al.* 1993]. The three approaches are compared with a rule based pursuit strategy and each is evaluated in terms of predator fitness scored, and the time period for which a prey is immobilized. For each of the four approaches teams of various sizes are compared. Cooperative behavior is only evolved for the predators, and each prey is able to move 20 percent faster than the predators. The behavior of each prey is not evolved, but instead uses static obstacle avoidance behavior. Functionally, each predator is the same in terms of movement and sensor capabilities. The collective task was for the predators to immobilize a prey. The predator team was rewarded fitness proportional to how much it was able to slow down a prey, where maximum fitness was rewarded if a prey was immobilized. A control experiment using a single predator, demonstrated that at least two predators are needed to accomplish this task. Experimental results support a hypothesis that the multiple pools approach, which encourages behavioural specialization, would yield a superior performance in terms of the two measures defined to quantify evolved prey-capture strategy performance and this superior performance would prove consistent for all group sizes tested.

2 Artificial Evolution Approaches

Four sets of experiments were run in order to test each of the three artificial evolution approaches, and the rule-based approach. Ten replications of each experiment were made, where each ran for 500 generations. The predator team 'lived' for 10 *epochs*, where each epoch consisted of 1000 *cycles* of simulation time. Each epoch constituted a test scenario where all predators and prey were tested for different, randomly generated orientations and starting positions in the environment. Six different group configurations of predators and prey were tested for each of the four approaches. The group types were as follows. *Group type 1*: 3 predators and 1 prey; *Group type 2*: 3 predators and 3 prey; *Group type 3*: 5 predators and 1 prey; *Group type 4*: 5 predators and 3 prey; *Group type 5*: 8 predators and 1 prey; *Group type 6*: 8 predators and 3 prey.

Single Pool Approach: As illustrated in figure 1 (left) this approach generates and tests *n* copies of a single genotype, meaning that the predator team is homogenous.

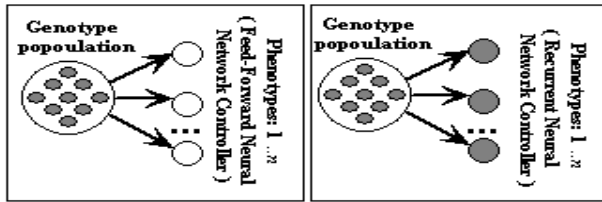


Figure 1. **Left:** Single Pool – Each predator phenotype corresponds to a genotype selected from a population and copied n times. **Right:** Plasticity – As with Single Pool, though phenotypes implement a recurrent neural network controller.

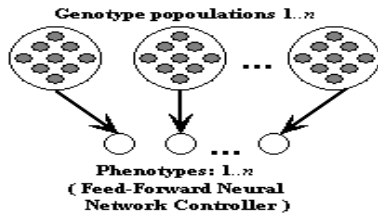


Figure 2. Multiple Pools Approach – Predator phenotypes correspond to n different genotypes selected from n separate pools of genotypes.

In this approach there is no plasticity so the predators cannot adapt during their lifetime. The fitness assigned to each predator is simply the fitness calculated for the single genotype that specifies the predator team. The main advantage of this approach is its simplicity in terms of behavioral encoding and calculation of team fitness.

Plasticity Approach: As illustrated in figure 1 (right) this approach generates and tests n copies of a single genotype, so that as with the *single pool* approach, the predator team is homogenous. The difference is that individual phenotypes are able to adapt during their lifetime as a result of a recurrent neural network learning process. The advantage of the plasticity approach is that it allows for specialization of behaviour by individual predators without being affected by the problem of needing to estimate fitness contribution of different predators to the team as a whole. For both the single pool and plasticity approaches, every individual genotype in the population is tested against $n-1$ randomly selected genotypes from the same population. This process is repeated for all 10 epochs of a predator's lifetime.

Multiple Pools Approach: As illustrated in figure 2 this approach takes a single genotype from every population of genotypes where the number of populations corresponds to the number of predators. Each genotype is then decoded into a separate phenotype, where this set of phenotypes then represents the team of predators. In each generation, every individual genotype in a population is tested against $n-1$ other genotypes, randomly selected from one of the other populations of genotypes. This process is then repeated for every epoch. The advantage of the multiple pools approach is that it encourages behavioural specialization in the group of predators, in that the artificial evolution setup provides for more genetic diversity.

Rule Based Approach: This approach takes a predator previously evolved for obstacle avoidance and then extends

this behavior with a simple set of rules designed to encourage a cooperative encirclement pursuit strategy when at least two predators collectively approach a single prey. The rules cause a predator to move towards the right side of detected prey, and to push to the left against that side, while moving about the prey's periphery. If an obstacle is detected on the prey's right side, the predator will attempt to keep moving about the prey's periphery until a 'free-space' is found.

Evaluation of Approaches

In order to quantify the effectiveness of emergent prey capture strategies, two different measures were used to evaluate performance. The first was predator team fitness, where fitness awarded to the team was proportional to how much a prey was slowed during the team's lifetime. The second was prey capture time, which was the time period for which a prey was immobilized by the collective efforts of at least two predators. For both the *single pool* and *plasticity* approaches a single genotype specifies the entire predator team. That is, predators are clones of each other, so evaluation of team performance in this case is not problematic. The performance of a predator team executed under either of these approaches is simply measured as the fitness value assigned to the genotype that specifies the team. For the rule-based approach, a pseudo fitness function that emulates the single pool and plasticity approaches was implemented. In contrast to these approaches a predator team using the multiple pools approach is specified by n genotypes selected from n different populations. Hence, each genotype must be assigned an individual fitness score, and team performance evaluation needs to be computed by estimating the fitness contribution of each genotype to the team as a whole. A method of evaluation widely known as: *fitness sharing* [Bull and Holland, 1997] was implemented for the multiple pools approach, where an equal fitness score is assigned to each individual genotype, thereby assuming that each individual contributed to team performance equally. The advantage of this method is that fitness for individual genotypes is easily calculated and there is no disparity between team fitness and the fitness of individual team members.

Agents, Environment and Artificial Evolution

For all experiments a generational evolutionary algorithm using linear rank-based selection was used [Goldberg, 1989]. Each population contained 100 genotypes, where initial populations consisted of randomly generated genotypes. Genotype length was set to 24 genes, where each gene consisted of several bits encoding each neuron type and connection weights. At the turn of each generation, the 20 genotypes that have accumulated the highest fitness were allowed to reproduce. The total fitness of an individual genotype was the sum of all its fitness for all epochs of its life. Reproduction was done via generating five copies of each genotype in order to create the next generation.

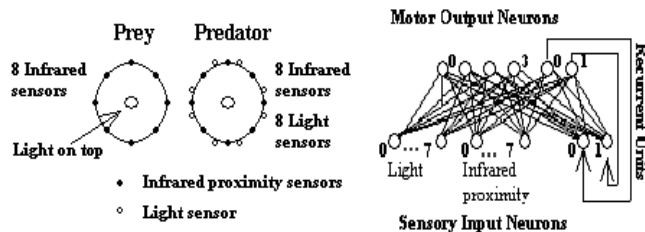


Figure 3. Left: Each predator and prey is a simulated *Khepera* robot. Predators have 8 infrared proximity sensors, and 8 light sensors. Prey have 8 infrared proximity sensors, and a light. Right: A feed forward neural network controller is used for the single pool and multiple pools approaches, and a recurrent neural network controller is used for the plasticity approach.

During this copying process 10 percent of the connection weights were mutated. Mutation added a random value between -1.0 and $+1.0$ to the weights current value. This process was repeated for the 500 generations that each simulation ran for. As illustrated in figure 3 (left), the body of each predator and prey is assumed to be a *Khepera* mobile robot [Mondada *et al.* 1993].

The robots used as predators were equipped with 8 infrared proximity sensors, and 8 light sensors positioned on the periphery of the *Khepera*. The robots used as prey were equipped with 8 infrared proximity sensors, as well as a light on its top. This light could be detected by the predator light sensors and was used so as each predator robot could distinguish fellow predators from a prey. Figure 3 (right) illustrates the feed-forward neural network consisting of an input and output layer with no hidden units [Nolfi and Parisi, 1997] that controlled all robots executed under the single pool and plasticity approaches. In the case of the predators, the input layer consisted of 16 units that encoded the activation level of the robots 16 sensors. These 16 input units were connected to 4 output units. Figure 3 (right) also illustrates the recurrent neural network controller used in the plasticity experiments, where the activation level of two additional output units was copied back into two additional input units. The first two output units represented the two motors of the robot and encoded the speed of the two wheels. These motor units controlled the robots behavior in the environment. The next two output units represented two teaching units that encoded a teaching input for the first two output units. The two motor units used this teaching input in order to learn using the back propagation procedure [Rumelhart *et al.* 1986]. In the plasticity experiments there were an additional two output units that were the recurrent units and contained activation values for the motors from the previous cycle. For the robots that were the prey, a network connecting 8 sensory input units to 4 motor output units was trained for an obstacle avoidance behavior before being placed in the environment.

The environment corresponded to a 1000cm x 1000cm arena with no obstacles. When a predator robot was placed in the environment, sensory input was received via the input units, and activation values were passed to the two motor units, and the teaching units. The activation value of the two

motor units was used to move the robot, thus changing the sensor input for the next simulation cycle. The activation value of the two teaching units was used to change the weights that connected the input units to the motor units using back propagation. This cycle was then repeated.

3 Results

Four sets of experiments were executed to test each of the three artificial evolution approaches and the rule-based approach. Each set of experiments tested the six group types and ten replications of each experiment were made.

Figure 4 illustrates the average fitness attained for all groups of predators using the rule-based, single pool, plasticity and multiple pools approaches. Figure 5 presents the average *prey capture time* attained for the four approaches. *Prey capture* refers to the instance when the prey is immobilized for a time interval $t_0..t_i$.

Evolved Behavior

Evolved behavior observed for each of the three artificial evolution approaches is described in this section. For the rule-based approach, even though behavior was not evolved as for the other approaches, a single cooperative pursuit strategy emerged as the result of interaction of multiple predators following a simple set of rules.

Single Pool: For all six group types tested only two cooperative prey-capture strategies consistently emerged. These strategies, termed: *encirclement* and *entrapment* are briefly described in the following. In the encirclement strategy at least three predators move to circle the prey, each moving in the same direction in close proximity to the prey, for some period of time. The strategy was rarely able to immobilize the prey and was only effective in slowing the prey for a short period of time, given that the predators were not able to coordinate their movements for an extended period. The entrapment strategy, illustrated in figure 6, also used at least three predators, where either one or two predators moved to each side of the prey, while another, termed: a *blocker*, moved around the flanking predators, to approach the prey from the front, in order to trap the prey in a triangular formation. In the example depicted in figure 6, P_1 and P_2 are the flanking predators and P_3 is the blocker. Note that at simulation time step $n+j$ when the prey moved to escape, the flanking predators moved also, turning one way to force the prey in a specific direction. The blocker then moved around in order to affront the prey again. This system of entrapment, movement, and then entrapment continued for several times before the prey was able to evade the predators.

Rule-Based: A single cooperative strategy emerged using at least two and at most three predators. The strategy was similar to the encirclement strategy that evolved under the single pool approach. As with the single pool version of encirclement, the strategy was rarely successful at immobilizing a prey due to the lack of coordination between predators as they collectively approached a prey.

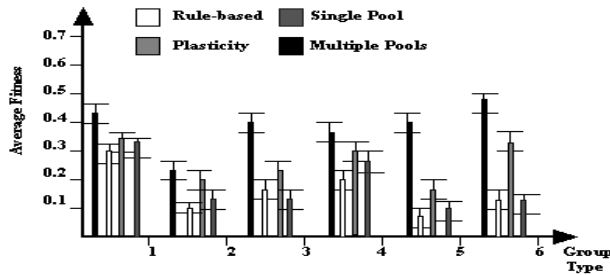


Figure 4. Average fitness presented, with standard error bars, for all group types tested under the three artificial evolution approaches, and the rule-based approach.

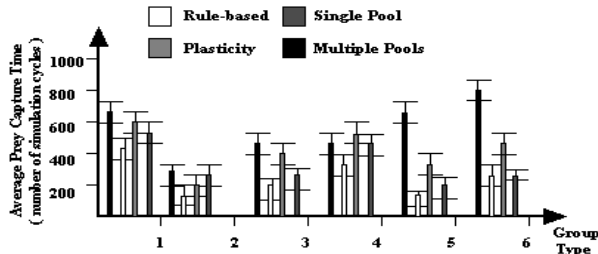


Figure 5. Average prey-capture time presented, with standard error bars, for all group types tested under the three artificial evolution approaches, and the rule-based approach.

Plasticity: In these experiments one emergent cooperative prey capture strategy was observed. This strategy, illustrated in figure 7, was a derivative of the entrapment strategy and termed: *role switcher*. In this strategy a form of behavioural specialization emerged during the lifetime of the predators. Specifically, two different types of behavioral roles emerged, termed *knocker* and *flanker*. A flanking predator moved along one side and just ahead of the prey forcing the prey to turn in a particular direction, whereas a knocking predator ‘knocked’ into the side of the prey forcing it to slow its movement. The role-switcher strategy was formed by a knocker predator moving along side a flanking predator and role of a flanker and knocker switching whenever a prey attempted evasion. Figure 7 depicts predator P_3 formerly flanking assumes the knocking role at simulation time step $n+k$ after the prey moves to evade, and predator P_1 formerly knocking assumes a flanking role. This dynamic adoption and switching of roles coordinated the movements of the predators and thus effectively slow a prey’s movement. Also, relative to the entrapment and encirclement strategies, role switcher was able to more frequently immobilize a prey.

Multiple Pools: As with the plasticity experiments the *role switcher* strategy was the only cooperative prey-capture strategy that consistently emerged, though a specific difference was noted. Namely, different predators adopted different roles from the beginning of their lifetimes. Specifically, particular predators always assumed the role of a *flanker*, while other predators always assumed the role of a *knocker*. Figure 8 depicts predators P_2 , P_3 and P_5 as maintaining the roles of flankers, and predators P_1 and P_4 maintaining the roles of knockers.

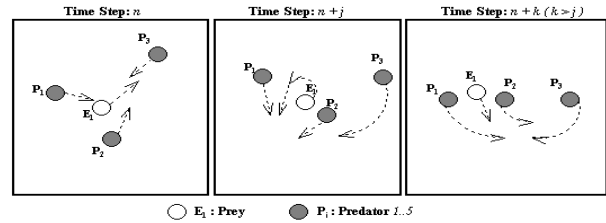


Figure 6. The cooperative entrapment pursuit strategy, illustrated for group type 1: 3 predators and 1 prey. The strategy was not successful at immobilizing an evader.

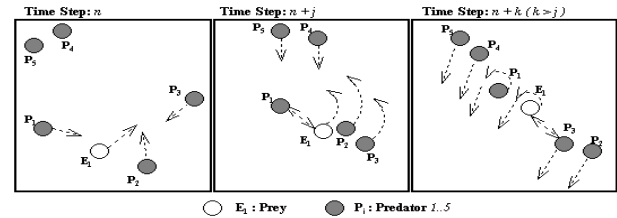


Figure 7. The role-switcher pursuit strategy, illustrated for group type 3: 5 predators and 1 prey. Note that a dynamic assumption of behavioral roles in the formation of the strategy is depicted here.

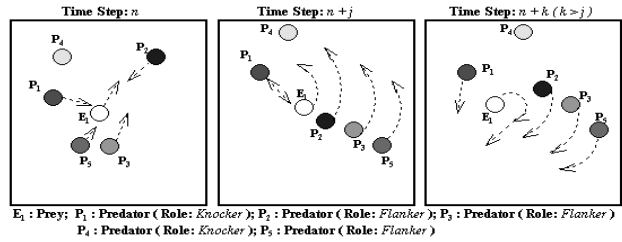


Figure 8. The role-switcher pursuit strategy, illustrated for group type 3: 5 predators and 1 prey. Here the assumption of behavioral roles is set at the beginning of a predator’s lifetime.

Thus, at simulation time step $n+j$ predator P_4 does not move to assume the role of a knocker in the strategy as it is already taken by predator P_1 at this stage. This adoption of roles that was maintained throughout the lifetime of the predators served to increase the effectiveness of the role switcher strategy in terms of the measures defined for prey capture strategy performance. Notably the strategy was consistently effective at immobilizing a prey, for all group types tested.

4 Analysis and Discussion

This discussion compares emergent prey-capture strategies with the rule-based pursuit strategy and relates each to the group types tested and the two performance measures.

Single Pool: Two cooperative prey-capture strategies, each using at least three predators, consistently emerged for all six group types tested. Though, relative to the plasticity and multiple pools approaches these emergent strategies performed poorly in terms of average *team fitness* and *prey-capture time*. These results are presented in figures 4 and 5 respectively and as shown were consistent for all group types tested. The low fitness and prey-capture time of the encirclement and entrapment strategies was found to be a result of interference that occurred as two or more predators

collectively approached a prey. This result was found to be due to confused infrared sensor readings of predators in close proximity to each other, and the fact that individual predators did not possess any memory, explicit form of communication, or coordination to facilitate a successful cooperative prey capture strategy. As illustrated in *figures 4* and *5*, this result is especially prevalent for *group types 5* and *6*, which used eight predators.

Rule-Based: From the interactions of predators following a set of rules, an encirclement strategy emerged, though as with the single pool version of the strategy it was effective for at most three predators. Also, note in *figures 4* and *5*, that similar to its single pool counterpart the strategy proved most effective for *group types: 1* and *4*. Given that in these experiments the chance of predators physically interfering with each other as they approached the prey was minimal. The rule-based approach, similar to the single pools approach, also performed best in group types where the predator to prey ratio was medium as in *group types 1* and *4*.

Plasticity: In experiments using this approach the role switcher strategy frequently emerged for all group types tested. In the role-switcher strategy a form of dynamic behavioral specialization emerged in at least three predators. This behavioral specialization was in the form of dynamic role adoption that emerged during the lifetime of a predator and varied from predator to predator depending upon the group type being tested. This dynamic role adoption facilitated cooperation between the predators affording the predator team a high average *fitness* and *prey-capture time* comparative to the single pool and rule-based approaches, for all group types tested. This result is presented in *figures 4* and *5* respectively. In experiments testing *group types 1, 3* and *5* at least three and at most six predators constituted the role-switcher strategy, where a maximum of three flanked either side, and the role of a flanking and knocking predator switched whenever the prey turned to evade. In experiments testing *group types 1* and *6*, at least three and at most eight predators formed the role-switcher strategy, where a maximum of two predators flanked either side of a prey. As with the experiments testing *group types 1, 3*, and *5* the role of the predator closest to the prey and the knocking predator switched whenever the prey turned so as to escape, though the difference of experimental results from testing *group types 1* and *6*, was that two sub-groups of predators emerged, meaning that two of the prey were simultaneously engaged by two predator sub-teams.

The dynamic assumption of roles during a predators lifetime allowed for the formation of sub-groups in the predator team, and thus served to yield a higher average team fitness and prey capture time. As presented in *figures 4* and *5* respectively, these two measures were relatively higher in experiments testing *group types 1, 4*, and *6*, and low in experiments testing *group types 3* and *5*. In experiments testing *group types 3* and *5*, this result was due to multiple predators attempting to assume the same behavioral role as they collectively approached the single prey. For *group type 2*, the 1 predator to 1 prey ratio

increased the chance of the strategy dispersing given an increased probability of distraction from a passing prey. In the other group types tested a higher predator to prey ratio reduced the number of predators that collectively approached the prey at any given time, thus increasing the likelihood of unhindered dynamic adoption of a behavioral role in the role-switcher strategy. It is theorized that behavioral specialization for role switching emerged as an indirect result of interference that occurred when at least three predators collectively approached a prey. Such interference was observed in the single pool experiments and often caused emergent strategies to fail prematurely, thereby making it more difficult for such strategies using at least three predators to be selected for and propagated by the evolutionary process. Where as, the *role-switcher* strategy was able to achieve a high team fitness and prey-capture time, it was at best only able to slow the prey, and never completely immobilizing the prey in experiments testing *group types 2* through to *5*. This is reflected in *figure 5*, which presents a relatively higher average prey capture time for experiments testing *group types 1* and *6*. Thus, the dynamic adoption of roles was only effective where the predator to prey ratio was sufficiently high, though not so high as in the case of *group type 5* where an excess of physical interference between predators prevented the role-switcher strategy from regularly forming.

Multiple Pools: In experiments using the multiple pools approach only a single cooperative prey-capture strategy emerged. This strategy was classified as a derivative of the role-switcher strategy using at least three predators that was observed in the plasticity experiments. Though the multiple-pools role-switcher used a genetic based specialization. In the plasticity experiments the adoption of specialized behaviour was dependent upon the positions of the predators at a given time, where as in the multiple pools experiments, different predators initially behaved differently and assumed genetically pre-determined roles prior to engaging in the multiple pools version of the *role-switcher* prey-capture strategy. That is, one predator always assumed the role of the knocker while the others always assumed the role of a flanker, or that of an idle predator. Idle predators, for example predator P_4 in *figure 8*, served the purpose of reducing any potential interference between predators as they collectively approach a prey. Idle predators also increased the chance of success of the prey-capture strategy via limiting the number of predators that constituted the strategy. The fact that the predators were genetically different is one explanation for the evolution of specialized behavioural roles that complemented each other in the emergence of the multiple pools role switcher strategy. In the experiments testing *group type 6*, two specialized sub-groups of predators, comprising two groups of four predators emerged.

The effectiveness of these two specialized sub-groups is illustrated in *figures 4* and *5*, which present a relatively high average fitness and prey-capture time for all group types, comparative to the single pool, rule-based and plasticity

approaches. Note that as with the plasticity approach, the multiple pools approach is more effective at exploiting an environment containing at least two prey via the formation of two specialized predator sub-groups. Thus, the multiple pools approach facilitated the evolution of specialized predator sub-groups to simultaneously capture two prey, where genetic based specialization proved to be more effective, in terms of measures for *fitness scored* and *prey capture time*. This comparison was made with the dynamic assumption of behavioral roles as done by predators using the plasticity approach.

5 Conclusions

This paper presented a set of experiments testing three different artificial evolution approaches and a rule-based approach for the synthesis of cooperative behaviour within a team of predators, where the task was to cooperatively capture at least two prey. Results compared the performance of these approaches for six different group configurations. The comparison was in terms of two measures designed to quantify the efficacy of emergent cooperative prey capture strategies. These measures were *predator fitness* and *prey-capture time*. Results presented indicated the multiple pools approach to be superior in terms of these two measures for all group types tested. The superiority of the multiple pools approach was found to be a result of a genetic form of behavioral specialization that assigned behavioral roles at the beginning of a predators lifetime. The multiple pools approach also facilitated the evolution of two specialized sub-groups of predators in scenarios using at least two prey. Evolution of these sub-groups improved the effectiveness of these strategies in terms of higher *team fitness* and *prey capture time*, comparative to the rule-based, single pool and plasticity approaches. Specifically, the specialized sub-groups aided in reducing interference between predators as they collectively approached a prey.

A comparison with other evolution of cooperation approaches in the pursuit-evasion domain [Denzinger and Fuchs, 1996], [Haynes and Sen, 1996] and [Yong and Miikkulainen, 2001] is difficult given the real world nature of the experiments described in this paper. Though the robots were simulated, the environment was a continuous domain and the simulation incorporated noise in sensory data, namely confused infrared sensor readings resulting from two or more Khepera's being in close proximity to each other. This noisy sensor data was a key reason for interference between multiple predators as they collectively approached a prey. Also, a continuous environment does not allow for the selection of distinct sets of situation/action values that are possible in grid world implementations [Denzinger and Fuchs, 1996] where a finite set of actions and resultant outcomes can be defined. While, the emergence of cooperation is simpler to analyze in these grid world domains, they are limited by their own implementations, so the study of mechanisms that facilitate emergent cooperation such as behavioral specialization is limited to trivial situations.

References

- [Benda *et al.* 1985] Benda, M., Jagannathan, V., and Dodhiawalla, R. An Optimal Cooperation of Knowledge Sources. Technical BCS-G2010-28, Boeing AI Center, August 1985.
- [Bull and Holland, 1997] Bull, L., and Holland, O. Evolutionary Computing in Multi-Agent Environments: Eusociality. In *Proceedings of the Second Annual Conference on Genetic Programming*, pages 347--352. San Francisco, USA, July, 1997.
- [Denzinger and Fuchs, 1996] Denzinger, J., and Fuchs, M. Experiments in Learning Prototypical Situations for Variants of the Pursuit Game. In *Proceedings of the Second ICMAS conference*, pages 48--55, Kyoto, Japan, August, 1996.
- [Goldberg, 1989] Goldberg, D. Genetic Algorithms. Addison Wesley, New York, USA. 1989.
- [Haynes and Sen, 1996] Haynes, T., and Sen, S. Evolving behavioral strategies in predators and prey. *Adaptation and Learning in Multi-Agent Systems*, Springer-Verlag, Berlin, 1996.
- [Kitano *et al.* 1999] Kitano, H., Asada, M., Noda, I., and Veloso, M. (1999). RoboCup: Today and Tomorrow - What we have learned. *Artificial Intelligence*, vol. 110(2): 193-214. Elsevier Science Publishers, Amsterdam, Holland.
- [Korf, 1992] Korf, R. E. A simple solution to pursuit games. *Working Papers of the Eleventh International Workshop on DAL*, pages 195--213. Geneva, Switzerland, February, 1992.
- [Levy and Rosenschein, 1992] Levy, R., and Rosenschein, J. S. A Game Theoretic Approach to Distributed Artificial Intelligence and the Pursuit Problem. In *Decentralized AI III*, pages 129--146, Kaiserslautern, Germany, August, 1992. Springer-Verlag.
- [Mondada *et al.* 1993] Mondada, F., Franzi, E., and Ienne, P. Mobile Robot Miniaturization: A tool for Investigation in Control Algorithms. In *Proceedings of Third International Symposium on Experimental Robotics*, pages 501--513, Kyoto, Japan, October, 1993. Springer-Verlag.
- [Nishimura and Ikegami, 1997] Nishimura, S. I., and Ikegami, T. Emergence of Collective Strategies in a Prey-Predator Game Model. *Artificial Life*, 3(1): 243--260. July, 1997.
- [Nolfi and Parisi, 1997] Nolfi S. and Parisi D. Learning to adapt to changing environments in evolving neural networks. *Adaptive Behavior*, 5(1): 75--98. July, 1997.
- [Quinn, 2000] Quinn, M. Evolving cooperative homogeneous multi-robot teams. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 1798--1803, Takamatsu, Japan, October 2000, IEEE Press.
- [Rumelhart *et al.* 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning internal representations by error propagation. *Parallel Distributed Processing, Volume 1: Foundations*. MIT Press, Cambridge, 1986.
- [Schultz and Grefenstette, 2000] Schultz, A. C. and Grefenstette, J. J. Continuous and Embedded Learning in Autonomous Vehicles: Adapting to sensor failures. In *Unmanned Ground Vehicle Technology II*, Proceedings of SPIE, vol. 4024(1): 55-62, American Association for Artificial Intelligence, 2000.
- [Yong and Miikkulainen, 2001] Yong, C. H., and Miikkulainen, R. Cooperative Co-evolution of Multi-Agent Systems. Technical Report AI01-287. Department of Computer Science, University of Texas, July 2001.

MULTIAGENT SYSTEMS

EVOLUTION AND GENETIC ALGORITHMS

When Evolving Populations is Better than Coevolving Individuals: The Blind Mice Problem

Thomas Miconi

34 Rue Desbordes-Valmore

75016 Paris, France

thomas.miconi@free.fr

Abstract

This paper is about the evolutionary design of multi-agent systems. An important part of recent research in this domain has been focusing on collaborative coevolutionary methods. We expose possible drawbacks of these methods, and show that for a non-trivial problem called the “blind mice” problem, a classical GA approach in which whole populations are evaluated, selected and crossed together (with a few tweaks) finds an elegant and non-intuitive solution more efficiently than cooperative coevolution. The difference in efficiency grows with the number of agents within the simulation. We propose an explanation for this poorer performance of cooperative coevolution, based on the intrinsic fragility of the evaluation process. This explanation is supported by theoretical and experimental arguments.

1 Introduction

Evolutionary algorithms are methods that apply the principles of darwinian evolution to the generation and adaptation of artificial, logical entities (function parameters, rulesets, programs...). Their usability as a search technique has been supported both analytically (e.g. the Schema Theorem [Goldberg, 1989] for genetic algorithms), and empirically by uncountable applications. However, the overwhelming majority of these applications are about the generation of individuals.

Collective evolution, that is, the generation and/or adaptation of collaborating populations of agents, has attracted comparatively less attention. There has been significant research in this domain though, especially over the last decade. This research led to algorithms of ever-growing complexity. This paper will first describe some of the work in that field, and more particularly the principle of cooperative coevolution, which seems to be the most popular type of method today.

We then expose what we feel are possible drawbacks of cooperative evolution, and propose a simpler way to adapt the canonical genetic algorithm to the generation of populations. We describe an experiment, based on the “blind mice” problem, and show that while an adapted genetic algorithm works pretty well with that problem, cooperative coevolution has more difficulties. Finally, we give an explanation for these

difficulties and for the difference in behaviour with the simpler genetic algorithm.

2 (Some of the) Related work

The simplest way to evolve a team of collaborating agents is to have all agents be identical, that is, to have homogeneous populations. These methods are not really different from individual evolution, except at evaluation time: to evaluate a given genotype, N agents are created out of this genotype instead of just one, and the resulting population is evaluated. Then start again with a different genotype, etc. While being very rigidly constrained, this method makes perfect sense in situations where one does not need heterogeneity at all. This method was used by [N.Zaera *et al.*, 1996] to evolve small groups of fish-like animats, controlled by neural networks, to perform extremely simple tasks (dispersion, aggregation, etc.).

A similar method was used by [Luke, 1998] to evolve competitive teams of soccer players for the Robocup competition [Kitano *et al.*, 1995]. The author used an adapted version of Genetic Programming [Koza, 1992]. There was also an attempt at introducing a limited degree of heterogeneity by decomposing teams into small sub-teams (defenders, attackers, etc.) and evolving different program trees for each such sub-teams. However, because of the enormous search space (and of the delays imposed by the Robocup server software), this approach proved intractable in practice: GP runs took days to produce meaningful results. Lack of time thus prevented the semi-heterogeneous teams from outperforming homogeneous teams.

A way to obtain some degree of heterogeneity is to have only one population and make it change gradually over time, replacing some agents by others based on some evaluation method. These new agents can be obtained by crossover or by duplication with mutation. If there is a way to evaluate the impact of one given individual, it is perfectly possible to perform a simple genetic algorithm over the population. This is, in essence, the idea behind classifier systems ([Holland and Reitman, 1978]), where a set of rules cooperate to control an animat, and where individuals are evaluated after the animat’s performance through a credit-sharing system.

In the same vein, we proposed a simple scheme in ([Miconi, 2001]), in which all agents were given an arbitrary index, and agents of index K could only mate with agents whose

indices fell within the $[K-r; K+r]$ range. Evaluation occurred simply by replacing one of the two parents by the offspring, then the second parent, and keeping the best of these two populations (with the possibility of discarding any changes if it decreased the performance of the system). This simple algorithm led to the emergence of sub-species that appeared, grew and shrank according to the needs of the population. An interesting feature of this algorithm was its incrementality, which allowed for long-term, adaptive evolution of the system.

Trying to obtain fully heterogeneous systems brings us to another level of complexity, right into the realm of cooperative coevolution. While coevolution has been most frequently applied in a *competitive* way (by confronting individuals to each other and using the result of this confrontation as an evaluation for individuals), it can also be used in a *cooperative* way, in order to evolve sets of collaborating agents.

In cooperative coevolutionary [Potter and DeJong, 1994] methods, each agent within the system is actually taken from a hidden subpopulation, or *pool*. To evaluate a given individual, it is associated with a set of *collaborators* (one from each other pool) and the resulting population is evaluated as a whole. The resulting score is then attributed to the currently evaluated individual. Based on this evaluation method, the classical GA cycle (evaluate, select and reproduce) is applied to each pool in turn, as many times as needed. In the first version of the cooperative coevolutionary algorithm (CCGA-1) collaborators are chosen by taking the best individual from every pool. However, in the CCGA-2 version, evaluation is refined by re-evaluating every agent with random collaborators, then taking the better score obtained between these two evaluations. The number of collaborators, the way these collaborators are chosen, the way the overall score is computed (averaging the different scores, or taking the best score, or taking the worst score, etc.) are important parameters that can influence the performance of the algorithms. The influence of these parameters has been studied to some extent by [Wiegand *et al.*, 2001], but this study applied only to simple function optimization problems with only two variables.

As happens frequently with good ideas, cooperative coevolution has been (re-)discovered a number of times under different names. Enforced subpopulations (ESP), for example, are exactly like cooperative coevolution, in which each agent is evaluated with only one set of collaborators: the best agents from all other pools. In other words, ESP is the CCGA-1 algorithm. While this method was initially devised for the evolution of neural networks [Gomez and Mikkulainen, 1997], it was successfully applied to multi-agent evolution by [Yong and Mikkulainen, 2001], who used it in a predator-prey simulation. The algorithm managed to find efficient strategies for predators, such as having two predators “chase” the prey while another one blocked it.

For some reason, the idea of simply using the standard genetic algorithm to whole populations seems to have fallen slightly out of fashion. The most probable reason is that it is simply too obvious to be talked about. The second reason is that it does have intrinsic drawbacks, such as a more massive search space. The third one is that it requires a few modifications to be adapted to the evolution of populations. All these aspects are discussed in section 3.2.

3 Two methods for evolving heterogeneous populations

3.1 Cooperative coevolution

Cooperative coevolution is quite an elaborate mechanism. Intuition indicates (and evidence confirms) that by focusing on each and every agent, it requires a huge number of evaluations to converge towards a solution. This algorithm concentrates on optimizing each individual agent in regard with the rest of the population; it is rather different from the more holistic approach of traditional GAs, in which full genomes are manipulated, and the (co-)adaptation of genes emerges naturally from selection, crossover and mutation - at least, in theory.

Why would it not be possible to simply use traditional GAs for the generation of multi-agent systems, regarding whole systems (not just each agent within them) as individuals? A simple answer is that this approach leads to very large genotypes, since the genotype for an “individual” has to code for several agents instead of just one, and the resulting search space might become intractable for GAs. Cooperative coevolution can thus be seen as a simple way to decompose a big problem into several smaller ones, even though these smaller problems are still strongly interlaced with each other.

However cooperative coevolutionary algorithms seem to have an important drawback: they basically evaluate each agent by assessing its impact on the performance of the whole system. The problem is that when the number of agents within the system grows, the influence of one single agent over the system’s performance tends to decrease, thus possibly making its assessment more difficult. This may become troublesome when the problem has a stochastic component, as is the case in many simulations. In this case, evaluating the same population several times can lead to different results. The consequence of this may be more important than one might think, as we will see below.

But first, it might be interesting to see how classical genetic algorithms can be adapted to the evolution of populations, and whether these population-oriented genetic algorithms can compete with cooperative coevolution.

3.2 Population-oriented genetic algorithms

Genetic algorithms work by evaluating individuals, selecting some of them according to their performance, crossing them together and mutating them, and starting over again. It is possible to apply exactly the same method to whole multi-agent systems. We can evaluate populations, cross them together (thus creating new populations that inherit agents from both parents), mutate them by changing one of their agents, etc.

However, the fact is that multi-agent systems are not simple individuals. They do have an obvious level of decomposition (the agent), and this can be exploited in several ways.

The most obvious idea is that in order to cover the search space efficiently, one must not only make new populations out of existing agents, one must also create new agents. To do this, we may introduce an *inner crossover* operator that allows us to cross two agents together. Thus, when creating a new population by importing agents from both parents, some of these imported agents would actually be the result of an inner crossover between agents from the parents.



Figure 1: Normal crossover (top) can be “spiced up” with inner crossover between individual agents (bottom). This allows for the creation of new agents, which is necessary to cover the search space efficiently

It is possible to make an analogy with traditional GAs: from the viewpoint of the whole population, these “crossed” agents have some similarity with bit-wise mutations in the standard genetic algorithm. They are part of the children’s genotype, yet they were not present in any of the parents’ genotypes. However, these are not exactly random mutations, since the genetic material still comes from the parents’ genotypes. This suggests that at first sight, “inner crossover” rate should be slightly higher than the usual mutation rate in a classical GA (usually about 2%-5% for each bit).

Another possibility is to enhance traditional crossover by occasionally swapping agents between populations in the final offspring. This, too, could allow for a better covering of the search space. However, we will not explore that possibility in the present paper.

4 Application: The Blind Mice problem

4.1 Description of the experiment

The experiment presented here is based on the “blind mice” problem. A flock of mice, controlled by simple feed-forward neural networks, have to escape a number of cats running after them in a toroidal world.

Now the “game” has three very simple rules:

1. The cats can see the mice and always run after the closest mouse around.
2. The mice run faster than the cats.
3. The mice can *not* see the cats. Neither can they see each other (they are “blind”). Their only input is a pair of numbers: the X and Y coordinates of the center of the flock.

When a cat touches a mouse, the cat is teleported to another, random location, and the population’s “catch counter” is increased; nothing else is changed, and the simulation is not interrupted in any way.

Let us consider these rules: they seem to make the problem extremely difficult for the mice. How is it possible to escape predators that can see you, but that you can’t see? Running around as fast as possible will just make them bump into any cat coming from the opposite direction. The same is true for random movement strategies. Given the enormous asymmetry of information between mice and cats, the survival chances of the poor rodents appear to be desperately thin. Even for a human designer, finding a solution to this

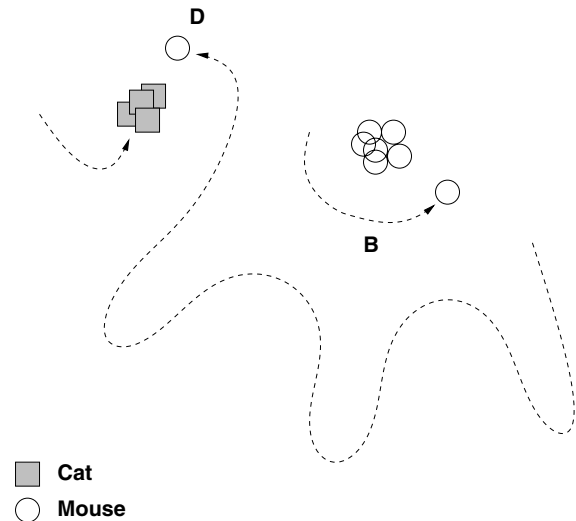


Figure 2: The successful strategy. Mouse D attracts the cats, mouse B plays a “balancing” role, and other mice move together in a tight flock

problem is not a trivial task. Yet, as we will see below, evolution managed to come up with an elaborate solution to this problem.

In our experiments, the mice are controlled by simple feed-forward neural networks with 2 inputs, 2 outputs and 5 hidden neurons. The two inputs are the coordinates of the center of gravity of the flock, that is, the sum of the X- (resp. Y-) coordinates of all mice, divided by the number of mice. The two outputs are two real numbers in the $[-3.0; +3.0]$ range, indicating the horizontal and vertical speed of the mouse. All weights are real numbers in the $[-1.0; +1.0]$ range. All simulations use 4 cats.

4.2 The evolved strategy

There seems to be an optimal strategy for this problem. This strategy emerged in all successful runs, sometimes with variants. This strategy is described in Figure 2. No other strategy led to a really efficient behaviour.

Let us explain this strategy: as we can see, most mice are aggregated together and move in a tight flock. This minimizes the probability that a “stray” cat might touch them, but it is not sufficient in itself to ensure a minimal catch rate. The really important behaviour is that of the mouse labelled D (the “Dancer”).

This mouse has a strange behaviour: it seems to revolve around the rest of the flock, but not in a strictly circular fashion. Instead, it constantly bounces around the flock, always staying at a respectable distance from it, and moving very fast along its path. The purpose of this behaviour becomes obvious when one sees the position of the cats: they are all following this “dancing” mouse, because it is simply the closest to them, thus leaving the rest of the flock alone.

In other words, the purpose of this dancing behaviour is simply to attract all the cats. The dancer moves very fast (so that it cannot be touched by cats), but along a sinuous path, so that:

- It can “drag” cats more efficiently in the initial stage, when cats and mice are at random position
- It never gets too far from the cats, which allows the cats to follow it endlessly even though it runs much faster than them.

The final touch of this strategy can be seen in the behaviour of the mouse labelled B (as “Balance”), even though it didn’t appear in all successful runs. This mouse also revolves around the flock, but much closer to it. In some runs it moves around the flock in a circular fashion, in other runs it bounces around it, but it usually stays on the opposite side of the flock with respect to the dancer. We believe that this mouse has a “balancing” role, in that it counterbalances the effect of the Dancer on the position of the center of gravity of the population, thus allowing the flock to be more stable.

Many variants appeared, such as having several dancers, or no balancer. But the essential traits of the strategy were consistent: aggregation of most mice, except for one or a few to attract the cats away from the flock.

Note that this strategy is very interdependent: each agent’s performance is highly dependant on other agents’ behaviour. This is even more true when you consider that in order to behave that way, they must calculate their trajectories out of only one input: the position of the center of gravity of the whole flock, which is based on the position of all other agents. This fact plays a significant role in the results described below.

5 Experimental results

5.1 Experimental settings

We used two algorithms for this problem: a simple genetic algorithm, adapted with an inner crossover operator (as described above), and a full-featured cooperative coevolutionary algorithm. Both methods were used with 7, then with 15 mice. We used 100 populations of 7 (resp. 15) mice for the first algorithm, and 7 (resp. 15) pools of 100 mice for the second one. Each algorithm was run several times with different random seeds.

In the first algorithm, reproduction of populations occurred through tournament selection and 1-point crossover at a rate of 60%. Every time two populations were thus crossed together, an inner crossover rate of 10% was applied, meaning that each mouse in the offspring had a 10% chance to be the result of the crossing of the parents’ corresponding mice. In the second algorithm, reproduction within each pool occurred by tournament selection and 1-point crossover at a rate of 30%, which proved to be the most efficient. In both algorithms, mutation appeared only when crossing two mice together, by choosing a new random value for a connection weight with a 5% probability.

Note that in both method, we use a limited form of elitism, in that the best individual from a given generation was preserved in the next generation. This ensured better performance - and made the obtained results even more puzzling, as explained below.

Finally, cooperative coevolution specifies that each agent must be evaluated with a set of collaborators. Several sets of

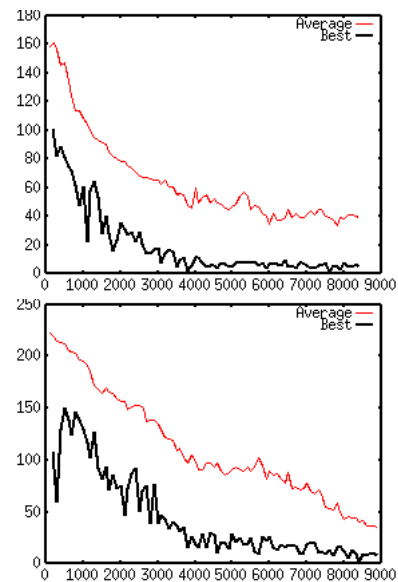


Figure 3: Performance of a population-adapted genetic algorithm, with 7 mice (top) and 15 mice (bottom). The y-axis indicates the number of mice caught during this evaluation round, while the x-axis indicates the number of evaluations. Both the fitness of the best population and the average fitness of all populations are shown.

collaborators can be used in turn in order to refine the evaluation, and the final result can be calculated from these successive evaluations in various ways (average, best, random...). However, we found that with this problem, increasing the number of collaborators (and thus the number of evaluation rounds) brought absolutely nothing, and was even damaging if the final score was anything else than the best score found. The most successful method was simply to evaluate each mouse by joining it with the best individual from each pool, exactly as in the enforced subpopulations algorithm. This is not a surprising result, however, as we will explain it below.

5.2 Comparison of results

The first algorithm (simple genetic algorithm with two levels of crossover) proved remarkably efficient with this problem. All runs led to the strategy described in section 4.2, whatever the number of mice within the simulation, although of course it took more time with 15 mice than with 7. Two typical runs, with respectively 7 and 15 mice, are described in Figure 3.

Cooperative coevolution led to different results. With 7 mice, in some runs, the algorithm failed to evolve any competitive behaviour. In other runs it managed to find the good strategy, but took more evaluations than with the previous algorithm. Many runs, however, achieved performance comparable with that of the simple genetic algorithm. With 15 mice, the success rate was much lower. Most runs did not converge after 10000 evaluations. Others converged, then suddenly diverged quickly. All the runs, with 7 or 15 mice, exhibited a intriguing pattern of oscillation.

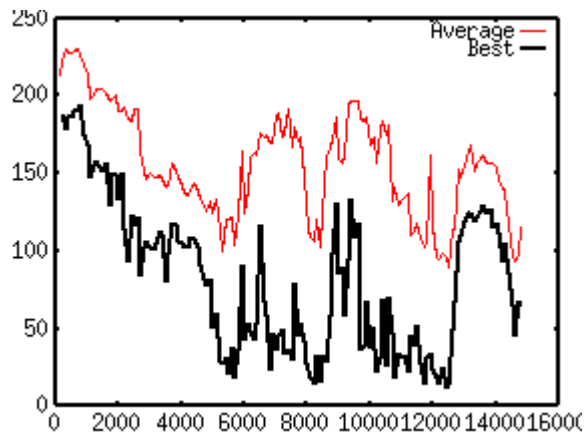


Figure 4: Performance of the best run for a cooperative co-evolutionary algorithm. The curves indicate the performance of the best individual in the currently evaluated pool, and the average fitness of individuals in this pool. Notice the brutal variations in the curves.

The fitness curve described in Figure 4 shows a good example of this pattern. It is the result of the most successful run with 15 mice. The population seems to converge towards a better behaviour, but then it suddenly diverges and seems to loose all that had been found. The pattern starts again a few cycles later.

This is not what one could expect. In this algorithm, the population remains quite stable, being composed of the best individual from each pool. Only one individual can be changed at each cycle, and only for a better one, since the best individual from previous generation is preserved. So how do we explain these brutal changes in fitness happening every now and then?

Our explanation is that this pattern is the result of cooperative coevolution's main drawback: when the number of agents grow, and when the result of evaluation undergoes important stochastic variations, its performance is bound to decrease.

What happens is that at some point, the algorithm does find a "good" population, and keeps optimizing it by replacing each agent by a better one in turn - as it should do. But then at some point, because of the stochastic variations in the evaluation process, one of these "good" agents is found to perform not as well as another, non-optimal one. In classical evolutionary algorithms, these occasional mistakes are blurred over a large number of trials and errors, and besides, one population's performance has no impact on another's. But in cooperative coevolution, this exceptional case is sufficient to wreck the behaviour of the whole population, because it can replace an essential agent (say, a Dancer) by a poorly performing agent (say, a random wanderer).

Then other agents from other pools are evaluated in turn, but since the conditions have changed dramatically, other agents are selected instead of the previous, quasi-optimal ones. Thus, an error in the selection of one agent has an influence over all other pools. When the evaluation cycle comes back to the first pool, the previously essential agent

may not be selected again, because the rest of the population has co-adapted towards a different state. However, it is possible that the this agent is selected again, thus resulting in a sharp increase in performance - until the next "error" in the evolutionary process. This seems to cause the up-and-down oscillations in the fitness curve.

There is little we can do about this problem. As we said, increasing the number of collaborators in a classical way (that is, by selecting random sets of collaborators) simply does not work, which is understandable: in this problem, the solution requires highly interdependent behaviours. An agent's performance can only be good if other agents play their role. Evaluating each agent by fitting it in a random population is not likely to help reach this state. In a problem with many strongly interdependent agents, adding random collaborators doesn't seem to be a helpful solution.¹

We could devise more elaborate schemes (e.g. using the second-best from each pool as a second set of evaluators) but that would not suppress the problem. It seems that in problems with a non-deterministic component, cooperative coevolution is essentially fragile, or at least, more so than the standard genetic algorithm.

6 Conclusion

We have described possible pitfalls in cooperative coevolution, and we have proposed a way to apply the canonical genetic algorithm to populations of agents. We have applied both algorithms to a non-trivial problem, the blind mice problem, that is both conceptually simple and non-trivial to solve. For this problem, the populations-oriented genetic algorithm significantly outperformed cooperative coevolution. We proposed an explanation for the difference in results, based on the intrinsic fragility of cooperative coevolution in regard to excessive stochastic variations in the evaluation process.

As a final note, an old French proverb goes: "Quand le chat n'est pas là, les souris dansent" (When the cat is away, the mice dance). Artificial evolution has demonstrated that the opposite can be true as well.

References

- [Goldberg, 1989] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.
- [Gomez and Mikkulainen, 1997] F. Gomez and R. Mikkulainen. Incremental evolution of complex general behavior. *Adaptive Behavior*, 5:317–342, 1997.
- [Holland and Reitman, 1978] J. H. Holland and J. S. Reitman. Cognitive systems based on adaptive algorithms. In

¹Strangely enough, in [Potter and DeJong, 1994], random collaborators were introduced precisely for problems with strongly interacting variables, such as the Rosenbrock function; however, for this kind of problem, this idea was only applied to two-variables functions. In this case, choosing a random collaborator in addition to the better one may allow the algorithm to escape a local minima. But as the number of agents grows, it seems that evaluating one agent with a set of totally random collaborators could hardly bring any valuable information.

- D. A. Waterman and F. Hayes-Roth, editors, *Pattern Directed Inference Systems*, pages 313–329. Academic Press, 1978.
- [Kitano *et al.*, 1995] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. Robocup: The robot world cup initiative. In *Proceedings of IJCAI-95 Workshop on Entertainment and AI/ALife*, 1995.
- [Koza, 1992] John Koza. *Genetic Programming*. MIT Press, 1992.
- [Luke, 1998] Sean Luke. Genetic programming produced competitive soccer softbot teams for robocup97. In John Koza, editor, *Proceedings of the Third Annual Genetic Programming Conference (GP98)*, pages 204–222, 1998.
- [Miconi, 2001] Thomas Miconi. A collective genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 876–883. Morgan Kaufmann, 2001.
- [N.Zaera *et al.*, 1996] N.Zaera, D.Cliff, and J.Bruten. (not) evolving collective behaviours in synthetic fish. In *Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior (SAB96)*, pages 635–644. MIT Press, 1996.
- [Potter and DeJong, 1994] M. A. Potter and K. A. DeJong. A cooperative coevolutionary approach to function optimization. In *Proceedings of The Third Parallel Problem Solving from Nature*, pages 249–257. Springer-Verlag, 1994.
- [Wiegand *et al.*, 2001] R. Paul Wiegand, William C. Liles, and Kenneth A. DeJong. An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 1235–1242. Morgan Kaufmann, 2001.
- [Yong and Mikkulainen, 2001] C.H. Yong and R. Mikkulainen. Cooperative coevolution of multi-agent systems. Technical Report AI01-287, University of Texas at Austin, 2001.

Improving Coevolutionary Search for Optimal Multiagent Behaviors

Liviu Panait

Department of Computer Science
George Mason University
Fairfax, VA 22030
lpanait@cs.gmu.edu

R. Paul Wiegand

Krasnow Institute
George Mason University
Fairfax, VA 22030
paul@tesseract.org

Sean Luke

Department of Computer Science
George Mason University
Fairfax, VA 22030
sean@cs.gmu.edu

Abstract

Evolutionary computation is a useful technique for learning behaviors in multiagent systems. Among the several types of evolutionary computation, one natural and popular method is to coevolve multiagent behaviors in multiple, cooperating populations. Recent research has suggested that coevolutionary systems may favor stability rather than performance in some domains. In order to improve upon existing methods, this paper examines the idea of modifying traditional coevolution, biasing it to search for maximal rewards. We introduce a theoretical justification of the improved method and present experiments in three problem domains. We conclude that biasing can help coevolution find better results in some multiagent problem domains.

1 Introduction

Multi-agent learning is an area of intense research, and is challenging because the problem dynamics are often complex and fraught with local optima. These difficulties have made evolutionary computation (EC) an attractive approach to learning multiagent behaviors (for example, [Iba, 1996; Luke *et al.*, 1998; Wu *et al.*, 1999; Bull *et al.*, 1995; Bassett and De Jong, 2000; Bull, 1997]). This work has led to interesting research questions in applying EC in a multiagent setting, including communication, representation, generalization, teamwork, and collaboration strategies.

As it is very general (and relatively knowledge-poor), evolutionary computation is particularly useful in problems that are of high dimensionality, are non-Markovian, or yield few heuristic clues about the search space that otherwise would make reinforcement learning or various supervised learning methods good choices. We believe that multiagent learning problem domains often exhibit such features. These problem domains are often complex and “correct” actions cannot be known beforehand in a given situation. Further, even relatively simple problems can require large numbers of external and, even more challenging, internal state variables. Last, many such problems exhibit changing environments, even ones that adapt to make the problem harder for the learner (due to the presence of co-learning opponents).

EC fits nicely with multiagent systems because it is already *population-oriented*: it searches over a set of multiple agents (the individuals). Further, an EC population may be broken down into distinct subpopulations, each yielding agents to be tested together in a multiagent environment, with each subpopulation “evolving” in parallel. This notion of separately evolving, interacting populations of agents is known as *coevolution*. Coevolution has proven a useful technique for multiagent problems where the quality of agent is typically assessed in the context of competing or cooperating peers.

But coevolution is no panacea. Recent research has shown that a coevolutionary system does not necessarily search for better teams of agents, but can instead search for agent populations that represent *stable equilibria* in the cooperative search space [Ficici and Pollack, 2000; Wiegand *et al.*, 2002b]. This paper will explore this problem, then introduce a method for biasing coevolution so that the search for stability coincides with optimization for improvement.

We continue this paper with a brief description of coevolution and present an experimental and a theoretical framework. We then suggest a method for biasing the coevolutionary process, describe a theoretical investigation on how biasing modifies the search space, and discuss experimental results on three problem domains. The paper ends with a set of conclusions and directions for future work.

2 Evolutionary Computation and Coevolution

Evolutionary computation is a family of techniques, known as *evolutionary algorithms*, widely used for learning agent behaviors. In EC, abstract Darwinian models of evolution are applied to refine populations of agents (known as individuals) representing candidate solutions to a given problem. An evolutionary algorithm begins with an initial population of randomly-generated agents. Each member of this population is then evaluated and assigned a fitness (a quality assessment). The EA then uses a fitness-oriented procedure to select agents, breeds and mutates them to produce child agents, which are then added to the population, replacing older agents. One evaluation, selection, and breeding cycle is known as a generation. Successive generations continue to refine the population until time is exhausted or a sufficiently fit agent is discovered.

Coevolutionary algorithms (CEAs) represent a natural approach to applying evolutionary computation to refine mul-

tiagent behaviors. In a CEA, the fitness of an individual is based on its interaction with other individuals in the population: thus the fitness assessment is context-sensitive and subjective. In *competitive* systems, agents benefit at the expense of other agents; but in *cooperative* systems, agents succeed or fail together in collaboration. The focus of this paper is in cooperative coevolutionary algorithms. Interesting CEA issues include communication [Bull *et al.*, 1995], teamwork, and collaboration [Bull, 1997].

A standard approach [Potter, 1997] to applying cooperative coevolutionary algorithms (or CCEAs) to an optimization problem starts by identifying a static decomposition of the problem representation into subcomponents, each represented by a separate population of individuals. For example, if a task requires two agents whose collaboration must be optimized, one might choose to use two populations, one per agent in the task. The fitness of an individual in a population is then determined by testing the individual in collaboration with one or more individuals from the other population. Aside from this collaborative assessment, each population follows its own independent evolution process in parallel with other populations.

2.1 Formalizing the CCEA

An appealing abstract mathematical model for this system comes from the Biology literature: Evolutionary Game Theory (EGT) [Maynard-Smith, 1982; Hofbauer and Sigmund, 1998]. EGT provides a formalism based on traditional game theory and dynamical systems techniques to analyze the limiting behaviors of interacting populations under long-term evolution. For specifics about applying EGT to the analysis of multi-population cooperative coevolutionary algorithms, see [Wiegand *et al.*, 2002a].

In this paper, we consider only two-population models. In such a model, a common way of expressing the rewards from individual interactions is through a pair of *payoff matrices*. We assume a symmetric model such that when individuals from the first population interact with individuals from the second, one payoff matrix A is used, while individuals from the second population receive rewards defined by the transpose of this matrix (A^T). In our theoretical exploration of EGT in this paper, we will use an *infinite population*: thus a population can be thought of not as a set of individuals, but rather as a finite-length vector $\vec{x}$ of proportions, where each element in the vector is the proportion of a given individual configuration (popularly known as a *genotype* or, as we will term it, a *strategy*) in the population. As the proportions in a valid vector must sum to one, all legal vectors make up what is commonly known as the *unit simplex*, denoted Δ^n , where n here is the number of distinct strategies possible, $\vec{x} \in \Delta^n : x_i \in [0, 1], \sum_{i=1}^n x_i = 1$.

Formally we can model the effects of evaluation and proportional selection over time using a pair of difference equations, one for each population. The proportion vectors for the two populations are $\vec{x}$ and $\vec{y}$ respectively. Neglecting the issue of mutation and breeding and concentrating only on the effects of selection, we can define the dynamical system of a two-population cooperative coevolutionary algorithm as:

$$\vec{u} = A\vec{y} \quad (1)$$

$$\vec{w} = A^T\vec{x} \quad (2)$$

$$x'_i = \left(\frac{u_i}{\vec{x} \cdot A\vec{y}} \right) x_i \quad (3)$$

$$y'_i = \left(\frac{w_i}{\vec{y} \cdot A^T\vec{x}} \right) y_i \quad (4)$$

...where $\vec{x}'$ and $\vec{y}'$ represent the new population distributions for the next generation. Here it is assumed that an individual's fitness is assessed through pair-wise collaborations with *every* member of the cooperating population. We call this idea *complete mixing*. The equations above describe a two-step process. First, the vectors $\vec{u}$ and $\vec{w}$ are derived; these represent the fitness assessments of strategies in the generations $\vec{x}$ and $\vec{y}$ respectively. Note that an infinite population model considers the fitness assessment for a strategy, and not for a particular instance of that strategy (an individual). Then selection is performed by computing the proportion of the fitness of a specific strategy over the sum fitness of the entire population.

2.2 Optimization versus Balance

CCEA researchers apply these algorithms hoping to *optimize* the collaborations between the populations, but it isn't clear that this system is meant to do this. In fact, the system seeks a form of *balance* between strategies, which may not correspond with what we, as external viewers of the system, would consider optimal. In the context of a payoff matrix, an optimal position is the pair of strategies that yield the highest payoff for the cooperating agents. This position is a stable attracting fixed point of such a system; but it is also the case that there are other suboptimal points, which can also attract trajectories [Wiegand *et al.*, 2002b]. Indeed, it is possible that most, if not all, trajectories can be pulled toward suboptimal spots. These points correspond to Nash equilibria: suboptimal combinations of strategies where if any *one* strategy is changed, the net reward for both agents will decrease.

As a result, individuals in a CCEA are not necessarily refined to be the optimal subcomponent of the optimal component; instead they are refined to be jacks-of-all-trades that dovetail nicely with the *current individuals from the other population*. What does this mean for practitioners wanting to coevolve "optimal" (or perhaps, even "good") cooperative strategies using a coevolutionary algorithm? It means that CEAs are *not* necessarily optimizers in the sense that one might intuitively expect them to be. Something must be done to modify the existing algorithms or our expectations of what these algorithms really do.

3 Biasing for Optimal Cooperation

One reason CCEAs tend toward "balance" is that an individual's fitness is commonly assessed based on how well it performs with immediate individuals from the other population. To find optimal cooperation, the search process may need to be more optimistic than this: assessing fitness based more on the *highest-reward* interactions between an individual and various members of the other population. A previous investigation in this direction is reported in [Wiegand *et al.*, 2001]:

		Agent 2					Agent 2		
Agent 1		a	b	c	Agent 1		a	b	c
	a	11	penalty	0		a	10	0	penalty
	b	penalty	7	6		b	0	2	0
	c	0	0	5		c	penalty	0	10

Table 1: Joint reward matrixes for the Climb (left) and Penalty (right) domains.

assessing an individual’s fitness based on its *maximum* performance with other agents in a collaborative domain was shown to yield better results than when using the mean or minimum performance. The idea presented in this paper is relatively simple: base an individual’s fitness on a combination of its immediate reward while interacting with individuals in the population, and on an estimate for the reward it would have received had it interacted with its *ideal collaborators*. The fraction of reward due to the immediate (as opposed to the ideal) interaction changes during the course of the run.

We note that this notion of bias towards maximum possible reward has also been used in the reinforcement learning literature in subtly different ways than we use it here. For example, maximum reward was used by [Claus and Boutilier, 1998] to modify the exploration strategy of the agent, and by [Lauer and Riedmiller, 2000] to modify the update rule for the Q table. To some extent, the “Hall of Fame” method introduced by [Rosin and Belew, 1997] for competitive coevolution is also related to biased cooperative coevolution.

We justify the use of such a bias in a CCEA as follows. Recall that if an individual’s fitness is based on its immediate interaction with individuals from the other population, then $\vec{u} = A\vec{y}$ and $\vec{w} = A^T\vec{x}$, as described in equations 1 and 2. Now, let us consider a function $\max_A$ that returns a column vector corresponding to the maximum value of each row in matrix A . Now, if an individual’s fitness is based on its maximum possible performance in conjunction with any individual from the other population, then we may modify equations 1 and 2 to be $\vec{u} = \max_A^T$ and $\vec{w} = \max_{A^T}$.

In this modified system, the tendency to optimize performance is clear. At each iteration of the model, the fitness of each strategy will be its best possible fitness. If there is a unique maximum, that result will have the highest fitness and so the proportion of the corresponding strategy will increase in the next step. When the global maxima are not unique, the resulting fixed point is a mixed strategy, with weights split between those maxima.

The reason for this is straightforward: the problem has lost the dimensionality added due to the nature of the interactions between the agents. Without this, the problem reduces to a simple evolutionary algorithm: regardless of the content of the opposing population, the fitness measure for a given strategy is the same. As shown in [Vose, 1999], an infinite population model of this reduced evolutionary algorithm will converge to a unique global maximum.

But it is difficult to imagine how a real CCEA algorithm would know the maximum possible reward for a given individual *a priori*. One approach is to use historical information during the run to approximate the maximum possible collabora-

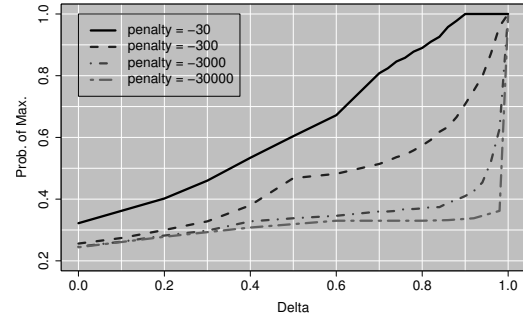


Figure 1: Probability of converging to the optimum as the bias parameter δ is varied between 0 and 1.

ative fitness for an individual. However, if the approximation is too large (or has too strong an effect on the overall fitness), and if it appears too early in the evolutionary run, then it can deform the search space to drive search trajectories into sub-optimal parts of the space from which they cannot escape. On the other hand, if the approximation affects the fitness measurement very weakly, and too late in the run, then it may not be of much help, and the system will still gravitate towards “balance”.

To better see this tradeoff, we again alter equations 1 and 2, this time adding a bias weight parameter δ . Now, $\vec{u} = (1 - \delta) \cdot A\vec{y} + \delta \cdot \max_A^T$ and $\vec{w} = (1 - \delta) \cdot A^T\vec{x} + \delta \cdot \max_{A^T}$. Varying δ between 0 and 1 will control the degree to which the model makes use of the bias. Consider the *Climb* payoff matrix on the left side of Table 1. We select 500 initial points of the dynamical system uniformly at random from $\Delta^n \times \Delta^m$, and iterate the system until it converges. While convergence is virtually guaranteed in traditional two-matrix EGT games [Hofbauer and Sigmund, 1998], it is not necessarily guaranteed in our modified system. In our experimental results, however, we obtained convergence in all cases to within some degree of machine precision. Figure 1 shows the probability, for various levels of δ , of the dynamical system converging to the optimum when the penalty is set to -30 , -300 , -3000 or -30000 . Notice that, as the penalty worsens, the transition between optimal and suboptimal convergence becomes more severe. This suggests that for some problems, any benefits provided by this type of bias may be quite sensitive to the degree of bias.

4 Experiments

While this theoretical discussion helps justify our intuition for including a performance bias in fitness evaluation, it is not immediately applicable to real problems. In a more realistic setting, simplifying model assumptions such as infinite populations, lack of variational operators, complete mixing, and *a priori* knowledge of the maximum payoff are not possible. To convert theory into practice, we have adopted an approximation to the performance bias that is based on historical information gathered during the evolutionary run. We also decreased the bias through the course of a run to take advantage of the fact that initial partners are likely to be weak, while later partners are stronger.

We performed several experiments to compare simple co-

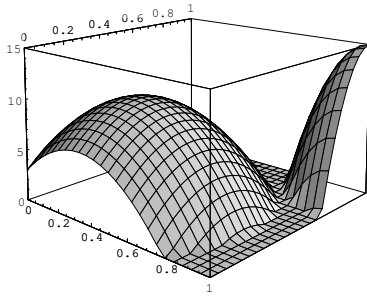


Figure 2: Joint reward in the continuous Two Peaks domain

evolution (SC) with biased coevolution (BC) in three problem domains detailed later. Both SC and BC base fitness on the immediate performance of an individual in the context of individuals from one other cooperating population. BC additionally includes a bias factor: part of the fitness is based on an approximation of what an individual’s fitness would be were it to cooperate with its ideal partners.

We compared these two techniques in combination with two approaches to representing an individual. In the Pure Strategy Representation (PSR), an individual represented a single strategy. PSR individuals stored a single integer representing the strategy in question. A PSR individual bred children through mutation: a coin was repeatedly tossed, and the individual’s integer was increased or decreased (the direction chosen at random beforehand) until the coin came up heads. In the Mixed Strategy Representation (MSR), an individual represented not a single strategy but a probability distribution over all possible strategies. When evaluating an MSR individual with a partner agent, 50 independent trials were performed, and each time each agent’s strategy was chosen at random from the agent’s probability distribution. MSR individuals used one-point crossover, followed by adding random Gaussian noise ($\mu = 0, \sigma = 0.05$) to each of the distribution values, followed by renormalization of the distribution. Observe that using MSR creates a potentially more difficult problem domain than using PSR, for reasons of search space size and stochasticity of the fitness result.

We chose a common approach to cooperative coevolution fitness assessment. An individual is assessed twice to determine fitness: once with a partner chosen at random, then once partnered with the individual in the other population that had received the highest fitness in the previous generation. An individual’s fitness is set to the maximum of these two assessments. During a fitness assessment, an individual receives some number of *rewards* for trying certain strategies in the context of partners. For a PSR individual, the assessment was simply the single reward it received for trying its strategy with its partners. As an MSR individual tried fifty strategies, its assessment was the mean of the fifty rewards it received.

SC and BC differ in that BC adds into the reward a bias term, that is, $Reward \leftarrow (1 - \delta) \cdot Reward + \delta \cdot MaxReward$, where δ is a decreasing bias rate that starts at 1.0 and linearly decreases until it reaches 0 when 3/4 of the maximal run length has passed. Ideally, the *MaxReward* bias factor would be the highest possible reward received for trying that partic-

	Penalty			
	-30	-300	-3000	-30000
SC+MSR	21% 7.58	16% 7.04	16% 4.61	15% -7.69
BC+MSR	100% 10.6	100% 10.0	100% 9.17	97% 1.25
SC+PSR	100% 11.0	100% 11.0	100% 11.0	100% 11.0
BC+PSR	100% 11.0	100% 11.0	100% 11.0	100% 11.0

Table 2: Proportion of runs that converged to global optimum and average best individual fitness, Climbing Domain

	Penalty			
	-10	-100	-1000	-10000
SC+MSR	100% 9.69	66% 6.92	60% 6.24	61% 3.55
BC+MSR	100% 9.71	100% 9.40	100% 8.99	100% 3.36
SC+PSR	100% 10.0	100% 10.0	100% 10.0	100% 10.0
BC+PSR	100% 10.0	100% 10.0	100% 10.0	100% 10.0

Table 3: Proportion of runs that converged to global optimum and average best individual fitness, Penalty Domain

ular strategy, over all possible partner strategies. In the experiments in this paper, we chose to approximate *MaxReward* by setting it to the maximum reward seen so far in the run for the given strategy.

In all experiments, the most fit individual survived automatically from one generation to the next. To select an individual for breeding, we chose two individuals at random with replacement from the population, then selected the fitter of the two. Each experiment was repeated 100 times. The experiments used the ECJ9 software package [Luke, 2002].

4.1 Problem Domains

We experimented with three different single-stage game domains: two simpler ones (Climb and Penalty) introduced in [Claus and Boutilier, 1998], and a more complex artificial problem (Two Peaks). Evolutionary runs in the Climb and Penalty problem domain lasted 200 generations and used 20 individuals per population. Runs in the Two Peaks domain lasted 500 generations and used populations of 100 individuals each.

The joint reward matrices for the Climb and the Penalty domains are presented in Table 1. The domains are difficult because of the penalties associated with miscoordinated actions and the presence of suboptimal collaborations that avoid penalties. Figure 2 presents a continuous version of the Two Peaks coordination game, where the x and y axes represent the continuous range of actions for the two agents, and the z axis shows the joint reward. The reward surface has two peaks, one lower but spread over a large surface, and the other one higher but covering a small area. Because an agent’s strategy space is continuous over $[0, 1]$, we discretized it into increasingly difficult sets of 8, 16, 32, 64 or 128 strategies. The discretizations result in slightly different optimal values.

5 Results

Tables 2–4 present the proportion (out of 100 runs) that converged to the global optimum, plus the mean fitness of the best individuals in the runs. MSR individuals were considered optimal if and only if the optimal strategy held over 50

	Discretization Level (Number of Actions)				
	8	16	32	64	128
SC+MSR	0% 8.98	0% 7.48	0% 6.70	0% 6.47	0% 6.36
BC+MSR	0% 8.99	0% 7.51	0% 6.70	0% 6.47	0% 6.36
SC+PSR	100% 11.0	41% 11.6	32% 11.5	42% 12.1	48% 12.4
BC+PSR	100% 11.0	71% 12.8	72% 13.4	61% 13.0	70% 13.0

Table 4: Proportion of runs that converged to global optimum and average best individual fitness, Two Peaks Domain

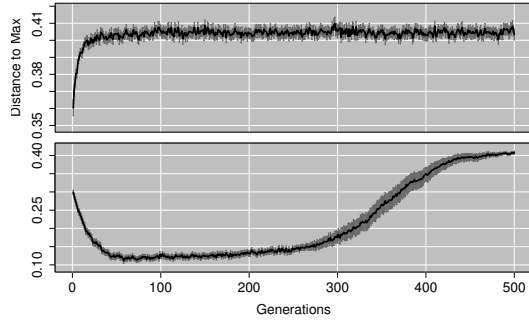


Figure 3: Distance from best-of-generation individuals to optimal strategy for the 8 actions Two Peaks domain using SC (top) and BC (bottom).

percent of the distribution (in fact, most optimal MSR individuals had over 90 percent).

Biased coevolution consistently found the global optima as often as, or more often than, standard coevolution. The only times where standard coevolution held its own was in the Climbing and Penalty domains, where PSR individuals found the optimum 100% of the time, as well as in the harder Two Peaks domain, where no MSR individuals found the optimum. For those problems when individuals found the optimum less than 100% of the time, we also compared differences in mean best fitness of a run, using a two-factor ANOVA with repetitions, factored over the method used and the problem domain.

The ANOVA results allow us to state with 95% confidence that biased coevolution is better than simple coevolution when MSR is used in the Climbing domain, and also in the Two Peaks domain when PSR is used; the tests give only a 90% confidence for stating that BC+MSR is better than SC+MSR in the Penalty domain.

In order to better understand what happens when using MSR in the Two Peaks domains, we plotted the average euclidian distance from the best individual per generation to the known global optima (Figures 3 and 4). The graphs present the 95% confidence interval for the mean of the fitnesses. Investigations showed that SC converged to suboptimal interactions (the lower, wider peak in Figure 2) in all cases. On the other hand, the trajectories of the search process are radically different when using BC. Let's take a closer look as to why this might be so.

As we learned from our discussion surrounding Figure 1,

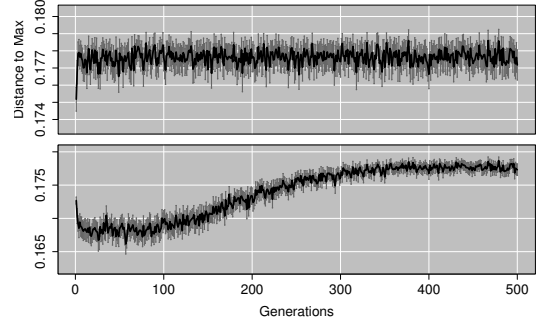


Figure 4: Distance from best-of-generation individuals to optimal strategy for the 32 actions Two Peaks domain using SC (top) and BC (bottom).

successful applications of this biasing method are tied to successfully determining the appropriate degree of bias to apply. Due to MSR's increased difficulty, it may be more challenging to find an appropriate balance for the bias. Figure 3 suggests exactly this. Notice that, in the early part of the run (when δ is strong), the algorithm tends towards the optimal solution; however, as the bias is reduced, it becomes overwhelmed and the trajectories are eventually drawn toward the suboptimal local attractor. Moreover, as the problem becomes larger (i.e., Figure 4, as well as others not shown), this failure occurs earlier in the run. This suggests more careful attention is needed to set the parameters and to adjust the bias rate when using MSR versus PSR. Indeed, by running longer and allowing for more interactions during evaluation, we were able to obtain convergence to the global optimum when using MSR (not shown).

6 Conclusions and Future Work

Although cooperative coevolution has been successfully applied to the task of learning multiagent behaviors, as research about these algorithms advances, it becomes increasingly clear that these algorithms may favor stability over optimality for some problem domains. In this paper, we develop a very simple idea: improve coevolution through the use of a maximum reward bias. We introduce a theoretical justification for the idea, then present experimental evidence that confirms that biasing coevolution can yield significantly better results than standard coevolution when searching for optimal collaborations. Our work further reveals that domain features greatly influence the levels of biasing necessary for convergence to optima: for some problems the performance changes slowly when the level of bias is modified, while for other domains there is a rapid degradation in results. This suggests that, while adding some kind of maximum reward bias can be helpful, there is still work to be done in understanding how best to apply this bias in different problem domains.

Our initial experimental results in this paper suggest that it is effective to use a history as an approximation to the true maximal collaborative reward for a given strategy. For future

work we intend to extend these experiments to problem domains with search spaces much larger than the ones used in these experiments. In such domains, the number of strategies may be very large, even infinite. Keeping an effective history of strategies may thus be infeasible in certain circumstances; we intend to explore ways to sample the space or cache the most significant strategy results. Repeated games, such as the Iterated Prisoner's Dilemma, or stochastic games, may also require different approaches to biasing coevolution. Understanding these issues, we hope, can lead to significant improvements in cooperative coevolution's effectiveness as a multi-agent optimization technique.

7 Acknowledgements

This research was partially supported through Department of Army grant DAAB07-01-9-L504, ONR grant N00014-01-1-0703, and a gift from SRA International.

References

- [Bassett and De Jong, 2000] Jeffrey K. Bassett and Kenneth A. De Jong. Evolving behaviors for cooperating agents. In Z. Ras, editor, *Proceedings from the Twelfth International Symposium on Methodologies for Intelligent Systems*, pages 157–165, Charlotte, NC., 2000. Springer-Verlag.
- [Bull *et al.*, 1995] Lawrence Bull, Terrence C. Fogarty, and M. Snaith. Evolution in multi-agent systems: Evolving communicating classifier systems for gait in a quadrupedal robot. In L. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA)*, pages 382–388. Morgan Kaufmann, 1995.
- [Bull, 1997] L. Bull. Evolutionary computing in multi-agent environments: Partners. In Thomas Baeck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA)*, pages 370–377. Morgan Kaufmann, 1997.
- [Claus and Boutilier, 1998] Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-98)*. MIT Press, 1998.
- [Ficici and Pollack, 2000] S. Ficici and J. Pollack. A game-theoretic approach to the simple coevolutionary algorithm. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lut-ton, J.J. Merelo, and H.-P. Schwefel, editors, *Proceedings of the Sixth International Conference on Parallel Problem Solving from Nature (PPSN VI)*, pages 467–476. Springer-Verlag, 2000.
- [Hofbauer and Sigmund, 1998] Josef Hofbauer and Karl Sigmund. *Evolutionary Games and Population Dynamics*. Cambridge University Press, 1998.
- [Iba, 1996] H. Iba. Emergent cooperation for multiple agents using genetic programming. In J. Koza, editor, *Late Breaking Papers of the Genetic Programming Conference*, pages 66–74, Stanford, CA, 1996. Stanford University Book-store.
- [Lauer and Riedmiller, 2000] M. Lauer and M. Riedmiller. An algorithm for distributed reinforcement learning in co-operative multi-agent systems. In *Proceedings of the Seventeenth International Conference in Machine Learning*, 2000.
- [Luke *et al.*, 1998] S. Luke, C. Hohn, J. Farris, G. Jackson, and J. Hendler. Co-evolving soccer softbot team coordination with genetic programming. In *RoboCup-97: Robot Soccer World Cup I (Lecture Notes in AI No. 1395)*, pages 398–411, Berlin, 1998. Springer-Verlag.
- [Luke, 2002] Sean Luke. ECJ 9: A Java EC research system. <http://www.cs.umd.edu/projects/plus/ec/ecj/>, 2002.
- [Maynard-Smith, 1982] John Maynard-Smith. *Evolution and the Theory of Games*. Cambridge University Press, 1982.
- [Potter, 1997] M. Potter. *The Design and Analysis of a Computational Model of Cooperative CoEvolution*. PhD thesis, George Mason University, Fairfax, Virginia, 1997.
- [Rosin and Belew, 1997] C. Rosin and R. Belew. New methods for competitive coevolution. *Evolutionary Computation*, 5(1):1–29, 1997.
- [Vose, 1999] M. Vose. *The Simple Genetic Algorithm*. MIT Press, 1999.
- [Wiegand *et al.*, 2001] R. Paul Wiegand, William Liles, and Kenneth De Jong. An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In L. Spector *et al.*, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2001*, pages 1235–1242. Morgan Kaufmann, 2001.
- [Wiegand *et al.*, 2002a] R. Paul Wiegand, William Liles, and Kenneth De Jong. Analyzing cooperative coevolution with evolutionary game theory. In D. Fogel, editor, *Proceedings of CEC 2002*, pages 1600–1605. IEEE Press, 2002.
- [Wiegand *et al.*, 2002b] R. Paul Wiegand, William Liles, and Kenneth De Jong. Modeling variation in cooperative coevolution using evolutionary game theory. In Richardo Poli, Jonathan Rowe, and Kenneth De Jong, editors, *Foundations of Genetic Algorithms (FOGA) VII*, pages 231–248. Morgan Kaufmann, 2002.
- [Wu *et al.*, 1999] Annie Wu, Alan C. Schultz, and Arvin Agah. Evolving control for distributed micro air vehicles. In *IEEE International Symposium on Conference on Computational Intelligence in Robotics and Automation (CIRA-99)*, 1999.

MULTIAGENT SYSTEMS

GAME PLAYING

Approximating Game-Theoretic Optimal Strategies for Full-scale Poker

D. Billings, N. Burch, A. Davidson, R. Holte, J. Schaeffer, T. Schauenberg, and D. Szafron

Department of Computing Science, University of Alberta

Edmonton, Alberta, T6G 2E8, Canada

Email: {darse,burch,davidson,holte,jonathan,terence,duane}@cs.ualberta.ca

Abstract

The computation of the first complete approximations of game-theoretic optimal strategies for full-scale poker is addressed. Several abstraction techniques are combined to represent the game of 2-player Texas Hold'em, having size $O(10^{18})$, using closely related models each having size $O(10^7)$. Despite the reduction in size by a factor of 100 billion, the resulting models retain the key properties and structure of the real game. Linear programming solutions to the abstracted game are used to create substantially improved poker-playing programs, able to defeat strong human players and be competitive against world-class opponents.

1 Introduction

Mathematical game theory was introduced by John von Neumann in the 1940s, and has since become one of the foundations of modern economics [von Neumann and Morgenstern, 1944]. Von Neumann used the game of poker as a basic model for 2-player zero-sum adversarial games, and proved the first fundamental result, the famous *minimax theorem*. A few years later, John Nash added results for N -player non-cooperative games, for which he later won the Nobel Prize [Nash, 1950]. Many decision problems can be modeled using game theory, and it has been employed in a wide variety of domains in recent years.

Of particular interest is the existence of *optimal solutions*, or *Nash equilibria*. An optimal solution provides a randomized mixed strategy, basically a recipe of how to play in each possible situation. Using this strategy ensures that an agent will obtain at least the game-theoretic value of the game, regardless of the opponent's strategy. Unfortunately, finding exact optimal solutions is limited to relatively small problem sizes, and is not practical for most real domains.

This paper explores the use of highly abstracted mathematical models which capture the most essential properties of the real domain, such that an exact solution to the smaller problem provides a useful approximation of an optimal strategy for the real domain. The application domain used is the game of poker, specifically Texas Hold'em, the most popular form of casino poker and the poker variant used to determine the world champion at the annual World Series of Poker.

Due to the computational limitations involved, only simplified poker variations have been solved in the past (*e.g.* [Kuhn, 1950; Sakaguchi and Sakai, 1992]). While these are of theoretical interest, the same methods are not feasible for real games, which are too large by many orders of magnitude ([Koller and Pfeffer, 1997]).

[Shi and Littman, 2001] investigated abstraction techniques to reduce the large search space and complexity of the problem, using a simplified variant of poker. [Takusagawa, 2000] created near-optimal strategies for the play of three specific Hold'em flops and betting sequences. [Selby, 1999] computed an optimal solution for the abbreviated game of *preflop Hold'em*.

Using new abstraction techniques, we have produced viable "pseudo-optimal" strategies for the game of 2-player Texas Hold'em. The resulting poker-playing programs have demonstrated a tremendous improvement in performance. Whereas the previous best poker programs were easily beaten by any competent human player, the new programs are capable of defeating very strong players, and can hold their own against world-class opposition.

Although some domain-specific knowledge is an asset in creating accurate reduced-scale models, analogous methods can be developed for many other imperfect information domains and generalized game trees. We describe a general method of problem reformulation that permits the independent solution of sub-trees by estimating the conditional probabilities needed as input for each computation.

This paper makes the following contributions:

1. Abstraction techniques that can reduce an $O(10^{18})$ poker search space to a manageable $O(10^7)$, without losing the most important properties of the game.
2. A poker-playing program that is a major improvement over previous efforts, and is capable of competing with world-class opposition.

2 Game Theory

Game theory encompasses all forms of competition between two or more agents. Unlike chess or checkers, poker is a game of *imperfect information* and *chance outcomes*. It can be represented with an *imperfect information game tree* having *chance nodes* and *decision nodes*, which are grouped into *information sets*.

Since the nodes in this tree are not independent, divide-and-conquer methods for computing sub-trees (such as the *alpha-beta* algorithm) are not applicable. For a more detailed description of imperfect information game tree structure, see [Koller and Megiddo, 1992].

A *strategy* is a set of rules for choosing an action at every decision node of the tree. In general, this will be a *randomized mixed strategy*, which is a probability distribution over the various alternatives. A player must use the same policy across all nodes in the same information set, since from that player's perspective they are indistinguishable from each other (differing only in the hidden information component).

The conventional method for solving such a problem is to convert the descriptive representation, or *extensive form*, into a system of linear equations, which is then solved by a linear programming (LP) system such as the *Simplex algorithm*. The optimal solutions are computed simultaneously for all players, ensuring the best worst-case outcome for each player.

Traditionally, the conversion to *normal form* was accompanied by an exponential blow-up in the size of the problem, meaning that only very small problem instances could be solved in practice. [Koller *et al.*, 1994] described an alternate LP representation, called *sequence form*, which exploits the common property of *perfect recall* (wherein all players know the preceding history of the game), to obtain a system of equations and unknowns that is only linear in the size of the game tree. This exponential reduction in representation has re-opened the possibility of using game-theoretic analysis for many domains. However, since the game tree itself can be very large, the LP solution method is still limited to moderate problem sizes (normally less than a billion nodes).

3 Texas Hold'em

A game (or *hand*) of Texas Hold'em consists of four stages, each followed by a round of betting:

Preflop: Each player is dealt two private cards face down (the *hole cards*).

Flop: Three *community cards* (shared by all players) are dealt face up.

Turn: A single community card is dealt face up.

River: A final community card is dealt face up.

After the betting, all active players reveal their hole cards for the *showdown*. The player with the best five-card poker hand formed from their two private cards and the five public cards wins all the money wagered (ties are possible).

The game starts off with two forced bets (the *blinds*) put into the *pot*. When it is a player's turn to act, they must either *bet/raise* (increase their investment in the pot), *check/call* (match what the opponent has bet or raised), or *fold* (quit and surrender all money contributed to the pot).

The best-known non-commercial Texas Hold'em program is *Poki*. It has been playing online since 1997 and has earned an impressive winning record, albeit against generally weak opposition [Billings *et al.*, 2002]. The system's abilities are based on enumeration and simulation techniques, expert knowledge, and opponent modeling. The program's weaknesses are easily exploited by strong players, especially in the 2-player game.

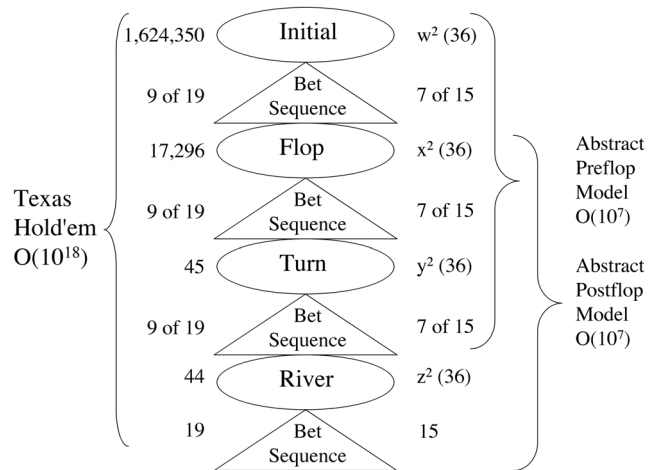


Figure 1: Branching factors for Hold'em and abstractions.

4 Abstractions

Texas Hold'em has an easily identifiable structure, alternating between chance nodes and betting rounds in four distinct stages. A high-level view of the imperfect information game tree is shown in Figure 1.

Hold'em can be reformulated to produce similar but much smaller games. The objective is to reduce the scale of the problem without severely altering the fundamental structure of the game, or the resulting optimal strategies. There are many ways of doing this, varying in the overall reduction and in the accuracy of the resulting approximation.

Some of the most accurate abstractions include *suit equivalence isomorphisms* (offering a reduction of at most a factor of $4! = 24$), *rank equivalence* (only under certain conditions), and *rank near-equivalence*. The optimal solutions to these abstracted problems will either be exactly the same or will have a small bounded error, which we refer to as *near-optimal* solutions. Unfortunately, the abstractions which produce an exact or near-exact reformulation do not produce the very large reductions required to make full-scale poker tractable.

A common method for controlling the game size is *deck reduction*. Using less than the standard 52-card deck greatly reduces the branching factor at chance nodes. Other methods include reducing the number of cards in a player's hand (*e.g.* from a 2-card hand to a 1-card hand), and reducing the number of board cards (*e.g.* a 1-card flop), as was done by [Shi and Littman, 2001] for the game of *Rhode Island Hold'em*. [Koller and Pfeffer, 1997] used such parameters to generate a wide variety of tractable games to solve with their Gala system.

We have used a number of small and intermediate sized games, ranging from eight cards (two suits, four ranks) to 24 cards (three suits, eight ranks) for the purpose of studying abstraction methods, comparing the results with known exact or near-optimal solutions. However, these smaller games are not suitable for use as an approximation for Texas Hold'em, as the underlying structures of the games are different. To produce good playing strategies for full-scale poker, we look for abstractions of the real game which do not alter that basic

structure.

The abstraction techniques used in practice are powerful in terms of reducing the problem size, and subsume those previously mentioned. However, since they are also much cruder, we call their solutions *pseudo-optimal*, to emphasize that there is no guarantee that the resulting approximations will be accurate, or even reasonable. Some will be low-risk propositions, while others will require empirical testing to determine if they have merit.

4.1 Betting round reduction

The standard rules of limit Hold'em allow for a maximum of four bets per player per round.¹ Thus in 2-player limit poker there are 19 possible betting sequences, of which two do not occur in practice.² Of the remaining 17 sequences, 8 end in a fold (leading to a terminal node in the game tree), and 9 end in a call (carrying forward to the next chance node). Using $k = \text{check}$, $b = \text{bet}$, $f = \text{fold}$, $c = \text{call}$, $r = \text{raise}$, and capital letters for the second player, the tree of possible betting sequences for each round is:

kK kBf kBc kBrf kBrC kBrRf kBrRc kBrRf kBrRc
bF bC bRf bRc bRf bRc bRf bRc

We call this local collection of decision nodes a *betting tree*, and represent it diagrammatically with a triangle.

With *betting round reduction*, each player is allowed a maximum of three bets per round, thereby eliminating the last two sequences in each line. The effective branching factor of the betting tree is reduced from nine to seven. This does not appear to have a substantial effect on play, or on the expected value (EV) for each player. This observation has been verified experimentally. In contrast, we computed the corresponding postflop models with a maximum of two bets per player per round, and found radical changes to the optimal strategies, strongly suggesting that that level of abstraction is not safe.

4.2 Elimination of betting rounds

Large reductions in the size of a poker game tree can be obtained by *elimination of betting rounds*. There are several ways to do this, and they generally have a significant impact on the nature of the game. First, the game may be *truncated*, by eliminating the last round or rounds. In Hold'em, ignoring the last board card and the final betting round produces a 3-round model of the actual 4-round game. The solution to the 3-round model loses some of the subtlety involved in the true optimal strategy, but the degradation applies primarily to advanced tactics on the turn. There is a smaller effect on the flop strategy, and the strategy for the first betting round may have no significant changes, since it incorporates all the outcomes of two future betting rounds. We use this particular abstraction to define an appropriate strategy for play in the first round, and thus call it a *preflop model* (see Figure 2).

¹Some rules allow unlimited raises when only two players are involved. However, occasions with more than three legitimate raises are relatively rare, and do not greatly alter an optimal strategy.

²Technically, a player may fold even though there is no outstanding bet. This is logically dominated by not folding, and therefore does not occur in an optimal strategy, and is almost never seen in practice.

The effect of truncation can be lessened through the use of *expected value leaf nodes*. Instead of ending the game abruptly and awarding the pot to the strongest hand at that moment, we compute an average conclusion over all possible chance outcomes. For a 3-round model ending on the turn, we *roll-out* all 44 possible river cards, assuming no further betting (or alternately, assuming one bet per player for the last round). Each player is awarded a fraction of the pot, corresponding to their probability of winning the hand. In a 2-round preflop model, we roll-out all 990 2-card combinations of the turn and river.

The most extreme form of truncation results in a 1-round model, with no foresight of future betting rounds. Since each future round provides a refinement to the approximation, this will not reflect a correct strategy for the real game. In particular, betting plans that extend over more than one round, such as deferring the raise of a very strong hand, are lost entirely. Nevertheless, even these simplistic models can be useful when combined with expected value leaf nodes.

Alex Selby computed an optimal solution for the game of *preflop Hold'em*, which consists of only the first betting round followed by an EV roll-out of the five board cards to determine the winner [Selby, 1999]. Although there are some serious limitations in the strategy based on this 1-round model, we have incorporated the Selby preflop system into one of our programs, *PsOpt1*, as described later in this section.

In contrast to truncating rounds, we can *bypass* certain early stages of the game. We frequently use *postflop models*, which ignore the preflop betting round, and use a single fixed flop of three cards (see Figure 1).

It is natural to consider the idea of *independent betting rounds*, where each phase of the game is treated in isolation. Unfortunately, the betting history from previous rounds will almost always contain contextual information that is critical to making appropriate decisions. The probability distribution over the hands for each player is strongly dependent on the path that led to that decision point, so it cannot be ignored without risking a considerable loss of information. However, the naive independence assumption can be viable in certain circumstances, and we do implicitly use it in the design of *PsOpt1* to bridge the gap between the 1-round preflop model and the 3-round postflop model.

Another possible abstraction we explored was *merging* two or more rounds into a single round, such as creating a combined 2-card turn/river. However, it is not clear what the appropriate bet size should be for this composite round. In any case, the solutions for these models (over a full range of possible bet sizes), all turned out to be substantially different from their 3-round counterparts, and the method was therefore rejected.

4.3 Composition of preflop and postflop models

Although the nodes of an imperfect information game tree are not independent in general, some decomposition is possible. For example, the sub-trees resulting from different preflop betting sequences can no longer have nodes that belong to the

same information set.³ The sub-trees for our postflop models can be computed in isolation, provided that the appropriate preconditions are given as input. Unfortunately, knowing the correct conditional probabilities would normally entail solving the whole game, so there would be no advantage to the decomposition.

For simple postflop models, we dispense with the prior probabilities. For the postflop models used in *PsOpti0* and *PsOpti1*, we simply ignore the implications of the preflop betting actions, and assume a uniform distribution over all possible hands for each player. Different postflop solutions were computed for initial pot sizes of two, four, six, and eight bets (corresponding to preflop sequences with zero, one, two, or three raises, but ignoring *which* player initially made each raise). In *PsOpti1*, the four postflop solutions are simply appended to the Selby preflop strategy (Figure 2). Although these simplifying assumptions are technically wrong, the resulting play is still surprisingly effective.

A better way to compose postflop models is to estimate the conditional probabilities, using the solution to a preflop model. With a tractable preflop model, we have a means of estimating an appropriate strategy at the root, and thereby determine the consequent probability distributions.

In *PsOpti2*, a 3-round preflop model was designed and solved. The resulting pseudo-optimal strategy for the preflop (which was significantly different from the Selby strategy) was used to determine the corresponding distribution of hands for each player in each context. This provided the necessary input parameters for each of the seven postflop betting sequences that carry over to the flop stage. Since each of these postflop models has been given (an approximation of) the perfect recall knowledge of the full game, they are fully compatible with each other, and are properly integrated under the umbrella of the preflop model (Figure 2). In theory, this should be equivalent to computing the much larger tree, but it is limited by the accuracy and appropriateness of the proposed preflop betting model.

4.4 Abstraction by bucketing

The most important method of abstraction for the computation of our pseudo-optimal strategies is called *bucketing*. This is an extension of the natural and intuitive concept that has been applied many times in previous research (e.g. [Sklansky and Malmuth, 1994] [Takusagawa, 2000] [Shi and Littman, 2001]). The set of all possible hands is partitioned into equivalence classes (also called *buckets* or *bins*). A *many-to-one mapping function* determines which hands will be grouped together. Ideally, the hands should be grouped according to *strategic similarity*, meaning that they can all be played in a similar manner without much loss in EV.

If every hand was played with a particular *pure strategy* (ie. only one of the available choices), then a perfect mapping function would group all hands that follow the same plan, and

³To see this, each decision node of the tree can be labeled with all the cards known to that player, and the full path that led to that node. Nodes with identical labels differ only in the hidden information, and are therefore in the same information set. Since the betting history is different for these sub-trees, none of the nodes are inter-dependent.

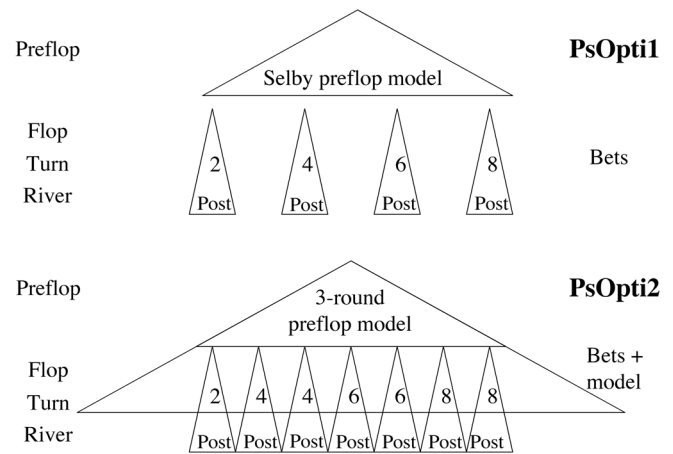


Figure 2: Composition of *PsOpti1* and *PsOpti2*.

17 equivalence classes for each player would be sufficient for each betting round. However, since a *mixed strategy* may be indicated for optimal play in some cases, we would like to group hands that have a similar probability distribution over action plans.

One obvious but rather crude bucketing function is to group all hands according to strength (ie. its *rank* with respect to all possible hands, or the probability of currently being in the lead). This can be improved by considering the roll-out of all future cards, giving an (unweighted) estimate of the chance of winning the hand.

However, this is only a one-dimensional view of hand types, in what can be considered to be an N -dimensional space of strategies, with a vast number of different ways to classify them. A superior practical method would be to project the set of all hands onto a two-dimensional space consisting of (roll-out) hand strength and hand potential (similar to the hand assessment used in *Poki*, [Billings *et al.*, 2002]). Clusters in the resulting scattergram suggest reasonable groups of hands to be treated similarly.

We eventually settled on a simple compromise. With n available buckets, we allocate $n - 1$ to *roll-out hand strength*. The number of hand types in each class is not uniform; the classes for the strongest hands are smaller than those for mediocre and weak hands, allowing for better discrimination of the smaller fractions of hands that should be raised or re-raised.

One special bucket is designated for hands that are low in strength but have *high potential*, such as good draws to a flush or straight. This plays an important role in identifying good hands to use for bluffing (known as *semi-bluffs* in [Sklansky and Malmuth, 1994]). Comparing postflop solutions that use six strength buckets to solutions with five strength plus one high-potential bucket, we see that most bluffs in the latter are taken from the special bucket, which is sometimes played in the same way as the strongest bucket. This confirmed our expectations that the high-potential bucket would improve the selection of hands for various betting tactics, and increase the overall EV.

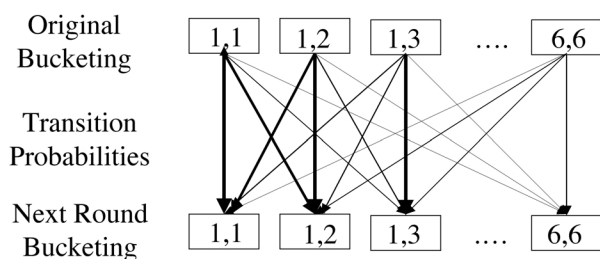


Figure 3: Transition probabilities (six buckets per player).

The number of buckets that can be used in conjunction with a 3-round model is very small, typically six or seven for each player (*ie.* 36 or 49 pairs of bucket assignments). Obviously this results in a very coarse-grained abstract game, but it may not be substantially different from the number of distinctions an average human player might make. Regardless, it is the best we can currently do given the computational constraints of this approach.

The final thing needed to sever the abstract game from the underlying real game tree are the *transition probabilities*. The chance node between the flop and turn represents a particular card being dealt from the remaining stock of 45 cards. In the abstract game, there are no cards, only buckets. The effect of the turn card in the abstract game is to dictate the probability of moving from one pair of buckets on the flop to any pair of buckets on the turn. Thus the collection of chance nodes in the game tree is represented by an $(n \times n)$ to $(n \times n)$ transition network as shown in Figure 3. For postflop models, this can be estimated by walking the entire tree, enumerating all transitions for a small number of characteristic flops. For preflop models, the full enumeration is more expensive (encompassing all $\{48 \text{ choose } 3\} = 17296$ possible flops), so it is estimated either by sampling, or by (parallel) enumeration of a truncated tree.

For a 3-round postflop model, we can comfortably solve abstract games with up to seven buckets for each player in each round. Changing the distribution of buckets, such as six for the flop, seven for the turn, and eight for the river, does not appear to significantly affect the quality of the solutions, better or worse.

The final linear programming solution produces a large table of mixed strategies (probabilities for fold, call, or raise) for every reachable scenario in the abstract game. To use this, the poker-playing program looks for the corresponding situation based on the same hand strength and potential measures, and randomly selects an action from the mixed strategy.

The large LP computations typically take less than a day (using CPLEX with the barrier method), and use up to two Gigabytes of RAM. Larger problems will exceed available memory, which is common for large LP systems. Certain LP techniques such as *constraint generation* could potentially extend the range of solvable instances considerably, but this would probably only allow the use of one or two additional buckets per player.

5 Experiments

5.1 Testing against computer players

A series of matches between computer programs was conducted, with the results shown in Table 1. Win rates are measured in small bets per hand (sb/h). Each match was run for at least 20,000 games (and over 100,000 games in some cases). The variance per game depends greatly on the styles of the two players involved, but is typically ± 6 sb. The standard deviation for each match outcome is not shown, but is normally less than ± 0.03 sb/h.

The “bot players” were:

PsOpti2, composed of a hand-crafted 3-round preflop model, providing conditional probability distributions to each of seven 3-round postflop models (Figure 2). All models in this prototype used six buckets per player per round.

PsOpti1, composed of four 3-round postflop models under the naive uniform distribution assumption, with 7 buckets per player per round. Selby’s optimal solution for *preflop Hold’em* is used to play the preflop ([Selby, 1999]).

PsOpti0, composed of a single 3-round postflop model, wrongly assuming uniform distributions and an initial pot size of two bets, with seven buckets per player per round. This program used an always-call policy for the preflop betting round.

Poki, the University of Alberta poker program. This older version of *Poki* was not designed to play the 2-player game, and can be defeated rather easily, but is a useful benchmark.

Anti-Poki, a rule-based program designed to beat *Poki* by exploiting its weaknesses and vulnerabilities in the 2-player game. Any specific counter-strategy can be even more vulnerable to adaptive players.

Aadapti, a relatively simple adaptive player, capable of slowly learning and exploiting persistent patterns in play.

Always Call, very weak benchmark strategy.

Always Raise, very weak benchmark strategy.

It is important to understand that a game-theoretic optimal player is, in principle, *not designed to win*. Its purpose is to *not lose*. An implicit assumption is that the opponent is also playing optimally, and nothing can be gained by observing the opponent for patterns or weaknesses.

In a simple game like RoShamBo (also known as Rock-Paper-Scissors), playing the optimal strategy ensures a break-even result, regardless of what the opponent does, and is therefore insufficient to defeat weak opponents, or to win a tournament ([Billings, 2000]). Poker is more complex, and in theory an optimal player *can* win, but only if the opponent makes *dominated errors*. Any time a player makes any choice that is part of a randomized mixed strategy of some game-theoretic optimal policy, that decision is not dominated. In other words, it is possible to play in a highly sub-optimal manner, but still break even against an optimal player, because those choices are not strictly dominated.

Since the pseudo-optimal strategies do no opponent modeling, there is no guarantee that they will be especially effective against very bad or highly predictable players. They must rely only on these fundamental strategic errors, and the margin of victory might be relatively modest as a result.

No.	Program	1	2	3	4	5	6	7	8
1	PsOpti1	X	+0.090	+0.091	+0.251	+0.156	+0.047	+0.546	+0.635
2	PsOpti2	-0.090	X	+0.069	+0.118	+0.054	+0.045	+0.505	+0.319
3	PsOpti0	-0.091	-0.069	X	+0.163	+0.135	+0.001	+0.418	+0.118
4	Aadapti	-0.251	-0.118	-0.163	X	+0.178	+0.550	+0.905	+2.615
5	Anti-Poki	-0.156	-0.054	-0.135	-0.178	X	+0.385	+0.143	+0.541
6	Poki	-0.047	-0.045	-0.001	-0.550	-0.385	X	+0.537	+2.285
7	Always Call	-0.546	-0.505	-0.418	-0.905	-0.143	-0.537	X	=0.000
8	Always Raise	-0.635	-0.319	-0.118	-2.615	-0.541	-2.285	=0.000	X

Table 1: Computer vs computer matches (sb/h).

The critical question is whether such errors are common in practice. There is no definitive answer to this question yet, but preliminary evidence suggests that dominated errors occur often enough to gain a measurable EV advantage over weaker players, but may not be very common in the play of very good players.

The first tests of the viability of pseudo-optimal solutions were done with *PsOpti0* playing *postflop Hold'em*, where both players agree to simply call in the preflop (thereby matching the exact pre-conditions for the postflop solution). In those preliminary tests, a poker master (the first author) played more than 2000 hands, and was unable to defeat the pseudo-optimal strategy. In contrast, *Poki* had been beaten consistently at a rate of over 0.8 sb/h (which is more than would be lost by simply folding every hand).

Using the same no-bet preflop policy, *PsOpti0* was able to defeat *Poki* at a rate of +0.144 sb/h (compared to +0.001 sb/h for the full game including preflop), and defeated *Aadapti* at +0.410 sb/h (compared to +0.163 sb/h for the full game).

All of the pseudo-optimal players play substantially better than any previously existing computer programs. Even *PsOpti0*, which is not designed to play the full game, earns enough from the postflop betting rounds to offset the EV losses from the preflop round (where it never raises good hands, nor folds bad ones).

It is suspicious that *PsOpti1* outperformed *PsOpti2*, which in principle should be a better approximation. Subsequent analysis of the play of *PsOpti2* revealed some programming errors, and also suggested that the bucket assignments for the preflop model were flawed. This may have resulted in an inaccurate pseudo-optimal preflop strategy, and consequent imbalances in the prior distributions used as input for the postflop models. We expect that this will be rectified in future versions, and that the *PsOpti2* design will surpass *PsOpti1* in performance.

5.2 Testing against human players

While these results are encouraging, none of the non-pseudo-optimal computer opponents are better than intermediate strength at 2-player Texas Hold'em. Therefore, matches were conducted against human opponents.

More than 100 participants volunteered to play against the pseudo-optimal players on our public web applet (www.cs.ualberta.ca/~games/poker/), including many experienced players, a few masters, and one world-class player. The programs provided some fun opposition, and ended up with a winning record overall. The results are summa-

Player	Hands	Posn 1	Posn 2	sb/h
Master-1 early	1147	-0.324	+0.360	+0.017
Master-1 late	2880	-0.054	+0.396	+0.170
Experienced-1	803	+0.175	+0.002	+0.088
Experienced-2	1001	-0.166	-0.168	-0.167
Experienced-3	1378	+0.119	-0.016	+0.052
Experienced-4	1086	+0.042	-0.039	+0.002
Intermediate-1	2448	+0.031	+0.203	+0.117
Novice-1	1277	-0.159	-0.154	-0.156
All Opponents	15125			-0.015

Table 2: Human vs *PsOpti2* matches.

Player	Hands	Posn 1	Posn 2	sb/h
thecount	7030	-0.006	+0.103	+0.048
Master-1	2872	+0.141	+0.314	+0.228
Master-2	569	-0.007	+0.035	+0.014
Master-3	425	+0.047	+0.373	+0.209
Experienced-1	4078	-0.058	+0.164	+0.053
Experienced-2	511	+0.152	+0.369	+0.260
Experienced-3	2303	-0.252	+0.128	-0.062
Experienced-5	610	-0.250	-0.229	-0.239
Intermediate-1	16288	-0.145	+0.048	-0.049
Intermediate-2	478	-0.182	+0.402	+0.110
Novice-1	5045	-0.222	-0.010	-0.116
Novice-2	485	-0.255	-0.139	-0.197
Novice-3	1017	-0.369	-0.051	-0.210
Novice-4	583	-0.053	-0.384	-0.219
Novice-5	425	-0.571	-0.296	-0.433
All Opponents	46479			-0.057

Table 3: Human vs *PsOpti1* matches.

ized in Table 2 and Table 3. (Master-1 is the first author, Experienced-1 is the third author).

In most cases, the relatively short length of the match leaves a high degree of uncertainty in the outcome, limiting how much can be safely concluded. Nevertheless, some players did appear to have a definite edge, while others were clearly losing.

A number of interesting observations were made over the course of these games. It was obvious that most people had a lot of difficulty learning and adjusting to the computer's style of play. In poker, knowing the basic approach of the opponent is essential, since it will dictate how to properly handle many situations that arise. Some players wrongly attributed intelligence where none was present. After losing a 1000-game match, one experienced player commented "the bot has

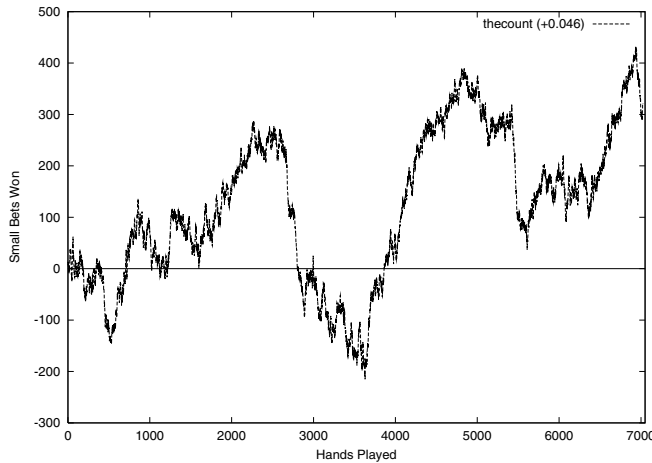


Figure 4: Progress of the “thecount” vs *PsOpt1*

me figured out now”, indicating that its opponent model was accurate, when in fact the pseudo-optimal player is oblivious and does no modeling at all.

It was also evident that these programs do considerably better in practice than might be expected, due to the emotional frailty of their human opponents. Many players commented that playing against the pseudo-optimal opponent was an exasperating experience. The bot routinely makes unconventional plays that confuse and confound humans. Invariably, some of these “bizarre” plays happen to coincide with a lucky escape, and several of these *bad beats* in quick succession will often cause strong emotional reactions (sometimes referred to as “going on tilt”). The level of play generally goes down sharply in these circumstances.

This suggests that a perfect game-theoretic optimal poker player could perhaps beat even the best humans in the long run, because any EV lost in moments of weakness would never be regained. However, the win rate for such a program could still be quite small, giving it only a slight advantage. Thus it would be unable to exert its superiority convincingly over the short term, such as the few hundred hands of one session, or over the course of a world championship tournament. Since even the best human players are known to have biases and weaknesses, opponent modeling will almost certainly be necessary to produce a program that surpasses all human players.

5.3 Testing against a world-class player

The elite poker expert was **Gautam Rao**, who is known as “thecount” or “CountDracula” in the world of popular online poker rooms. Mr. Rao is the #1 all-time winner in the history of the oldest online game, by an enormous margin over all other players, both in total earnings and in dollar-per-hand rate. His particular specialty is in *short-handed games* with five or fewer players. He is recognized as one of the best players in the world in these games, and is also exceptional at 2-player Hold’em. Like many top-flight players, he has a dynamic ultra-aggressive style.

Mr. Rao agreed to play an exhibition match against

PsOpt1, playing more than 7000 hands over the course of several days. The graph in Figure 4 shows the progression of the match.

The pseudo-optimal player started with some good fortune, but lost at a rate of about -0.2 sb/h over the next 2000 hands. Then there was a sudden reversal, following a series of fortuitous outcomes for the program. Although “thecount” is renowned for his mental toughness, an uncommon run of bad luck can be very frustrating even for the most experienced players. Mr. Rao believes he played below his best level during that stage, which contributed to a dramatic drop where he lost 300 sb in less than 400 hands. Mr. Rao resumed play the following day, but was unable to recover the losses, slipping further to -200 sb after 3700 hands. At this point he stopped play and did a careful reassessment.

It was clear that his normal style for maximizing income against typical human opponents was not effective against the pseudo-optimal player. Whereas human players would normally succumb to a lot of pressure from aggressive betting, the bot was willing to call all the way to the showdown with as little as a Jack or Queen high card. That kind of play would be folly against most opponents, but is appropriate against an extremely aggressive opponent. Most human players fail to make the necessary adjustment under these atypical conditions, but the program has no sense of fear.

Mr. Rao changed his approach to be less aggressive, with immediate rewards, as shown by the +600 sb increase over the next 1100 hands (some of which he credited to a good run of cards). Mr. Rao was able to utilize his knowledge that the computer player did not do any opponent modeling. Knowing this allows a human player to systematically probe for weaknesses, without any fear of being punished for playing in a methodical and highly predictable manner, since an oblivious opponent does not exploit those patterns and biases.

Although he enjoyed much more success in the match from that point forward, there were still some “adventures”, such as the sharp decline at 5400 hands. Poker is a game of very high variance, especially between two opponents with sharp styles, as can be seen by the dramatic swings over the course of this match. Although 7000 games may seem like a lot, Mr. Rao’s victory in this match was still not statistically conclusive.

We now believe that a human poker master can eventually gain a sizable advantage over these pseudo-optimal prototypes (perhaps +0.20 sb/h or more is sustainable). However, it requires a good understanding of the design of the program and its resulting weaknesses. That knowledge is difficult to learn during normal play, due to the good information hiding provided by an appropriate mixture of plans and tactics. This “cloud of confusion” is a natural barrier to opponent learning. It would be even more difficult to learn against an adaptive program with good opponent modeling, since any methodical testing by the human would be easily exploited. This is in stark contrast to typical human opponents, who can often be accurately modeled after only a small number of hands.

6 Conclusions and Future Work

The pseudo-optimal players presented in this paper are the first complete approximations of a game-theoretic optimal strategy for a full-scale variation of real poker.

Several abstraction techniques were explored, resulting in the reasonably accurate representation of a large imperfect information game tree having $O(10^{18})$ nodes with a small collection of models of size $O(10^7)$. Despite these massive reductions and simplifications, the resulting programs play respectably. For the first time ever, computer programs are not completely outclassed by strong human opposition in the game of 2-player Texas Hold'em.

Useful abstractions included betting tree reductions, truncation of betting rounds combined with EV leaf nodes, and bypassing betting rounds. A 3-round model anchored at the root provided a pseudo-optimal strategy for the preflop round, which in turn provided the proper contextual information needed to determine conditional probabilities for post-flop models. The most powerful abstractions for reducing the problem size were based on bucketing, a method for partitioning all possible holdings according to strategic similarity. Although these methods exploit the particular structure of the Texas Hold'em game tree, the principles are general enough to be applied to a wide variety of imperfect information domains.

Many refinements and improvements will be made to the basic techniques in the coming months. Further testing will also continue, since accurate assessment in a high variance domain is always difficult.

The next stage of the research will be to apply these techniques to obtain approximations of Nash equilibria for N -player Texas Hold'em. This promises to be a challenging extension, since multi-player games have many properties that do not exist in the 2-player game.

Finally, having reasonable approximations of optimal strategies does not lessen the importance of good opponent modeling. Learning against an adaptive adversary in a stochastic game is a challenging problem, and there will be many ideas to explore in combining the two different forms of information. That will likely be the key difference between a program that can *compete* with the best, and a program that *surpasses* all human players.

Quoting "thecount":

"You have a very strong program. Once you add opponent modeling to it, it will kill everyone."

Acknowledgments

The authors would like to thank Gautam Rao, Sridhar Mutyala, and the other poker players for donating their valuable time. We also wish to thank Daphne Koller, Michael Littman, Matthew Ginsberg, Rich Sutton, David McAllester, Mason Malmuth, and David Sklansky for their valuable insights in past discussions.

This research was supported in part by grants from the Natural Sciences and Engineering Research Council of Canada (NSERC), the Alberta Informatics Circle of Research Excellence (iCORE), and an Izaak Walton Killam Memorial post-graduate scholarship.

References

- [Billings *et al.*, 2002] D. Billings, A. Davidson, J. Schaeffer, and D. Szafron. The challenge of poker. *Artificial Intelligence*, 134(1–2):201–240, 2002.
- [Billings, 2000] D. Billings. The first international roshambo programming competition. *International Computer Games Association Journal*, 23(1):3–8, 42–50, 2000.
- [Koller and Megiddo, 1992] D. Koller and N. Megiddo. The complexity of two-person zero-sum games in extensive form. *Games and Economic Beh.*, 4(4):528–552, 1992.
- [Koller and Pfeffer, 1997] D. Koller and A. Pfeffer. Representations and solutions for game-theoretic problems. *Artificial Intelligence*, pages 167–215, 1997.
- [Koller *et al.*, 1994] D. Koller, N. Megiddo, and B. von Stengel. Fast algorithms for finding randomized strategies in game trees. *STOC*, pages 750–759, 1994.
- [Kuhn, 1950] H. W. Kuhn. A simplified two-person poker. *Contributions to the Theory of Games*, 1:97–103, 1950.
- [Nash, 1950] J. Nash. Equilibrium points in n -person games. *National Academy of Sciences*, 36:48–49, 1950.
- [Sakaguchi and Sakai, 1992] M. Sakaguchi and S. Sakai. Solutions of some three-person stud and draw poker. *Mathematics Japonica*, pages 1147–1160, 1992.
- [Selby, 1999] A. Selby. Optimal heads-up preflop poker. 1999. www.archduke.demon.co.uk/simplex.
- [Shi and Littman, 2001] J. Shi and M. Littman. Abstraction models for game theoretic poker. In *Computers and Games*, pages 333–345. Springer-Verlag, 2001.
- [Sklansky and Malmuth, 1994] D. Sklansky and M. Malmuth. *Texas Hold'em for the Advanced Player*. Two Plus Two Publishing, 2nd edition, 1994.
- [Takusagawa, 2000] K. Takusagawa. Nash equilibrium of Texas Hold'em poker, 2000. Undergraduate thesis, Computer Science, Stanford University.
- [von Neumann and Morgenstern, 1944] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

Last-Branch and Speculative Pruning Algorithms for Maxⁿ

Nathan Sturtevant

UCLA, Computer Science Department
Los Angeles, CA 90024
nathanst@cs.ucla.edu

Abstract

Previous work in pruning algorithms for maxⁿ multi-player game trees has produced shallow pruning and alpha-beta branch-and-bound pruning. The effectiveness of these algorithms is dependant as much on the range of terminal values found in the game tree as on the ordering of nodes. We introduce last-branch and speculative pruning techniques which can prune any constant-sum multi-player game tree. Their effectiveness depends only on node-ordering within the game tree. As b grows large, these algorithms will, in the best case, reduce the branching factor of a n -player game from b to $b^{(n-1)/n}$. In Chinese Checkers these methods reduce average expansions at depth 6 from 1.2 million to 100k nodes, and in Hearts and Spades they increase the average search depth by 1-3 ply.

1 Introduction

While the minimax algorithm with alpha-beta pruning [Knuth and Moore, 1975] has dominated the study of 2-player games, a clearly dominant algorithm has not emerged in the study of n -player games.

The standard algorithm for n -player games is maxⁿ [Luckhardt and Irani, 1986]. Only some maxⁿ game trees can be pruned using established techniques such as shallow pruning [Korf, 1991] and alpha-beta branch-and-bound pruning [Sturtevant and Korf, 2000]. This is because the amount of pruning possible under these algorithms is dependant on both the node ordering and the range of terminal values found in the game. Common heuristics for games such as Hearts and Chinese Checkers do not have appropriate terminal values for pruning. In two-player games, however, pruning is only dependant on the node ordering in the tree.

We present here last-branch and speculative pruning techniques which can be used to prune any n -player constant-sum maxⁿ game tree. The effectiveness of these algorithms depends only on node ordering within the game tree. Last-branch pruning is similar to a directional algorithm [Pearl,

1984], as it examines the successors of each node left-to-right without returning to previously searched nodes. It does require that you know when you are searching the last branch of a node. Speculative pruning, however, is not a directional algorithm as it may re-search some branches of the tree. Last-branch pruning is a special case of speculative pruning.

The remainder of the paper is as follows. In Section 2 we present a brief overview of sample multi-player domains. In Section 3 we cover previous techniques for pruning maxⁿ trees. Section 4 covers our new techniques, with results from various domains in Section 5.

2 Multi-Player Domains

A multi-player game is one in which there are 3 or more players or teams competing against each other. Many 4-player games are designed for 2 teams, making them two-player games from a theoretical standpoint. We highlight a few of the many interesting multi-player games here.

2.1 Chinese Checkers

One of the best known multi-player board games is Chinese Checkers. This game is played on a board in the shape of a 6-pointed star, with players' pieces starting on opposite sides of the board. The goal is to move your pieces across the board faster than your opponent(s) can. Chinese Checkers can be played with 2-6 players, and is not played in teams.

2.2 Card Games

We use two games here as examples, Spades and Hearts. Spades is usually played with 4 players, but has a 3-player variant. The goal of Spades is to maximize your score. Hearts is usually played with 4 players, but can be played with 3 or more. The goal of Hearts is to minimize your score.

Many card games, including these, are trick-based. Because a trick cannot be given up after being taken, the tricks taken during the game provide a monotonically increasing lower-bound on the final score of the game. In the case of Spades, players get 1 point per trick taken, while in Hearts each player gets 1 point for every heart in the tricks they take, as well as 13 points for the queen of spades. A complete game is made up of multiple hands, where the goal is to minimize or maximize your points over all the hands.

Both Spades and Hearts are imperfect-information

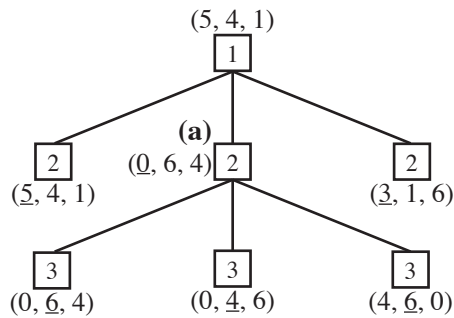


Figure 1. A 3-player $\max^n$ tree fragment.

games, while the descriptions and trees in this paper assume games have perfect information. In card games, however, we often use perfect-information methods to search imperfect-information games through Monte-Carlo simulation. This involves generating sets of hands that are similar to the hands we expect our opponents to have. We then use traditional perfect-information methods to search these hands, combining and analyzing the results to make the next play. These methods have been successfully used in the Bridge program GIB. [Ginsberg, 2001]

3 Pruning Algorithms for $\max^n$

The $\max^n$ algorithm can be used to play games with any number of players. For two-player games, $\max^n$ reduces to minimax. In a $\max^n$ tree with n players, the leaves of the tree are n -tuples, where the i th element in the tuple is the i th player's score. At the interior nodes in the game tree, the $\max^n$ value of a node where player i is to move is the child of that node for which the i th component is maximum. This can be seen in Figure 1.

In this tree fragment there are three players. The player to move is labeled inside each node. At node (a), Player 2 is to move. Player 2 can get a score of 6 by moving to the left or right and a score of 4 by moving to the middle node. We break ties to the left, so Player 2 will choose the left branch, and the $\max^n$ value of node (a) is (0, 6, 4). Player 1 acts similarly at the root selecting the left branch, because the first component of the $\max^n$ value there, 5, is greater his score from the middle branch, 0, and his score at the right branch, 3.

3.1 Shallow Pruning

Shallow pruning refers to cases where a bound on a node is used to prune branches from the child of that node. The minimum requirements for pruning a $\max^n$ tree with shallow pruning are a lower bound on each players' score and an upper bound, $\maxsum$, on the sum of all players' scores. We demonstrate this in Figure 2. In this figure, all scores have a lower bound of 0, and $\maxsum$ is 10.

Player 1 searches the left branch of the root in a depth-first manner, getting the $\max^n$ value (5, 4, 1). Thus, we know that Player 1 will get at least 5 points at the root. Since there are only 10 points available, we know the other players will get less than 5 points at this node.

At node (a), Player 2 searches the left branch to get a

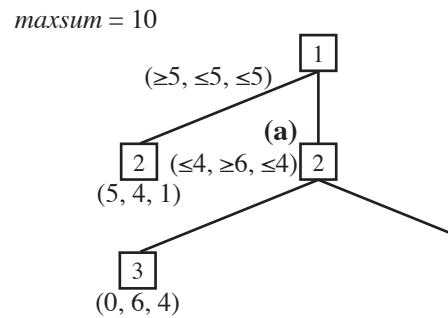


Figure 2. Shallow pruning in a 3-player $\max^n$ tree.

score of 6. Because $\maxsum$ is 10 we know that Player 1's score will never exceed 4 at node (a), so Player 1 will never choose to move towards (a) at the root, and the remaining children of (a) can be pruned.

For shallow pruning to actually occur in a maximization game, each player must have a score that ranges between 0 and $\maxsum$. [Sturtevant and Korf, 2000] There are similar bounds for minimization games. In the average case, no asymptotic gain can be expected from shallow pruning. [Korf, 1991] The basic problem with shallow pruning is that it works by comparing the scores of only 2 out of n players in the game, and it is unlikely that 2 players will have their scores sum to $\maxsum$. This contrasts with the large gains available from using alpha-beta pruning with 2-player minimax.

3.2 Branch and Bound Pruning

If a monotonic heuristic is present in a game, it can also be used to prune a $\max^n$ tree. The full details of how this occurs is contained in [Sturtevant and Korf, 2000]. An example of a monotonic heuristic is the number of tricks taken in Spades. Once a trick has been taken, it cannot be lost. This guarantee can provide a lower bound on a player's score, and an upper bound on one's opponents scores, which can be used in branch-and-bound pruning to prune the game tree. Alpha-beta branch-and-bound pruning combines the actual scores of the 2 players compared in shallow pruning with the heuristic information from the other $n-2$ players to prune the tree.

3.3 Pruning in Practice

These algorithms have mixed performances in practice. There are no obviously useful monotonic heuristics for Chinese Checkers, and most useful cutoff evaluation functions for the game do not meet the requirements for shallow pruning. Thus, in practice it is not possible to use any of these techniques to prune a Chinese Checkers game tree.

If we use the number of points taken in the game tree so far as the evaluation function, we will be able to use shallow pruning in Spades and branch-and-bound pruning in both Spades and Hearts. This is fine if you can search the entire game tree, but we are currently unable to do that. If we add a predictive component to the evaluation function, these pruning techniques are much less effective. Thus a trade-off has to be made between the quality of the cutoff evaluation function

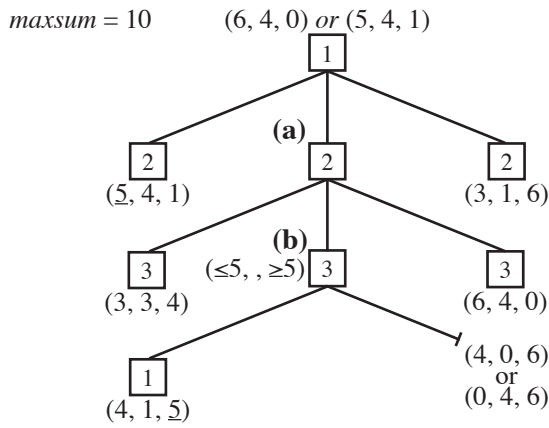


Figure 3: The failure of deep pruning.

and the potential amount of pruning in the game tree.

3.4 Deep Pruning

Deep pruning refers to when the bound at a node is used to prune a grandchild or lower descendant of that node. [Korf, 1991] shows that, in the general case, deep pruning can incorrectly affect the $\max^n$ value of the game tree. We demonstrate this in Figure 3. After searching the left branch of node (b), Player 3 is guaranteed 5 points, and Player 1 is guaranteed to get no more than 5 points at node (b). So, we can conclude that the $\max^n$ value of node (b) will never be the $\max^n$ value of the game tree, because Player 1 is already guaranteed a score of 5 at the root, and he can do no better than that at node (b). It is still possible, however, that the $\max^n$ value at (b) can affect the final $\max^n$ value of the tree.

For instance, if the actual $\max^n$ value of (b) is (4, 0, 6), Player 2 will prefer the move (6, 4, 0) at node (a), and this will be the $\max^n$ value of the game tree. But, if the $\max^n$ value of (b) is (0, 4, 6), Player 2 will prefer this value, and so Player 1 will choose (5, 4, 1) to be the $\max^n$ value of the game tree. Thus, in general deep pruning is not valid.

4 New Pruning Techniques

We now present our new algorithms. They have the same minimum requirements to prune as shallow pruning, namely a lower bound on each player's score and an upper bound on the sum of scores. This is slightly weaker than requiring a game to be constant-sum, meaning that the sum of all scores at every node is constant, but in either case we can usually make adjustments to the evaluation function in a game to be able to prune effectively.

4.1 Requirements for Pruning

Last-branch and speculative pruning both examine portions of $\max^n$ game trees to find nodes whose $\max^n$ value can never be the $\max^n$ value of the tree. They take different approaches, however, when they prune these nodes.

Returning to Figure 3, we know that Player 1 will never get a better value than 5 at node (b). But, to prune at (b) correctly, we must show that Player 1 cannot get a better $\max^n$ value at the root from either node (b) or node (a), as the

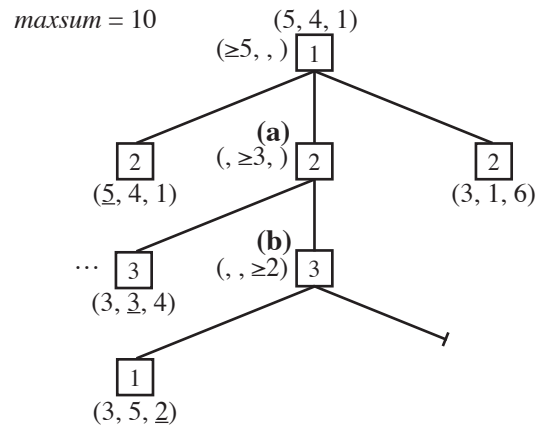


Figure 4: Combining $\max^n$ scores to limit value propagation.

values at (a) may interact with unseen values at (b) to affect Player 2's, and thus Player 1's move. In this case, deep pruning failed because the value at the right child of (a) was better for Player 2 than a previous child of (a). If the children of (a) were ordered optimally for Player 2, or if there was no right child at (a), the deep prune could not have affected the $\max^n$ value of the tree.

While shallow pruning only considers 2 players' bounds when pruning, we can actually use n players' bounds in a n -player game. Since each player has already searched one or more branches when we reach node (b) in Figure 4, we have a lower bound on each player's score. In this case, Player 1 has a lower bound of 5 from the left branch of the root, Player 2 has a bound of 3 from the left branch of (a), and Player 3 has a bound of 2 from the left branch of (b). The sum of these bounds is 10, which is greater than or equal to $\maxsum$. We can thus show that any unseen value at (b) cannot be the $\max^n$ value of the tree.

Lemma 1: If, in a $\max^n$ game tree, the sum of lower bounds for a consecutive sequence of unique players meets or exceeds $\maxsum$, the $\max^n$ value of any child of the last player in the sequence cannot be the $\max^n$ value of the game tree.

Proof: We provide a proof by contradiction. Figure 5 shows a

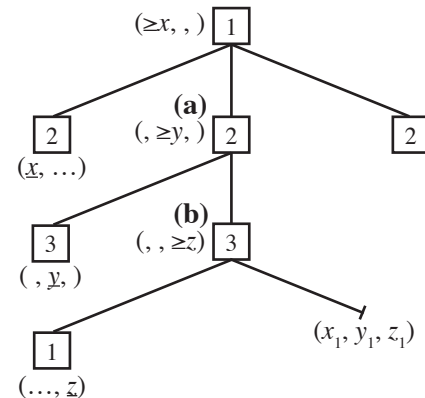


Figure 5: Combining scores to limit value propagation in general.

b	$b^{2/3}$	asymptotic b
2	1.5874	1.8393
3	2.0801	2.4675
4	2.5198	3.0000
5	2.9240	3.4755
10	4.6416	5.4191
1000	100.00	103.61

Table 8: Branching factor gains by speculative $\max^n$ in a 3-player game.

change the $\max^n$ value at the root of the tree, we will have to search node (b) again using new bounds. This occurs when Player 2's nodes are ordered suboptimally. With an optimal node ordering we will never have to re-search a subtree.

In general, we have to re-search pruned branches if, on a mid-level branch, we find a new value for which both that mid-level player and his parent have better scores. As with last-branch pruning, we can only prune when Player 1 prefers his current move over Player 2's partial $\max^n$ value. If we wish to preserve the order of tie-breaking in the tree, we must also retain some extra information about the order of nodes expanded. Nodes that can be pruned by last-branch pruning will be always be pruned by speculative pruning. Pseudo-code for speculative pruning can be found in Figure 7.

As can be seen, it is reasonably simple to perform speculative pruning in practice. In the 3-player implementation, the specmax^n function takes 3 arguments, the current node, the best score at the parent node, and the best score at the grandparent node.

At the line marked $\dagger$ we are checking to see if our parent can get a better score from the partial $\max^n$ value of this node than from one of his previously searched nodes. If this is the case, we cannot use the parent's bounds to help prune the children of the current node.

When a node is speculatively pruned, the specmax^n function returns NIL. All nodes that have speculatively pruned children are added to a list of pruned nodes, and if a child is found with a $\max^n$ value that better for both the current node and the parent node, then the speculatively pruned nodes will have to be re-searched. This pseudo-code assumes that players always alternate plays in the tree, as in Chinese Checkers. In card games, where this may not be the case, additional checks are needed.

For 3 players, the best-case analysis of speculative pruning can be formulated as a recurrence, the solution of which is the equation $x^3 - x^2 - (b-1)x - (b-1)^2 = 0$. A complete derivation of the recurrence can be found in [Sturtevant, 2003]. Solving this equation for x gives the asymptotic branching factor, which, as b grows large, is $b^{2/3}$. For a general n -player game, the equation will be $x^n - x^{n-1} - (b-1)^1 x^{n-2} - (b-1)^2 x^{n-3} - \dots - (b-1)^{n-1} = 0$. As b grows large, this approaches $b^{n-1/n}$. We give sample values for b , given an optimal ordering of nodes in a 3-player game, in Table 8. The first column contains sample values for b , the second column contains $b^{2/3}$, and the third column contains the actual optimal asymptotic value of b as calculated from these equations.

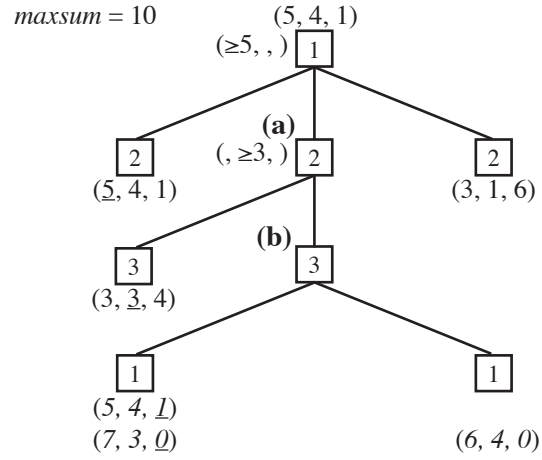


Figure 9. Discrete cut-off evaluations

4.4 Discrete Cutoff Evaluations

It is possible to use tighter bounds for pruning when the evaluation function has discrete as opposed to continuous values. This draws from the proof of lemma 1. In this proof we see that, for a value to affect the $\max^n$ value of the tree, $x_1 > x$, $y_1 > y$, and $z_1 > z$. Suppose the minimum delta of a player's score is μ . Since all players in the game must do strictly better than their previous value we can combine this into our bounds.

We demonstrate this in Figure 9. At the root of the tree, Player 1 is guaranteed a score of 5, and at node (a) Player 2 is guaranteed 3 points. In this example $\mu = 1$, so for these players both to prefer to move towards (b) they must get at least 6 and 4 points respectively. Because maxsum is 10, we know if Player 3 gets more than 0 points, Players 1 and 2 can't both get better than their current best scores. So, instead of pruning when Player 3 gets $10 - 5 - 3 = 2$ points, we can prune when Player 3 gets 1 point.

We can then use our tie-breaking rule to improve this. Because ties are broken to the left, we can prune if Player 3 gets 0 points at the left branch of (b) and Player 1 and 2 don't get 6 and 4 points respectively. If, for instance, the score is (7, 3, 0), Player 2 won't choose this value over the left branch of (a). In addition, Player 3 will only choose a better value than 0 from the unexpanded children of (b), which will meet our earlier conditions for pruning.

It follows from this that we can always prune if $\sum \text{scores} \geq \text{maxsum} - \mu(n-2)$, where $\sum \text{scores}$ are the current bounds for the players in the tree. Additionally, we can prune if $\sum \text{scores} \geq \text{maxsum} - \mu(n-1)$ and if on the first branch of the node being pruned the other $n-1$ players don't all have better scores than their current best bound.

5 Experimental Results

As speculative pruning includes last-branch pruning as a special case, we only report our experiments with speculative $\max^n$. In each experiment, re-expansions by speculative $\max^n$ are counted as part of the total number of node expansions, and node re-expansions in speculative pruning never out-

Chinese Checkers expansions at depth 6	
Plain Max ⁿ	1.2 million
Speculative Max ⁿ	100k

Table 10: Average expansions by maxⁿ in Chinese Checkers.

weighed the nodes savings from the additional pruning.

Also, while our sample search trees show cutoff evaluation functions that are seemingly independent of the other players, the actual cutoff evaluation functions used in these experiments are based on the scores of all players in the game. So, instead of just trying to maximize their own score, players are actually trying to maximize the difference between their score and their opponent's scores.

Finally, all our experiments on card games were run on the 3-player perfect-information variations of each game.

5.1 Chinese Checkers

In the past, maxⁿ search of Chinese Checkers trees has been limited to brute-force search, as the bounds in the game are not appropriate for shallow or branch-and-bound pruning. In our experiments we ordered moves according to which ones move a piece farthest across the board. Because the branching factor is very high (over 100 in the mid-game), we only considered the top 10 moves in our ordering at each node.

We played speculative maxⁿ in thirty 3-player games; each game is about 50 moves long. We then measured the average number of nodes expanded at depth 6 by speculative maxⁿ. Results are found in Table 10. While it takes 1.2 million nodes to complete iterative searches to this depth with no pruning, speculative maxⁿ expands an average of 100k nodes, while an optimal ordering of nodes would expand 84,927 nodes for this branching factor and depth. In some cases this optimal ordering is achieved in practice.

5.2 Hearts

It is more difficult to measure the efficiency of speculative maxⁿ in card games. This is because node expansions are highly dependant on the cards in your hand. Additionally, better play, resulting from deeper search, often results in more nodes being expanded throughout a game, as good plays early in the game force more computation later in the game. So, we measured our results in a slightly different manner in Hearts.

We played 300 hands of Hearts with a 500k node search limit. Hands were searched iteratively deeper until the node limit was reached, and re-expansions in speculative maxⁿ

counted towards the node limit. We then measured the average search depth by maxⁿ and speculative maxⁿ over all games. The results are found in Table 11. Averaging the search depth over all moves is misleading, because at many points in a hand the search depth is limited by the number of cards in your hand. Instead, we averaged the search depth for points in the tree where the search wasn't limited by the number of cards in the hand, and we also found the average point at which a hand could be searched to completion.

Speculative maxⁿ can search a hand to completion when, on average, there are 22.1 cards remaining in a hand, while maxⁿ can only do it when 20.9 cards remained. In the case where search isn't depth limited, speculative maxⁿ can search an average of 12.1 moves ahead, while maxⁿ can only search 11.1 moves deep.

There are three reasons we don't see more spectacular depth gains. First, in card games the order of player moves in the tree is less uniform, which will lessen the amount of pruning available. Second, we know that the ordering heuristic we used in Hearts can be improved, as it had to re-search many more times than in Chinese Checkers and Spades. Finally, the asymptotic branching factor of Hearts is much lower than Chinese Checkers, so the actual gains are smaller.

5.3 Spades

In Spades, like Hearts, we played 300 games with a 500k node search limit, and then measured the average search depth over all games. The results are also found in Table 11. While speculative maxⁿ could search a hand to completion when it contained, on average, 24.3 cards, maxⁿ could only do it when 21.7 cards remained. When the search was not depth limited, speculative maxⁿ was only able to search depths 15.4 while maxⁿ could only search to depth 11.1.

5.4 Performance against Paranoid Algorithm

The paranoid algorithm [Sturtevant and Korf, 2000] is another algorithm for playing multi-player games. It reduces a game to a 2-player game by assuming one's opponents have formed a coalition. This represents a different decision rule than standard maxⁿ. The trade-off for this less plausible decision rule is a gain in search depth. All techniques from 2-player game research can be used under the paranoid algorithm, and alpha-beta pruning in a n -player paranoid game tree will, in the best case, reduce the number of nodes expanded in a game tree with n players from b^d to $b^{d-(n-1)/n}$.

Although paranoid and speculative maxⁿ have the same asymptotic growth, paranoid will produce smaller trees in practice because it can prune more than n levels away from

	Hearts average search depth		Spades average search depth	
	complete tree	partial tree	complete tree	partial tree
Plain Max ⁿ	20.9	11.1	21.7	11.1
Speculative Max ⁿ	22.1	12.1	24.3	15.4
Paranoid	25.8	17.1	33.5	22.4

Table 11: Average search depth with 500k node expansions in Hearts and Spades.

	Hearts	Spades
Plain Max ⁿ / Paranoid	8.50 / 8.50	5.55 / 5.78
Spec. Max ⁿ / Paranoid	8.01 / 8.99	5.72 / 5.61

Table 12: Average score in Spades and Hearts.

where a bound originates.

A comparison between paranoid and maxⁿ can be found in [Sturtevant, 2002]. Given the gains from speculative maxⁿ, it is worth making a comparison against paranoid.

It has been shown experimentally [Sturtevant, 2002] that the paranoid algorithm outperforms standard maxⁿ at fixed depths in Chinese Checkers. Since adding speculative pruning doesn't change maxⁿ's decision rule, we don't see a gain in maxⁿ's performance against paranoid with speculative maxⁿ. But, it may now be worth considering strategies that are more mixed between standard maxⁿ and the paranoid strategy, as paranoid's advantage in search depth has been greatly lessened.

To measure the differences between paranoid and maxⁿ in card games we followed the same strategy in Hearts and Spades. In both of these games we played 50 hands with a 500k node search limit. Each of these hands were played multiple times to account for player positions on the table. We measured the average search depth for paranoid over these games, and we also compared the average score for maxⁿ with and without speculative pruning. Both algorithms expand nodes at the same rate, so there is no significant difference in CPU time between the two algorithms.

In Table 11 we see that paranoid was able to search deeper than speculative maxⁿ in both Hearts and Spades. In Spades, paranoid can do highly efficient zero-window searches [Pearl, 1980], allowing it to search entire hands when they contain 33.5 cards on average, over 9 cards better than speculative maxⁿ. In Hearts, paranoid can search 3.7 cards deeper than speculative maxⁿ.

But, despite paranoid's advantage in search depth, these results do not translate into quality of play. We see this in Table 12. Each entry in the table contains the listed algorithm's average score followed by paranoid's average score. In Hearts, paranoid and maxⁿ had virtually the same score after all games. But, with the additive pruning and search depth in speculative maxⁿ, it was able to beat paranoid by almost 1 point per hand. (Lower scores are better.) In Spades, where maxⁿ did slightly worse than paranoid, speculative maxⁿ was able to do better than paranoid, despite a much shallower search. (Higher scores are better.) These results show that given speculative pruning, maxⁿ has the potential to be the best decision rule for card games.

6 Conclusions and Future Work

We have seen here that last-branch and speculative pruning combine to provide large theoretical and practical gains for searching maxⁿ trees. This is the first pruning technique to be developed for maxⁿ that is effective on a wide range of games. While other pruning techniques have required very specific cutoff evaluation functions in order to prune, last-

branch and speculative maxⁿ pruning only require a game to have a lower bound on each player's score and an upper bound on the sum of all scores.

Also, while the paranoid algorithm must make assumptions about the strategies its opponents use in order to prune, last-branch and speculative pruning make no such assumptions, but can still be adapted to account for opponents strategies. Thus, these algorithms are quite promising for use in multi-player games.

Speculative maxⁿ pruning only prunes over n levels in the game tree. It is possible to extend this idea to prune even deeper into the tree. However, it is unclear if the cost of book-keeping and re-search for these cases offsets the potential gains. This is an area of our ongoing research.

Acknowledgments

We would like to thank Rich Korf and Alex Fukunaga for their comments and suggestions on these techniques. This work was supported in part by NASA and JPL under contract No. 1229784 and by NSF contract EIA-0113313.

References

- [Ginsberg, 2001] GIB: Imperfect Information in a Computationally Challenging Game, *Journal of Artificial Intelligence Research*, 14, 2001, 303-358.
- [Knuth and Moore, 1975] An Analysis of Alpha-Beta Pruning, *Artificial Intelligence*, vol. 6 no. 4, 1975, 293-326.
- [Korf 1991] Multiplayer Alpha-Beta Pruning, *Artificial Intelligence*, vol. 48 no. 1, 1991, 99-111.
- [Luckhardt and Irani, 1986] An algorithmic solution of N-person games, *Proceedings AAAI-86*, Philadelphia, PA, 158-162.
- [Pearl, 1980] Asymptotic properties of minimax trees and game-searching procedures. *Artificial Intelligence* vol 14 no 2, 1980, 113-138.
- [Pearl, 1984] *Heuristics*, Addison-Wesley, Reading, MA, 1984.
- [Sturtevant, 2002] A Comparison of Algorithms for Multi-Player Games, *Proceedings of the 3rd International Conference on Computers and Games*, 2002.
- [Sturtevant and Korf, 2000] On Pruning Techniques for Multi-Player Games, *Proceedings AAAI-00*, Austin, TX, 201-207.
- [Sturtevant, 2003] Multi-Player Games: Algorithms and Approaches, PhD Thesis, UCLA.

MULTIAGENT SYSTEMS

LOGIC-BASED MAS AND
COMMUNICATION LANGUAGES

Protocol Conformance for Logic-based Agents

Ulrich Endriss and Nicolas Maudet and Fariba Sadri and Francesca Toni

Department of Computing, Imperial College London

180 Queen's Gate, London SW7 2AZ, UK

{ue,maudet,fs,ft}@doc.ic.ac.uk

Abstract

An agent communication protocol specifies the “rules of encounter” governing a dialogue between agents in a multiagent system. In non-cooperative interactions (such as negotiation dialogues) occurring in open societies it is crucial that agents are equipped with proper means to check, and possibly enforce, conformance to protocols. We identify different levels of conformance (weak, exhaustive, and robust conformance) and explore how a specific class of logic-based agents can exploit a new representation formalism for communication protocols based on simple if-then rules in order to either check conformance *a priori* or enforce it at runtime.

1 Introduction

A *protocol* specifies the “rules of encounter” governing a dialogue between two or more communicating agents [Rosen-schein and Zlotkin, 1994]. It specifies which agent is allowed to say what in any given situation. It will usually allow for several alternative utterances in every situation and the agent in question has to choose one according to its *strategy*. The protocol is *public*, while each agent's strategy is *private*. Protocols can help to define a suitable standardised semantics for agent communication languages in order to allow actual interoperability in open environments. Instead of being related to some (virtually inaccessible) private mental state of the agent as proposed by KQML [Labrou and Finin, 1998] and FIPA [2002], the meaning of a dialogue move refers to some publicly agreed and verifiable conversational state. For instance, one may equate the meaning of a dialogue move with the set of possible responses to this dialogue move as defined by the public protocol [Pitt and Mamdani, 1999]. Of course, this does not disqualify the idea of a private semantics, but simply emphasises that standardisation cannot be achieved at this level. Following these ideas, Pitt and Mamdani [1999] have established the notion of a layered semantics for agent communication languages which integrates both private and public levels, and each agent must implement both of them to actually be able to converse with other agents.

When considering interactions that are not necessarily co-operative (typically negotiation), it cannot be safely assumed that agents will always follow the rules of encounter specified

by a particular protocol. It is then crucial to provide proper means of evaluating how well agents are adapted to these protocols, in a sense that has to be precisely defined in the context of the interaction. In this paper, we introduce three different *levels of conformance* (weak, exhaustive, and robust conformance) and show that a logical representation of the protocols for logic-based agents greatly facilitates determining (both *a priori* and at runtime) whether or not these agents behave in conformance to the protocols. We also show how to *enforce* conformance at runtime for such agents and suggest a preliminary definition for the notion of *protocol competence*, an attempt to measure how well an agent is adapted to a given protocol beyond the basic requirements of conformance.

The remainder of this paper is organised as follows. In Section 2 we introduce a new formalism for the representation of protocols based on if-then-rules. Section 3 motivates the introduction of our three levels of conformance. Abductive logic programming has recently been used as a basis for defining (private) strategies for agents negotiating over the distribution of resources [Sadri *et al.*, 2001]. Section 4 introduces these agents, illustrates how to check protocol conformance for them, and shows how the agents themselves can ensure their conformance without requiring any extra reasoning machinery. Section 5 concludes the paper.

2 Representing Protocols

In this paper, we will assume some restrictions on the kind of dialogues that we want to deal with. The dialogues we are considering only involve *two agents* which *sequentially alternate* dialogue moves. These restrictions (notably avoiding concurrency) allow us to concentrate on a particular class of protocols, namely those representable by means of *deterministic finite automata* (DFAs), of which there are numerous examples to be found in the literature [Pitt and Mamdani, 1999; Dignum and Greaves, 2000]. Our aim for this section is to introduce a protocol representation formalism based on simple if-then-rules and to show that it is adequate to represent the kinds of protocols that we are interested in here.

We recall here that a DFA consists of (i) a set of states (including an initial state, and a set of final states), (ii) a set of events, and (iii) a transition function δ which maps pairs of states and events to states. Figure 1 shows an example, taken from [Pitt and Mamdani, 1999]. Events are occurrences of dialogue moves; states are the possible stages of the conver-

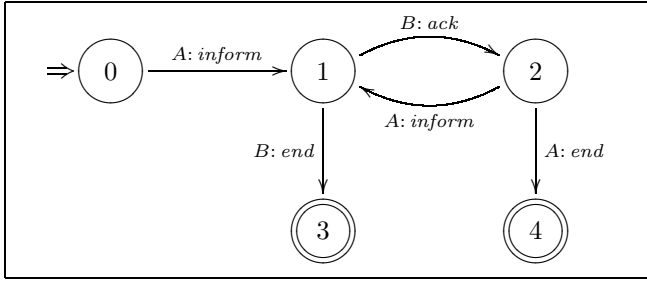


Figure 1: The continuous update protocol

sation. The protocol, entitled “continuous update protocol”, aims at continuously updating the other agent about the value of some proposition. Note that it is always the same agent A (the initiator) who informs the other agent B . The *legal* communicative behaviour of both agents is captured by such a protocol. We make this notion of legality precise in the following definition:

Definition 1 (Legality) *Given a DFA with transition function δ , a dialogue move P is a legal continuation wrt. a state S iff there exists a state S' such that $S' = \delta(S, P)$.*

We shall refer to *legal inputs* (respectively *outputs*) for an agent X as those legal continuations where X is the receiver (respectively the utterer) of the dialogue move.

Protocols such as that of Figure 1 can alternatively be represented as sets of if-then-rules which specify the set of correct responses for a particular incoming dialogue move. For example, to express that agent B could react to an *inform* move sent by A either by sending an acknowledgement or by terminating the dialogue, we may use the following rule:

$$\text{tell}(X, Y, \text{inform}(P), D, T) \Rightarrow \text{tell}(Y, X, \text{end}, D, T+1) \vee \text{tell}(Y, X, \text{ack}(P), D, T+1)$$

Note that variables are implicitly universally quantified, except those that only appear on the righthand side of an implication, which are existentially quantified. In general, in this representation, dialogue moves are instances of the schema

$$\text{tell}(X, Y, \text{Subject}, D, T),$$

where X is the utterer, Y is the receiver ($X \neq Y$), D the identifier of the dialogue, and T the time when the move is uttered. *Subject* is the type of the dialogue moves, i.e. a performative (such as *end*) of the communication language, possibly together with a content (as in *inform(P)*). We shall mostly use the abbreviated form $P(T)$ to dialogue moves, omitting the parameters not relevant in the discussion. For the sake of simplicity, we will assume that the start of the protocol is triggered by some external event *START*—it is possible to conceive this as the result of some meta-level negotiation process to agree on a particular protocol. The start signal *START* is sent by the system to exactly one agent and exactly once during a dialogue. Similarly, a dialogue ends once one of the agents sends the signal *STOP* to the system. Dialogue inputs for the agents are either dialogue moves sent by other agents or a *START* signal sent by the system.

Going back to the example of Figure 1, we observe that this automaton in fact represents two subprotocols, one for

the initiator, and one for its partner. We will refer to these two subprotocols as $\mathcal{P}_A$ and $\mathcal{P}_B$. They can be translated into a set (composed of two subsets) of if-then-rules as illustrated in Table 1. In general, given a protocol $\mathcal{P}$, we shall refer to the subprotocol guiding the behaviour of agent α as $\mathcal{P}_\alpha$.

This example suggests that we can simply translate protocols into if-then-rules where we have a single performative on the lefthand side. We call protocols that permit such a straightforward translation *shallow*. Shallow protocols correspond to DFAs where it is possible to determine the next state of the dialogue on the sole basis of the previous event. Of course, this is not always the case since it may be necessary to refer to the current state of the dialogue to determine the new state (think of two transitions with the same label leaving two different states and leading to two different states). In principle, any automata-based protocol can be transformed into a protocol that is shallow in this sense (by simply renaming any duplicate transitions). In fact, we have observed that many of the automata-based protocols proposed in the multiagent systems literature happen to be shallow already or could at least be made shallow by renaming only a small number of transitions.

Now suppose $P_1, \dots, P_n$ are the dialogue moves of our communication language (including *START* and *STOP*). In the light of the above remarks, we will more generally translate DFAs into two sets of rules of the form

$$P_j(T) \Rightarrow \bigvee_{i \in I} P_i(T+1) \text{ with } 1 \leq j \leq n \text{ and } I \subseteq \{1, \dots, n\}$$

where the righthand side of the rule defines the possible continuations wrt. the protocol after the input P_j (that we will sometimes refer to as the *trigger* of the rule). To ensure that this protocol is *well-formed*, we will require that the two sets of rules meet some requirements (R1–R5): there has to be at least one rule with *START* on the lefthand side in the protocol, and *START* may never occur on the righthand side (R1a, initial); there is at least one rule with *STOP* on the righthand side and *STOP* never occurs on the lefthand side (R1b, final); any dialogue move occurring on the righthand side of the first subprotocol also occurs on the lefthand side of the second one, and vice versa (R2, matching); every subprotocol includes the additional rule

$$\text{tell}(X, Y, S_1, T, D) \wedge \text{tell}(X, Y, S_2, T, D) \wedge S_1 \neq S_2 \Rightarrow \perp$$

to avoid concurrent moves (R3, non-concurrency); for each rule occurring in a subprotocol, if X is the receiver and Y the utterer of the dialogue move occurring on the lefthand side, it must be the case that X is the utterer and Y the receiver of every dialogue move occurring on the righthand side (R4, alternating); all dialogue moves occurring on the lefthand side of the rules of each subprotocol are distinct from each other (R5, distinct triggers).

The meaning of each rule which appears in a protocol is intuitively clear: it specifies for any expected dialogue move the set of correct responses the agent may utter in reply. The following definitions make these notions more precise:

Definition 2 (Expected inputs) *The set of expected inputs for an agent α wrt. a protocol $\mathcal{P}$ is defined as:*

$$\{P_j \mid [P_j(T) \Rightarrow \bigvee_{i \in I} P_i(T+1)] \in \mathcal{P}_\alpha\}$$

$$\begin{aligned}
\mathcal{P}_A : \begin{cases} \text{START}(X, Y, D, T) & \Rightarrow \text{tell}(Y, X, \text{inform}(P), D, T+1) \\ \text{tell}(X, Y, \text{ack}(P), D, T) & \Rightarrow \text{tell}(Y, X, \text{inform}(P'), D, T+1) \vee \text{tell}(Y, X, \text{end}, D, T+1) \\ \text{tell}(X, Y, \text{end}, D, T) & \Rightarrow \text{STOP}(Y, X, D, T+1) \end{cases} \\
\mathcal{P}_B : \begin{cases} \text{tell}(X, Y, \text{inform}(P), D, T) & \Rightarrow \text{tell}(Y, X, \text{ack}(P), D, T+1) \vee \text{tell}(Y, X, \text{end}, D, T+1) \\ \text{tell}(X, Y, \text{end}, D, T) & \Rightarrow \text{STOP}(Y, X, D, T+1) \end{cases}
\end{aligned}$$

Table 1: The continuous update protocol as a set of if-then-rules

Definition 3 (Correct responses) *The set of correct responses for an agent α wrt. a protocol $\mathcal{P}$ and a dialogue move P_j is defined as:*

$$\{P_i \mid [P_j(T) \Rightarrow \bigvee P_i(T+1)] \in \mathcal{P}_\alpha\}$$

Intuitively it is not sufficient for a dialogue move to be merely *expected* in order to be considered *legal*; it is also necessary that the move takes place at the right moment in the history of the dialogue. But note that it follows from the property of matching (R2) that the set of legal *inputs* for an agent at a certain stage of a dialogue is a subset of its expected inputs.

3 Levels of Conformance

Taking for granted that agents share the same language of communication, we are now going to define three different *levels of conformance* to a protocol. Note that we define these notions on the basis of the *observable* conversational behaviour of the agents (i.e. what they *utter*) without making further assumptions on how they actually come to *generate* these utterances. We start with the notion of *weak conformance*:

Definition 4 (Weak conformance) *An agent is weakly conformant to a protocol $\mathcal{P}$ iff it never utters any illegal dialogue moves (wrt. $\mathcal{P}$).*

The following theorem shows that, in the context of our shallow protocols, the concept of legality is reducible to that of correctness, meaning that we can avoid to inspect the legality of the input and thus avoid to deal with the complete dialogue history.

Theorem 1 *An agent that never utters an incorrect response in reply to an expected input of a shallow protocol $\mathcal{P}$ is weakly conformant to $\mathcal{P}$.*

Proof. (sketch) For shallow protocols, the current dialogue state is uniquely identifiable given the latest move in the dialogue. Hence, the notions of correct response and legal move coincide, i.e. an agent that never utters an incorrect response will never choose an illegal continuation and will therefore be weakly conformant to the protocol. $\square$

It is clear that any application governed by a protocol at least requires the level of weak conformance—otherwise it would not make sense to define a protocol in the first place. The notion of weak conformance captures that the agent does *not* utter any illegal moves, but does not actually require that the agent utters any dialogue move at all. For interactions where “silent moves” are undesirable, a stronger version of conformance is usually required. We make this idea precise with the notion of *exhaustive conformance*:

Definition 5 (Exhaustive conformance) *An agent is exhaustively conformant to a protocol $\mathcal{P}$ iff it is weakly conformant to $\mathcal{P}$ and it will utter at least one legal output move for any legal input of $\mathcal{P}$ it receives.*

Exhaustive conformance is certainly what is intuitively expected in *most* interactions—it is indeed often preferred to avoid considering silent moves as part of a protocol, at least to avoid confusion with lost messages. One may then argue that exhaustive conformance should be the minimum requirement for any interaction. We believe, however, it is worth making the distinction between weak and exhaustive conformance. The first reason is that there are examples where the lack of response can be considered to be part of the protocol. In such circumstances, it can be sufficient to design a weakly conformant agent, provided that silent moves will not have undesirable consequences. For instance, in a Dutch auction process “when there is no signal of acceptance from the other parties in the auction (other agents in the negotiation) the auctioneer makes a new offer which he believes more acceptable (by reducing the price). Here, because of the convention (protocol) under which the auction operates, a lack of response is sufficient feedback for the auctioneer to infer a lack of acceptance.” [Jennings *et al.*, 1998]. In this case, the agent can safely be designed to react appropriately only to the proposals it is ready to accept. But if we consider recent argumentation-based protocols inspired by dialectical models it is sometimes assumed that “silence means consent” [Amgoud *et al.*, 2000]. In this case, a lack of response can commit the receiver to some propositions—this is a typical case where it is crucial that agents are exhaustively conformant. The second reason for our distinction of weak and exhaustive conformance is that they are *conceptually* different since weak conformance only involves *not* uttering (any illegal moves), while exhaustive conformance involves uttering (some legal move). This implies substantially different approaches when the issues of checking and enforcing conformance are raised, as we shall see below.

Another important problem of agent communication is the need to deal with illegal incoming messages, and to react appropriately to recover from such violations. For instance, any FIPA-compliant communicative agent has to integrate a performative *not-understood* as part of its language [FIPA, 2002]. This motivates us to introduce the following notion of *robust conformance*:

Definition 6 (Robust conformance) *An agent is robustly conformant to a protocol $\mathcal{P}$ iff it is exhaustively conformant to $\mathcal{P}$ and for any illegal input move it will utter a special dialogue move (such as *not-understood*).*

Robust conformance goes a step further than exhaustive conformance since it requires that an appropriate response is uttered also in reply to illegal moves. Technically, this necessitates that the agent is able to identify the legality of an incoming dialogue move, i.e. it needs to be able to check conformance wrt. the *other* agent's subprotocol.

Note also that in the case where all agents in the society are known to be weakly conformant, it is theoretically unnecessary to deal with robust conformance (since no agent will ever utter an illegal move). Such an assumption would, however, somewhat contradict the “spirit” of an open society. We should also point out that in dialogues with a very high contingent of illegal utterances the additional **not-understood** moves may in fact burden communication channels unnecessarily and simply ignoring illegal moves would in fact be a better strategy.

4 Logic-based Agents

We are now going to consider the case of a specific class of agents based on abductive logic programming that have recently been used in the context of negotiation scenarios [Sadri *et al.*, 2001]. The communication *strategy* $\mathcal{S}$ of such an agent (which forms part of its so-called *knowledge base* $\mathcal{K}$) is represented as a set of integrity constraints of the following form:

$$P(T) \wedge C \Rightarrow P'(T+1)$$

On receiving dialogue move P at time T , an agent implementing this rule would utter P' at time $T+1$, provided condition C is entailed by its (*private*) knowledge base. Again, variables are understood to be implicitly quantified in the same way as our protocol-rules.

4.1 Checking Conformance

There are different ways of checking conformance. One way is to check conformance *on-the-fly*, i.e. to check step by step that every dialogue move uttered by an agent is conformant to the protocol. Another way—much more interesting when one has to design an agent to take part in an interaction—is to check conformance *a priori*, i.e. to check conformance by examining the specification of the agent rather than an actual dialogue. In general, this is a difficult task, because (i) the behaviour of the agent depends on some hardly tractable notions (e.g., beliefs and intentions), and (ii) conformance may depend on the history of the dialogue. We are now going to discuss some simple sufficient conditions for weak conformance (in the context of the logic-based agents introduced earlier) that may be checked *a priori*.

To begin with, we introduce the notion of *response space*. Intuitively, the response space of an agent specifies the possible moves that the agent can make when using a given strategy $\mathcal{S}$, *without* considering the specific conditions relating to its private knowledge base.

Definition 7 (Response space) *The response space of an agent with strategy $\mathcal{S}$ (henceforth noted $\mathcal{S}^*$) wrt. a communication language $\mathcal{L}$ is defined as the following set:*

$$\{P_j(T) \Rightarrow \bigvee \{P_i(T+1) \mid [P_j(T) \wedge C \Rightarrow P_i(T+1)] \in \mathcal{S}\} \mid P_j \in \mathcal{L}\}$$

with $\bigvee \{\} = \perp$.

That is, the response space is, essentially, the set of protocol constraints we get by first dropping all private conditions C and then conjoining implications with identical antecedents by collecting the corresponding consequents into a single disjunction. The reason why we define the disjunction of the empty set as $\perp$ will become clear when we consider the next theorem, which offers a very simple way to check weak conformance *a priori* for a logic-based agent. In particular, it avoids dealing with the dialogue history, and it does not make any assumptions on the content of the agent's knowledge base (except to require that it is possible to extract the response space, as previously described).

Theorem 2 *Let $\mathcal{P}$ be a protocol and let $\mathcal{S}^*$ be the response space of an agent A wrt. the language $\mathcal{L}$ of moves occurring in $\mathcal{P}$. If $\mathcal{S}^* \models \mathcal{P}$, then agent A is weakly conformant to $\mathcal{P}$.*

Proof. Let $\mathcal{S}^*$ be the response space of agent A wrt. the language $\mathcal{L}$ of protocol $\mathcal{P}$. Then for every rule $P \Rightarrow P_1 \vee \dots \vee P_n$ in $\mathcal{P}$, $\mathcal{S}^*$ will contain an implication $P \Rightarrow P'_1 \vee \dots \vee P'_m$ (with the same antecedent P)—and possibly a number of additional implications for dialogue moves in the language $\mathcal{L}$ that do not occur as triggers in $\mathcal{P}$. (All the antecedents of the implications in $\mathcal{S}^*$ are distinct.) Now suppose $\mathcal{S}^* \models \mathcal{P}$ holds. Observe that a formula of the form $P \Rightarrow P_1 \vee \dots \vee P_n$ in $\mathcal{P}$ can only be the consequence of an implication in $\mathcal{S}^*$ with the same antecedent P . But $P \Rightarrow P_1 \vee \dots \vee P_n$ will only follow from $P \Rightarrow P'_1 \vee \dots \vee P'_m$ provided we have $\{P'_1, \dots, P'_m\} \subseteq \{P_1, \dots, P_n\}$.

In other words, whenever $\mathcal{S}^* \models \mathcal{P}$ holds, then $\mathcal{S}^*$ will have the shape of a protocol that is a “syntactic restriction” of the protocol $\mathcal{P}$, possibly together with a number of irrelevant rules (with triggers not present in $\mathcal{P}$). Furthermore, by construction, any agent will be weakly conformant to the “protocol” represented by its response space. Hence, agent A will also be weakly conformant to $\mathcal{P}$, because any dialogue continuation that would be legal wrt. $\mathcal{S}^*$ would certainly have to be legal wrt. $\mathcal{P}$. $\square$

The opposite direction of Theorem 2 does not hold, because, looking at the form of strategies, it is clear that *private* conditions may prevent the agent from uttering a particular dialogue move. In other words, it could be the case that $\mathcal{S}^* \not\models \mathcal{P}$ but that the agent is still weakly conformant to $\mathcal{P}$ because of its specific knowledge base.

The same argument prevents us from having a theorem similar to Theorem 2 in the case of *exhaustive conformance* (and of course for robust conformance). Here we would have to look more precisely at how dialogue moves are actually being generated. The basic idea would be to check that, for every expected input in the protocol, the disjunction of the conditions C related to this move in the agent's strategy is entailed by the agent's knowledge base.

4.2 Enforcing conformance

Even when Theorem 2 is not applicable and an agent cannot be proven to be weakly conformant *a priori*, it is still possible to constrain the agent's behaviour in such a way as to simply prevent it from uttering illegal moves. The problem of *enforcing conformance* (referred to as *regimentation* by Jones

$S_1 :$	$\left\{ \begin{array}{l} \text{tell}(X, a, \text{inform}(P), D, T) \wedge \neg \text{friend}(a, X) \\ \text{tell}(X, a, \text{inform}(P), T) \wedge \text{friend}(a, X) \\ \text{tell}(X, a, \text{ack}, D, T) \\ \text{tell}(X, a, \text{question}(P), T) \wedge \text{friend}(a, X) \\ \text{tell}(X, a, \text{end}, D, T) \end{array} \right\}$	$\Rightarrow$	$\left\{ \begin{array}{l} \text{tell}(a, X, \text{end}, D, T+1) \\ \text{tell}(a, X, \text{ack}, D, T+1) \\ \text{tell}(a, X, \text{ack}, D, T+1) \\ \text{tell}(a, X, \text{inform}(P), D, T+1) \\ \text{STOP}(a, X, D, T+1) \end{array} \right\}$
		$\Rightarrow$	
		$\Rightarrow$	
		$\Rightarrow$	
		$\Rightarrow$	
$S_2 :$	$\left\{ \begin{array}{l} \text{tell}(X, a, \text{inform}(P), D, T) \wedge \text{enemy}(a, X) \\ \text{tell}(X, a, \text{inform}(P), D, T) \wedge \text{friend}(a, X) \\ \text{tell}(X, a, \text{inform}(P), D, T) \wedge \neg \text{friend}(a, X) \wedge \neg \text{enemy}(a, X) \\ \text{tell}(X, a, \text{ack}, D, T) \\ \text{tell}(X, a, \text{challenge}(P), D, T) \\ \text{tell}(X, a, \text{end}, D, T) \end{array} \right\}$	$\Rightarrow$	$\left\{ \begin{array}{l} \text{tell}(a, X, \text{end}, D, T+1) \\ \text{tell}(a, X, \text{ack}, D, T+1) \\ \text{tell}(a, X, \text{challenge}(P), D, T+1) \\ \text{tell}(a, X, \text{ack}, D, T+1) \\ \text{tell}(a, X, \text{end}, D, T+1) \\ \text{STOP}(a, X, D, T+1) \end{array} \right\}$
		$\Rightarrow$	
		$\Rightarrow$	
		$\Rightarrow$	
		$\Rightarrow$	

Table 2: Examples of dialogue strategies

and Sergot [1993]) is then to try to find easy (and hopefully automatic) ways to ensure that an agent will always be conformant to a given protocol.

We will now show how a simple filtering function may be defined in the context of our logic-based agents. Since this relies on the concrete behaviour of our agents, this requires some details about the operational model which underlies these agents. By a generated response we now mean a response produced by the abductive IFF proof procedure of Fung and Kowalski [1997]. In our case, the abducibles are the dialogue move(s) that the agent will actually generate as the consequence of the observation of another agent's dialogue move (see [Sadri *et al.*, 2002] for details).

Theorem 3 *An agent generating its dialogue moves wrt. the knowledge base $\mathcal{K} \cup \mathcal{P}$ will be weakly conformant to $\mathcal{P}$.*

Proof. (sketch) Let us assume that $P(T)$ is a legal input (wrt. protocol $\mathcal{P}$) that our agent receives, and that $P'(T+1)$ is a move generated as a response by the agent. Since the move is legal, it is expected in $\mathcal{P}$. The proof procedure will produce a tree and each branch of this tree will have one of the correct responses $P_i(T+1)$ together with the constraints of the dialogue strategy and the additional condition of non-concurrency (R3). Now if $P'(T+1)$ is a generated response of the proof procedure, $P'(T+1)$ must also be a conjunct on one of these branches. But such a branch will evaluate to false if $P'(T+1) \neq P_i(T+1)$, precisely because of the condition of non-concurrency. This means that the procedure will never generate an incorrect response. Therefore, by Theorem 1, our agent will be weakly conformant to $\mathcal{P}$. $\square$

Thus the filtering function is provided by the protocol itself. This result has very practical consequences. For instance, before entering an interaction, an agent can ensure its weak conformance to the public protocol which regulates the interaction (and thereby avoid possible penalties imposed by society), by adding the protocol to its own knowledge base. Now, while we can prevent an agent from uttering illegal dialogue moves, it is difficult to see how we could force it to utter some dialogue move (exhaustive conformance) without considering its private knowledge base and modifying its private strategy. We therefore believe that exhaustive conformance cannot be enforced automatically and should ideally be ensured during the specification of the agent by the designer.

Note that this filtering function will not block correct responses generated by the agent, unless two or more moves are being generated at the same time. This is again due to the non-concurrency condition (R3).

We should also point out that it is possible that an agent with knowledge base $\mathcal{K}$ would not utter any dialogue move given a particular input $P(T)$, while an agent with knowledge base $\mathcal{K} \cup \mathcal{P}$ (i.e. the same agent after “downloading” the protocol $\mathcal{P}$) would utter a (legal) move. This may, for instance, be the case if $P(T)$ never occurs on the lefthand side in the agent's original strategy and $\mathcal{P}$ includes a “deterministic” protocol rule such as $P(T) \Rightarrow P'(T+1)$. If this type of behaviour is not intended, we have to ensure that the communication languages of the agent and the protocol match (for instance, by adding the constraint $P(T) \Rightarrow \perp$ to the agent's strategy for every move P that is an expected input in $\mathcal{P}$).

4.3 Examples

We are now going to illustrate some of the points made earlier by means of example. Consider again the continuous update protocol of Table 1, and more specifically subprotocol $\mathcal{P}_B$. The set of constraints given in Table 2 represents two possible strategies for an agent a . Here, the $\neg$ -operator is understood as negation as failure. We assume that the predicates *friend* and *enemy* are fully defined in the knowledge base of our agent (that is, it can determine whether or not an agent X is a friend or an enemy), and we also assume that an integrity constraint stipulates that it is not possible to be at the same time friend and enemy of the same agent ($\text{friend}(a, X) \wedge \text{enemy}(a, X) \Rightarrow \perp$).

Dialogue strategy S_1 relies upon the communication language $\mathcal{L}_1$ which is based on the set $\{\text{inform}(P), \text{question}(P), \text{ack}, \text{end}\}$. It is interesting to note that no rule can be triggered after a *question* about P if it is the case that the agent a does not consider X as a friend. However, the agent a is still weakly and exhaustively conformant to $\mathcal{P}_B$ since this move can never be legally uttered within the protocol $\mathcal{P}$ by its partner (it is not a legal input for subprotocol $\mathcal{P}_B$).

Next consider dialogue strategy S_2 , which relies upon the communication language $\mathcal{L}_2$ based on the set $\{\text{inform}(P), \text{challenge}(P), \text{ack}, \text{end}\}$. Agent a is not weakly conformant to $\mathcal{P}_B$ since, in some situations (namely, when it does not consider X as friend, neither as an enemy), it may want to *challenge* the initiator X . This move is not expected

in the protocol $\mathcal{P}_A$. If the agent a decides to enforce weak conformance as described earlier in Theorem 3, it will remain silent in the situation previously described.

Finally, let us consider an agent with the response space $\{P \Rightarrow \perp \mid P \in \mathcal{L}\}$, that is, an agent that never utters any dialogue moves at all. It is clear that such an agent will be weakly conformant to any protocol. This certainly suggests that the notion of conformance *alone* is not sufficient to evaluate how well an agent is adapted to a protocol.

Another application of the notion of response space could be to assess how well an agent can explore a given protocol beyond the minimal requirement of being able to conform to it. Intuitively, given a protocol $\mathcal{P}$, we would expect a “competent” agent to have a response space that (almost) “covers” $\mathcal{P}$, namely it has the potential to utter as many dialogue moves as the protocol allows. This would offer a notion complementary to that of conformance.

Note also that none of the agents discussed here is robustly conformant. One of the reasons for this latter remark is the fact that these agents cannot notice that the dialogue move they receive is not legal if it is an expected input (for instance if their partner starts the dialogue with an *ack*). They would consequently react “appropriately” to these moves—which can be quite confusing.

5 Conclusion

This paper has introduced different levels of conformance as basic notions to check and enforce that the behaviour of an agent is adapted to a public protocol regulating the interaction in a multiagent system. These notions have been explored in the context of the logic-based agents of Sadri *et al.* [2001]. Our approach starts from an alternative representation formalism for communication protocols based on if-then-rules for the kinds of protocols that can be represented as DFAs. In particular, we argue that it is typically not necessary to consider the history of the dialogue besides the previous move to determine the possible legal dialogue continuations (*shallowness*). This allows us to show that a sufficient condition for an agent to be weakly conformant is to never give an incorrect response in reply to an expected input. In the context of logic-based agents we have used this result to propose a simple way to check a sufficient condition for weak conformance, which has the great advantage of ignoring the conditions related to the (private) knowledge base (by identifying an agent’s *response space*), and avoids to consider the dialogue history. We have then shown how the concrete behaviour of the IFF proof procedure can be used to define a regimentation policy to enforce weak conformance. Finally, we have illustrated these notions and their limitations by means of examples.

A number of results have previously been obtained in the context of agents based on abductive logic programming, e.g. the termination of certain types of negotiation dialogues [Sadri *et al.*, 2001]. On the one hand, because we mostly ignore the details of the knowledge bases of agents and do not restrict ourselves to any specific protocol here, we cannot obtain such strong results. On the other hand, our work provides more general tools for logic-based agents communicating by means of (various) public protocols.

The work presented here can be extended in a number of ways. In particular, we only deal with very specific kinds of dialogues (although they can be generated by a wide range of protocols). Amongst other things, we plan to explore whether our formalism is expressive enough to capture more complex dialogues, such as those taking place between more than two agents or those not representable using DFAs (e.g. dialogues where concurrency is allowed or where checking conformance requires reference to the content of a dialogue move rather than just the communicative act itself).

Acknowledgements. We would like to thank the IJCAI referees for their helpful comments. This research has been funded by the European Union as part of the SOCS project (Societies Of Computees), IST-2001-32530.

References

- [Amgoud *et al.*, 2000] L. Amgoud, N. Maudet, and S. Parsons. Modelling Dialogues using Argumentation. In *Proc. ICMAS-2000*, 2000.
- [Dignum and Greaves, 2000] F. Dignum and M. Greaves. Issues in Agent Communication: An Introduction. In F. Dignum and M. Greaves, editors, *Issues in Agent Communication*. Springer-Verlag, 2000.
- [FIPA, 2002] Foundation for Intelligent Physical Agents (FIPA). *Communicative Act Library Specification*, 2002. <http://www.fipa.org/specs/fipa00037/>.
- [Fung and Kowalski, 1997] T. H. Fung and R. Kowalski. The IFF Proof Procedure for Abductive Logic Programming. *Journal of Logic Programming*, 33(2):151–165, 1997.
- [Jennings *et al.*, 1998] N. Jennings, S. Parsons P. Noriega, and C. Sierra. On Argumentation-based Negotiation. In *Proc. IWMAS-1998*, 1998.
- [Jones and Sergot, 1993] A. Jones and M. Sergot. On the Characterisation of Law and Computer Systems: The Normative Systems Perspective. In *Deontic Logic in Computer Science: Normative System Specification*. John Wiley and Sons, 1993.
- [Labrou and Finin, 1998] Y. Labrou and T. Finin. Semantics and Conversations for an Agent Communication Language. In M. Huhns and M. Singh, editors, *Readings in Agents*. Morgan Kaufmann Publishers, 1998.
- [Pitt and Mamdani, 1999] J. Pitt and A. Mamdani. A Protocol-based Semantics for an Agent Communication Language. In *Proc. IJCAI-1999*. Morgan Kaufmann, 1999.
- [Rosenschein and Zlotkin, 1994] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter*. MIT Press, 1994.
- [Sadri *et al.*, 2001] F. Sadri, F. Toni, and P. Torroni. Dialogues for Negotiation: Agent Varieties and Dialogue Sequences. In *Proc. ATAL-2001*. Springer-Verlag, 2001.
- [Sadri *et al.*, 2002] F. Sadri, F. Toni, and P. Torroni. An Abductive Logic Programming Architecture for Negotiating Agents. In *Proc. JELIA-2002*. Springer-Verlag, 2002.

Hidden Uncertainty in the Logical Representation of Desires

Jérôme Lang

IRIT / UPS

31062 Toulouse Cedex

France

lang@irit.fr

Leendert van der Torre

CWI

Amsterdam

The Netherlands

torre@cwi.nl

Emil Weydert

University of Applied Sciences

Luxembourg

Luxembourg

emil.weydert@ist.lu

Abstract

In this paper we introduce and study a logic of desires. The semantics of our logic is defined by means of two ordering relations representing preference and normality as in Boutilier's logic QDT. However, the desires are interpreted in a different way: "in context A , I desire B " is interpreted as "the best among the most normal $A \wedge B$ worlds are preferred to the most normal $A \wedge \neg B$ worlds". We study the formal properties of these desires, illustrate their expressive power on several classes of examples and position them with respect to previous work in qualitative decision theory.

1 Introduction

Autonomous agents reason frequently about preferences, desires and goals. For example, Cohen and Levesque [1990] explore principles governing the rational balance among an agent's beliefs, goals, actions and intentions, Rao and Georgeff [1991] show how different types of rational agents can be modeled by imposing certain conditions on the persistence of an agent's beliefs, desires or intentions (the BDI model) and work in qualitative decision theory [Pearl, 1993; Boutilier, 1994; Bacchus and Grove, 1996; Doyle and Thomason, 1999; Thomason, 2000] illustrates how planning agents are provided with goals – defined as desires together with commitments – and charged with the task of discovering and performing a sequence of actions to achieve these goals.

In logical formalizations of preferences, desires and goals serve as a computationally useful partial specification or heuristic approximation of the relative preference over the possible results of a plan [Doyle, 1991]. In this paper, we focus on desires although our observations may be relevant for goals as well. There are three different interpretations of sentences like "I desire B " or "in context A , I desire B ".

First, desires may be formalized using only utility or preference. However, this leads to the problem that even if a utility function or a preference relation over worlds is fixed, there is no unique way to extend it to formulas or, equivalently, sets of worlds. This distinction between utilities and probabilities is the main reason why formalizing desires is more problematic than formalizing beliefs. Moreover, it does not seem to correspond to the meaning of desire in natural language. For

instance, suppose that I exceptionally get sunburned when the weather is hot and sunny. Then, expressing "I desire a hot and sunny weather" intuitively means that the most normal or typical worlds satisfying $hot \wedge sunny$ are preferred to the most normal worlds verifying $\neg (hot \wedge sunny)$, but it does not mean that I like exceptional effects of $hot \wedge sunny$ such as *sunburn*.

Secondly, desires may express a combination of utility and probability in a classical decision-theoretic context, assuming a probability distribution over worlds. "In context A , I desire B " can then be interpreted as a raise of expected utility: *my expected utility given $A \wedge B$ is higher than my expected utility given $A \wedge \neg B$* . A further development in terms of gain of expected utility is proposed in [Brafman and Friedman, 2001]. However, it is well-known that autonomous agents do not always have appropriate probabilistic information on the possible worlds. The probabilistic approach asks for costly, specific information and in practice often leads to an arbitrary choice of probability values.

Thirdly, desires may implicitly refer to the relative *plausibility* or *normality* of worlds. When an agent states "I desire B ", i.e., "I desire B to be satisfied", the agent often focuses on typical, normal worlds. The standard choice made in *qualitative decision theory*, see [Doyle and Thomason, 1999], is that uncertainty is described by a (*total*) *pre-order* expressing, e.g., plausibility, normality, or typicality. The main justification is that ordinal uncertainty is less committing, cognitively speaking, than numerical uncertainty.

The research question of this paper is how we can formalize desires to provide a realistic interaction between preference and normality, and such that realistic examples like examples 2.4, 3.3, and 3.5 can be formalized. E.g., in the latter an airline company desires for an overbooked plane that an individual passenger shows up, but at the same time it desires that not all passengers show up. We start with Boutilier's notion of so-called ideal goals [1994], but then we define conditional desires that refer to both preference and normality, and that turn out to be more satisfactory.

The layout of this paper is as follows. Section 2 recalls the basics of Boutilier's semantics and discusses the choices upon which it is based; we then explain how to extend it to represent desires in a more appropriate way. The details of this modified interpretation of desires are given in Section 3. We study several examples and in particular, we illustrate how our semantics interprets sets of desires that are usually considered contradictory.

2 Boutilier's qualitative decision theory logic

A first class of solutions to combine preference and normality makes use of a trade-off between qualitative utilities and probabilities as in [Pearl, 1993], but this is not well-suited to situations with non-extreme utilities. A second approach is given in [Boutilier, 1994], which makes use of two ordering relations, one for preference and one for normality, representing the ordinal counterparts of utility and probability. However, here we see – informally speaking – a lack of interaction between preference and normality.

2.1 Definitions

Boutilier [1994] interprets a desire in terms of ideal worlds: *if A then ideally B*, denoted by $\mathbf{I}(B|A)$, is a conditional desire expressing on the semantic level that among all A -worlds, the most preferred ones also satisfy B . Quoting Boutilier, “from a practical point of view, $\mathbf{I}(B|A)$ means that if the agent (only) knows A , and the truth value of A is fixed (beyond its control), then the agent ought to ensure B (...) The statement can be *roughly* interpreted as if A , do B .”

This definition enables the formalization of conditional desires in the conditional logic QDT, which is based on his logic CO. *Roughly*, the semantics of QDT is the following: a QDT-model is $M = \langle W, \geq_P, \geq_N, val \rangle$ where W is a set of possible worlds, val a valuation function and $\geq_P$ and $\geq_N$ are two total pre-orders over W , i.e., transitive connected relations, where a relation is connected iff $w \geq v$ or $v \geq w$ for all $w, v \in W$. They are also called total weak orders. $\geq_P$ is a *preference* order and $\geq_N$ a *normality* order: $w \geq_P w'$ (resp. $w \geq_N w'$) means that w is at least as preferred (resp. as normal) as w' . The conditional connectives $\mathbf{I}(-| -)$ and $\mathbf{N}(-| -)$ have the following truth conditions. The two modalities do not mix preference and normality; in particular, conditional desires $\mathbf{I}(-| -)$ do not use the normality relation at all.

- $M \models \mathbf{I}(B|A)$ if and only if either $\forall w \in W, M, w \models \neg A$, or there exists a $w \in W$ such that $M, w \models A$ and for all $w' \in W$ with $w' \geq_P w, M, w' \models A \rightarrow B$. $\mathbf{I}(B|A)$ expresses the conditional preference “if A then ideally B ”.
- $M \models \mathbf{N}(B|A)$: as $\mathbf{I}(B|A)$, except that $\geq_P$ is replaced by $\geq_N$. $\mathbf{N}(B|A)$ expresses the conditional default “if A then normally B ”.

In the rest of the paper we write $\mathbf{I}(B)$ for $\mathbf{I}(B|\top)$ and $\mathbf{N}(B)$ for $\mathbf{N}(B|\top)$, where $\top$ denotes a tautology.

Boutilier also introduces *ideal goals* that combine preference and normality. However, they use only the upper cluster of the normality relation, nothing else. On the semantic level, if we assume that there are no infinite ascending chains, then the truth conditions for ideal goals are as follows. $\mathbf{IG}(B|A)$ is true if the best of the most normal A worlds satisfy B .

- $M \models \mathbf{IG}(B|A)$ if and only if

$$\text{Max}(\geq_P, \text{Max}(\geq_N, \text{Mod}(A))) \subseteq \text{Mod}(B)$$

Therefore, Boutilier's interpretation of conditional desires relies on two strong assumptions, which we discuss in more detail below. The first assumption is the optimistic interpretation of desires due to the ideality assumption. The second assumption is the weak interaction between preference and normality due to the focus on the most normal worlds in a context.

2.2 Assumptions

Ideality

The ideality semantics consists in comparing sets of worlds by looking only at the most preferred worlds of these sets. It corresponds to an optimistic point of view in the sense that less preferred worlds are ignored in this process. What would be possible choices for comparing sets of worlds? If we do not want to bring in probabilistic information or assumptions such as equiprobability, then we are left with the following four basic alternatives, and variations or combinations thereof. Let A and B be two formulas. For the sake of simplicity we only consider the non-degenerate case where $\text{Mod}(A \wedge B)$ and $\text{Mod}(A \wedge \neg B)$ are both non-empty.

MM (ideality semantics): $\mathbf{I}^{MM}(B|A)$ if and only if the best $A \wedge B$ -worlds are preferred to the best $A \wedge \neg B$ -worlds.

mm : $\mathbf{I}^{mm}(B|A)$ if and only if the worst $A \wedge B$ -worlds are preferred to the worst $A \wedge \neg B$ -worlds.

mM : $\mathbf{I}^{mM}(B|A)$ if and only if the worst $A \wedge B$ -worlds are preferred to the best $A \wedge \neg B$ -worlds.

Mm : $\mathbf{I}^{Mm}(B|A)$ if and only if the best $A \wedge B$ -worlds are preferred to the worst $A \wedge \neg B$ -worlds.

We have that $\mathbf{I}^{mM}(B|A)$ implies both $\mathbf{I}^{mm}(B|A)$ and $\mathbf{I}^{MM}(B|A)$, and all of them imply $\mathbf{I}^{Mm}(B|A)$. Note that only MM and mm are consistent with the semantics of conditional logics; for mm , just take the reverse preference relation. The Mm variant is extremely weak and therefore tells us next to nothing. It may be useful only when it is extended with a non-monotonic reasoning mechanism. The mM variant is extremely strong and hard to satisfy; it certainly does not reflect the usual intuitive understanding of desires. It may be useful only when paired with normality, e.g., by focusing on the most normal worlds.

The mm variant has an underlying *pessimistic semantics*. It makes sense if “I desire B in context A ” is interpreted as “in context A , given that I expect the worst outcomes to occur, I am happy to see B true”. However, this semantics also does not fit well the intuitions behind the specification of desires. Consider the following example of two desires.

Example 2.1 (Game) Assume that an agent plays a game where two coins are tossed; he wins if both coins are heads (h) and loses otherwise. His preference ordering can be: $(h_1, h_2) >_P (h_1, \neg h_2) \sim_P (\neg h_1, h_2) \sim_P (\neg h_1, \neg h_2)$; the normality ordering $\geq_N$ is the one where all worlds are equally normal. But, surprisingly, neither $\mathbf{I}^{mm}(h_1)$ nor $\mathbf{I}^{mm}(h_2)$ are satisfied.

The example supports that Boutilier's ideality semantics best suits the commonsense intuitions concerning desired outcomes of actions, and we therefore do not want to give it up.

At a first glance, it may seem paradoxical to favor an optimistic interpretation of desires while most papers in qualitative decision theory argue in favor of a pessimistic (max-min) criterion for action selection [Brafman and Tennenholtz, 1997; Dubois *et al.*, 1998]. The paradox is however only in appearance. When specifying $\mathbf{I}(B|\top)$, for instance, an agent expresses that she has a preference for B , which is not at all the same as saying that she intends to take a decision making B true. This is illustrated by the following classical example.

Example 2.2 [Boutilier, 1994] Let $DS = \{ \mathbf{I}(\neg u), \mathbf{I}(u|r) \}$ where u and r stand for umbrella and raining. $\mathbf{I}(\neg u)$ expresses that the agent prefers not carrying an umbrella. This does not imply that if the agent has the choice between the actions take-umbrella and leave-umbrella (whose obvious outcomes are u and $\neg u$), she will choose leave-umbrella.

The example illustrates that the interpretation of desires and the action selection criterion are independent issues and we can consistently interpret desires in Boutilier’s ideality semantics while using a pessimistic criterion for action choice. This issue is discussed further in Section 4.

Weak interaction between normality and preference

This assumption is more problematic than the previous one. Boutilier’s interpretation of ideal goals makes a very rough use of the normality relation, since it consists in focusing first, and once for all, on the most normal worlds, independently of the desires expressed, and then in interpreting desires by the ideality semantics.

This has unfortunate consequences. Ideal goals are different from desires because we have the counterintuitive property that if normally p then p is an ideal goal. Moreover, Boutilier’s semantics makes intuitively coherent sets of desires inconsistent, such as the *dog and fence* example of de-feasible deontic logic [van der Torre and Tan, 1997].

Example 2.3 (Dog and fence)

$DS = \{ \mathbf{I}(\neg f), \mathbf{I}(f|d), \mathbf{I}(d) \}$

1. John does not want a fence around his cottage;
2. If John owns a dog, then he wants a fence;
3. John wants to own a dog.

Example 2.3 is inconsistent, and would still be inconsistent if we would replace the $\mathbf{I}$ modality by $\mathbf{IG}$.

Here is another example which is inconsistent in Boutilier’s semantics. Suppose that I am going to my travel agent just one day before Christmas vacation, when normally all flights are fully booked.

Example 2.4 (Airplane ticket)

$DS = \{ \mathbf{I}(r), \mathbf{I}(a), \neg \mathbf{I}(r \wedge a), \mathbf{N}(\neg r \wedge \neg a) \}$

1. I desire to have an airplane ticket to Rome;
2. I desire to have an airplane ticket to Amsterdam;
3. I do not desire to have both an airplane ticket to Rome and an airplane ticket to Amsterdam;
4. Normally, I will neither get an airplane ticket to Rome nor to Amsterdam.

Example 2.4 is inconsistent, and would still be so if we replaced the $\mathbf{I}$ modality by $\mathbf{IG}$. However, this set of desires has an intuitive interpretation: when I think of having a ticket to Rome, I think of the most normal world where I have a ticket to Rome, in which I do not have a ticket to Amsterdam, and vice versa, and I prefer this world to the most normal world where I do not have a ticket to Rome, in which I do not have a ticket to Amsterdam either. See another interpretation of this set of desires in example 3.6.

3 Hidden uncertainty

Example 2.4 illustrates that, even if the intuitive expression of the desires does not mention normality or uncertainty issues, it implicitly refers to these. This is why we talk of *hidden uncertainty* in the specification of desires.

3.1 Definitions

We now introduce a notion of desires with hidden uncertainty that better fits the intuitive meaning of desires. To simplify the definition, we assume – in contrast to Boutilier – the existence of maxima. We may guarantee this by stipulating, e.g., that there are no infinite ascending chains. $\mathbf{D}(B|A)$ means that there are maximally normal $A \wedge B$ -worlds which are strictly preferred to all the most normal $A \wedge \neg B$ -worlds.

Definition 3.1 Truth conditions for desires are as follows. $M \models \mathbf{D}(B|A)$ if and only if $\forall w' \in \text{Max}(\geq_N, \text{Mod}(A \wedge \neg B)) \exists w \in \text{Max}(\geq_N, \text{Mod}(A \wedge B))$ such that $w >_P w'$.

In other words, “in context A , I desire B ” is interpreted as “the best among the most normal $A \wedge B$ worlds are preferred to the most normal $A \wedge \neg B$ worlds”. As usual, we write $\mathbf{D}(B)$ instead of $\mathbf{D}(B|\top)$.

There exists also a probabilistic interpretation of desires which may help to clarify their meaning. The basic idea is to use the ϵ -semantics for normality, a super- ϵ -semantics – e.g. replacing ϵ by ϵ^ϵ – for utility, and the resulting concept of expected utility for interpreting desires.

In a sense, our definition of desires is still reminiscent of the ideality semantics, because for a desire $\mathbf{D}(B|A)$ we only consider the most normal $A \wedge B$ and the most normal $A \wedge \neg B$ worlds. However, the fact that we do not only consider the most normal A worlds makes a crucial difference. This is illustrated by the properties and examples below.

3.2 Properties

The definition guarantees that the conditional desire always holds if the implication is strict, i.e., if $A \wedge \neg B$ is inconsistent. When all worlds are equally normal, desires represent pure preference in the sense that both modalities $\mathbf{D}()$ and $\mathbf{I}()$ coincide. However, in general none of the following implications $\mathbf{D}(A|B) \Rightarrow \mathbf{I}(A|B)$, $\mathbf{I}(A|B) \Rightarrow \mathbf{D}(A|B)$, $\mathbf{D}(A|B) \Rightarrow \mathbf{IG}(A|B)$ and $\mathbf{IG}(A|B) \Rightarrow \mathbf{D}(A|B)$ holds. Furthermore it avoids – as opposed to some of its variants – the validation of conditional desires of the form $\mathbf{D}(\neg A|A)$ for consistent A .

The following example illustrates that we have no longer right weakening (strictly speaking, left weakening, given our notation) for the desires, i.e. $\mathbf{D}(A)$ does not imply $\mathbf{D}(A \vee B)$.

Example 3.1 Let M be the following model in which $\neg p$ is always more plausible than p , whereas p is always preferred to $\neg p$.

$$\begin{aligned} \geq_N: & \{(\neg p, q), (\neg p, \neg q)\} >_N \{(p, q), (p, \neg q)\}; \\ \geq_P: & (p, \neg q) >_P (p, q) >_P (\neg p, \neg q) >_P (\neg p, q) \end{aligned}$$

Then M satisfies $\mathbf{D}(p)$ but not $\mathbf{D}(p \vee q)$.

This absence of right weakening, a controversial property when reasoning with obligations in deontic logics, is also very natural for desires: take p = “the woman of my dreams falls in love with me” and q = “I receive my electricity bill” (needless to say where the most normal worlds are).

The violation of right weakening also explains the distinction between the two conflicts in the following example. The example also illustrates that the conjunction (or AND) rule is not valid for **D**.

Example 3.2 $\{\mathbf{D}(p), \mathbf{D}(\neg p)\}$ is inconsistent, but $\{\mathbf{D}(p), \mathbf{D}(\neg p \wedge q)\}$ is consistent when q is exceptional, as is witnessed by the following model:

$$\begin{aligned} \geq_N: & \{(p, \neg q), (\neg p, \neg q)\} >_N \{(p, q), (\neg p, q)\} \\ \geq_P: & (\neg p, q) >_P (p, \neg q) >_P \{(\neg p, \neg q), (p, q)\} \end{aligned}$$

Interestingly, many rules valid for **I**($B|A$) do no longer hold for **D**($B|A$), as illustrated by the two examples above. Or, more precisely, they hold only conditionally. The reason is a kind of context-dependence or “higher-order nonmonotonicity”. For instance, if $\text{Mod}(A \wedge B)$ increases or decreases, it does not follow that $\max(\geq_N, \text{Mod}(A \wedge B))$ increases or decreases as well. One way to design proof rules is to ensure with additional clauses that this implication holds, which has been done for decision-theoretic defaults [Brafman and Friedman, 2001]. However, our logic of desires also validates inference rules which do not directly depend on such conditions.

The first formula of the following proposition illustrates that the conjunction rule holds under normality conditions $\mathbf{N}(A|B \wedge C)$ and $\mathbf{N}(B|A \wedge C)$, and the latter two formulas hold without any conditions.

Proposition 3.1 *The following formulas are theorems of our logic:*

1. $\mathbf{D}(A|C) \wedge \mathbf{D}(B|C) \wedge \mathbf{N}(A|B \wedge C) \wedge \mathbf{N}(B|A \wedge C) \Rightarrow \mathbf{D}(A \wedge B|C)$;
2. $\mathbf{D}(A|B \wedge C) \wedge \mathbf{D}(B|C) \Rightarrow \mathbf{D}(A \wedge B|C)$;
3. $\mathbf{D}(A|A \vee B) \wedge \mathbf{D}(\neg A|\neg A \vee B) \Rightarrow \mathbf{D}(\neg B|\top)$;

Properties expressed in the dyadic modal logic are often hard to read. We therefore propose an alternative strategy for generating proof rules. The idea is to express desires with a suitable order modality $<_{nd}$ where $A <_{nd} B$ means that, taking into account normality and adopting an optimistic perspective, A is less desirable than B , or inconsistent. Formally speaking, we get the following truth condition for $<_{nd}$ (assuming finitely many worlds).

Definition 3.2 *Truth condition of preference is as follows. $M \models A <_{nd} B$ if and only if $\forall w' \in \text{Max}(\geq_N, \text{Mod}(A)) \exists w \in \text{Max}(\geq_N, \text{Mod}(B))$ such that $w >_P w'$.*

We have the following relationship between the conditional desires and preferences.

Proposition 3.2 *The following translation rules are valid between $<_{nd}$ and **D**($\cdot|\cdot$).*

- $M \models A <_{nd} B$ iff $M \models \mathbf{D}(\neg A|A \vee B)$
- $M \models \mathbf{D}(B|A)$ iff $M \models (A \wedge \neg B) <_{nd} (A \wedge B)$

Similarly, we can introduce $<_n$ for expressing propositional normality comparisons. Taken together, these modalities allow us to formulate a number of features which are conceptually easier to grasp and to handle than the corresponding conditional notions. The following proposition lists several properties using this alternative representation.

Proposition 3.3 *The following formulas are theorems of the logic. They are called left weakening (L1), right weakening (L2), left strengthening (L3), left impossibility (L4), right possibility (L5), asymmetry (L6), left disjunction (L7), right disjunction (L8), transitivity (L9), left cautious transitivity (L10), right cautious transitivity (L11).*

- L1** $(A <_{nd} B) \wedge (C <_n A) \rightarrow ((A \vee C) <_{nd} B)$
- L2** $(A <_{nd} B) \wedge \neg(B <_n C) \rightarrow (A <_{nd} (B \vee C))$
- L3** $(A <_{nd} B) \wedge \neg((A \wedge C) <_n A) \rightarrow ((A \wedge C) <_{nd} B)$
- L4** $(\perp <_{nd} A)$
- L5** $(A <_{nd} \perp) \rightarrow (A <_n \perp)$
- L6** $(A <_{nd} B) \wedge \neg(A <_n \perp) \wedge \neg(B <_n \perp) \rightarrow \neg(B <_{nd} A)$
- L7** $(A <_{nd} C) \wedge (B <_{nd} C) \rightarrow ((A \vee B) <_{nd} C)$
- L8** $(A <_{nd} B) \wedge (A <_{nd} C) \rightarrow (A <_{nd} (B \vee C))$
- L9** $(A <_{nd} B) \wedge (B <_{nd} C) \rightarrow (A <_{nd} C)$
- L10** $(A <_{nd} B) \wedge ((A \vee B) <_{nd} C) \rightarrow (A <_{nd} C)$
- L11** $(A <_{nd} (B \vee C)) \wedge (B <_{nd} C) \rightarrow (A <_{nd} C)$

Due to lack of space we must omit proofs and discussion.

3.3 Examples

Let us consider Example 2.3 and 2.4 taking hidden uncertainty into account.

Example 3.3 (Dog and fence, continued)

$\{\mathbf{D}(d), \mathbf{D}(\neg f), \mathbf{D}(f|d)\}$ is consistent. Here are four classes of models, which illustrate that for each normality ordering there are various preference orderings.

class 1: *dogs are exceptional (and nothing else)*

$$\begin{aligned} \geq_N: & \{(\neg d, f), (\neg d, \neg f)\} >_N \{(d, f), (d, \neg f)\} \\ \geq_P: & \text{any complete preordering satisfying constraints} \\ & (d, f) >_P (d, \neg f) \\ & (d, f) >_P (\neg d, \neg f) >_P (\neg d, f) \end{aligned}$$

class 2: *not having a fence is exceptional (and nothing else)*

$$\begin{aligned} \geq_N: & \{(\neg d, f), (d, f)\} >_N \{(\neg d, \neg f), (d, \neg f)\} \\ \geq_P: & \text{any complete preordering satisfying constraints} \\ & (\neg d, \neg f) >_P (\neg d, f), (d, f) >_P (d, \neg f) \\ & (d, f) >_P (\neg d, f) \end{aligned}$$

class 3: *dogs and not having a fence are both exceptional*

$$\begin{aligned} \geq_N: & (\neg d, f) >_N \{(d, f), (\neg d, \neg f)\} >_N (d, \neg f) \\ \geq_P: & \text{any complete preordering satisfying constraints} \\ & (\neg d, \neg f) >_P (\neg d, f), (d, f) >_P (\neg d, f) \\ & (d, f) >_P (d, \neg f) \end{aligned}$$

class 4: *neither dogs nor not having a fence is exceptional*

$$\begin{aligned} \geq_N: & (d, \neg f) >_N (\neg d, f) >_N (d, f) >_N (\neg d, \neg f) \\ \geq_P: & \text{any complete preordering satisfying constraints} \\ & (d, \neg f) >_P (\neg d, f), (d, f) >_P (\neg d, f) \end{aligned}$$

Example 3.4 (Airplane ticket, continued)

$DS = \{\mathbf{D}(r), \mathbf{D}(a), \neg \mathbf{D}(r \wedge a), \mathbf{N}(\neg r \wedge \neg a)\}$. Here is a set of models satisfying DS :

$$\begin{aligned} \geq_N: & (\neg r, \neg a) >_N \{(r, \neg a), (\neg r, a)\} >_N (r, a); \\ \geq_P: & \text{any order where } (r, \neg a), (\neg r, a) \text{ are strictly more preferred than the other worlds.} \end{aligned}$$

The following example is a reformulation of the lottery paradox in terms of desires.

Example 3.5 (overbooking)

The agent is an airline company which has sold 301 tickets for a flight on an airplane of 300 seats. For each seat occupied the company gains 100 utility units, but if all 301 persons show up, then the company loses 1000 utility units. The agent may consistently express $\{D(\text{show-up}(1)), \dots, D(\text{show-up}(301)), D(\neg(\text{show-up}(1) \wedge \dots \wedge \text{show-up}(301)))\}$, because, individually, passenger $\#i$ showing up makes the expected utility of the company increase (slightly), due to the fact that it is very unlikely that all passengers show up.

3.4 Normality and update

There are several perspectives on normality. One consists in viewing normality as distance to the current situation. When an agent figures out a “normal A -world” he often figures out the closest A -world to the actual world. This is in accordance with the principle used for evaluating counterfactuals. Update could come very intuitively into this framework.

The normality ordering is then defined by the proximity to the current world, which is defined by a faithful proximity relation in the sense of [Katsuno and Mendelzon, 1991], i.e., a collection of weak orders $\{\leq_w, w \in W\}$, where faithfulness is the condition: $w <_w w'$ for all $w' \neq w$; $w_1 \leq_w w_2$ means w_1 is closer to w than w_2 . We simply have $w_1 \geq_N w_2$ iff $w_1 \leq_w w_2$, and the set of most normal worlds (i.e., closest worlds to w) satisfying A is the *update* of w , denoted by $w \diamond A$. The simplest and most frequent choices are: $w_1 \leq_w w_2$ iff $\text{Diff}(w, w_1) \subseteq \text{Diff}(w, w_2)$ and $w_1 \leq_w w_2$ iff $|\text{Diff}(w, w_1)| \leq |\text{Diff}(w, w_2)|$, where $\text{Diff}(w, w')$ is the set of variables assigned a different value by w and w' .

Let us consider example 2.4 but with a slightly different interpretation. We do not have to suppose here that flights are normally fully booked. We just assume that *in the current situation, the agent does not have any airplane ticket*.

Example 3.6 (Airplane ticket, continued)

$DS = \{I(r), I(a), \neg I(r \wedge a), \neg r \wedge \neg a\}$

1. *I desire to have an airplane ticket to Rome;*
2. *I desire to have an airplane ticket to Amsterdam;*
3. *I do not desire to have both an airplane ticket to Rome and an airplane ticket to Amsterdam;*
4. *In the current situation I do not have any airplane ticket.*

If the proximity relation is such that (r, a) is closer to $(\neg r, a)$ and $(r, \neg a)$ than to $(\neg r, \neg a)$, and equally close to $(\neg r, a)$ and $(r, \neg a)$, then we get the following normality ordering: $(\neg r, \neg a) >_N (\neg r, a) \sim_N (r, \neg a) >_N (r, a)$, because the initial situation is $(\neg r, \neg a)$. Normality reflects proximity to the initial situation.

Of course, this assumes that a preliminary step has been done so as to translate the proximity to the current world into a normality ordering; this issue has been considered several times in the literature. See for instance [Grahne, 1991] for the intertranslation between update and conditional logics, and [Herzig, 1998] for a review of logics for belief update.

4 Related research

4.1 Two tasks in qualitative decision theory

Qualitative decision theory aims at developing mainly non-numerical – and therefore non-probabilistic – normative frameworks for decision making under uncertainty, e.g. looking for minimal sets of minimal behavioral properties or axioms of a rational agent that correspond to a given action selection criterion. Most approaches, e.g., [Brafman and Tenenhardt, 1996; Lehmann, 2001; Dubois *et al.*, 2002], use ordinal structures for preference and uncertainty.

In a decision-theoretic context we can distinguish at least the task of interpreting desires, which aims at reasoning about the mental state of the agent (what he likes and what he believes) and the task of selecting an action that *uses* the possible mental states induced by the upstream task, like qualitative analogs of maximum expected utility. Both tasks are complementary. The logic developed in this paper does not investigate criteria for action selection and therefore it is *not really a new approach to qualitative decision theory*. Our logic aims at *interpreting desires* as they can be expressed by agents, for instance in an interactive elicitation process. Our logic can infer some information about the normality ordering or the preference ordering of an agent’s mental state but cannot predict which action he will perform.

There are several ways in which our logic can be extended with action selection. In example 2.2, when a model contains both r and $\neg r$ normal worlds, $I(\neg u)$ merely expresses that preferred worlds satisfy $\neg u$ while the selection of action *take-umbrella* may well reflect that *the worst among the most normal effects of this action* are preferred to the worst among the most normal effects of *leave-umbrella*. The latter criterion is used by [Brafman and Tenenhardt, 1997] who model agents as pessimistic decision makers. Noticeably, it is similar to our interpretation of desires, except that worst states have to be focused on instead of best states. It would be worth extending our logical framework with a modality $\mathbf{PA}(\cdot)$: the preference of taking action α over action β would be expressed by $\mathbf{PA}(do(\alpha)|do(\alpha) \vee do(\beta))$, expressing that the worst normal effects of action α are preferred to the worst normal effects of action β . This modality would have the same properties as the $\mathbf{D}(\cdot)$ modality, except that the preference order has to be reversed: if $M = \langle W, \geq_P, \geq_N, val \rangle$ and $M' = \langle W, reverse(\geq_P), \geq_N, val \rangle$ then $M \models \mathbf{PA}(B|A)$ iff $M' \models \mathbf{D}(B|A)$. This extension is left for further research.

4.2 Formalisms of desires

There are several other formalisms that represent desires and pure preference, especially *ceteris paribus constraints* [Doyle and Wellman, 1991; Boutilier *et al.*, 1999]. However, this framework deals with pure preference only and not with uncertainty and normality, except maybe a preliminary attempt in [Tan and Pearl, 1994]. The combination of *ceteris paribus* and normality is an issue for further research.

Thomason’s framework [Thomason, 2000] builds on Reiter’s default logic and deals with both normality and preference defaults, but with a procedural strategy which departs from our completely semantical interpretation: goals are derived by first closing the facts under beliefs defaults, and thereafter under desire defaults. The same mechanism is used

in BOID architectures [Broersen *et al.*, 2002]. The goals are thus restricted to the most normal state.

Finally, decision-theoretic defaults [Brafman and Friedman, 2001; Poole, 1992] also have a semantics based on normality and preference. For example, Brafman and Friedman give a detailed analysis of the use of expected utility, they explain a drawback of straightforward expected utility, and they introduce defaults based on gain of utility $G(A)$. Moreover, they define weak and strong notions of defaults, where the first does not satisfy right weakening, because $G(A)$ is not informative about the behavior on subsets of A . This seems related to what we here called higher-order nonmonotonicity.

5 Summary

The research question of this paper is how desires can be formalized with a realistic interaction between preference and normality. We start with Boutilier's logic for QDT, in which preference and normality have been combined in a notion of ideal goal. However, there is only weak interaction, such that the dog and fence example as well as the airplane ticket example cannot be represented in a consistent way. We therefore interpret desires in a different way: "in context A , I desire B " is interpreted as "the best among the most normal $A \wedge B$ worlds are preferred to the most normal $A \wedge \neg B$ worlds".

We study various formal properties of these desires. We show that our $D(A|B)$ does not imply Boutilier's $I(A|B)$ or $IG(A|B)$, nor vice versa. We show that they do not satisfy weakening nor the conjunction rule. We show that these properties and many others hold under conditions of normality, and we show that some properties like transitivity and cumulativity hold unconditionally. We illustrate the expressive power on several classes of examples including the dog and fence and airline ticket examples, and we illustrate how update can be introduced in the framework. Finally we position the desires with respect to previous work in qualitative decision theory, where we mention as subjects of further research an extension with an action selection criterion, and the introduction of normality in *ceteris paribus* preferences.

References

- [Bacchus and Grove, 1996] F. Bacchus and A.J. Grove. Utility independence in a qualitative decision theory. In *Proceedings of KR'96*, pages 542–552, 1996.
- [Boutilier *et al.*, 1999] C. Boutilier, R. Brafman, H. Hoos, and D. Poole. Reasoning with conditional *ceteris paribus* statements. In *Proceedings of UAI'99*, pages 71–80, 1999.
- [Boutilier, 1994] C. Boutilier. Towards a logic for qualitative decision theory. In *Proceedings of KR'94*, pages 75–86, 1994.
- [Brafman and Friedman, 2001] R.I. Brafman and N. Friedman. On decision-theoretic foundations for defaults. *Artificial Intelligence*, 133:1–33, 2001.
- [Brafman and Tennenholtz, 1996] R. Brafman and M. Tennenholtz. On the foundations of qualitative decision theory. In *Proceedings of AAAI'96*, pages 1291–1296, 1996.
- [Brafman and Tennenholtz, 1997] R. Brafman and M. Tennenholtz. Modeling agents as qualitative decision makers. *Artificial Intelligence*, 94:217–268, 1997.
- [Broersen *et al.*, 2002] J. Broersen, M. Dastani, J. Hulstijn, and L. van der Torre. Goal generation in the BOID architecture. *Cognitive Science Quarterly*, 2(3-4):428–447, 2002.
- [Cohen and H.Levesque, 1990] Cohen and H.Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.
- [Doyle and Thomason, 1999] J. Doyle and R. Thomason. Background to qualitative decision theory. *AI magazine*, 20:2:55–68, 1999.
- [Doyle and Wellman, 1991] J. Doyle and M. P. Wellman. Preferential semantics for goals. In *Proceedings of AAAI'91*, pages 698–703, 1991.
- [Doyle, 1991] J. Doyle. Rationality and its rules in reasoning (extended abstract). In *Proceedings of AAAI'91*, pages 1093–1100, 1991.
- [Dubois *et al.*, 1998] D. Dubois, H. Prade, and R. Sabbadin. Qualitative decision theory with Sugeno integrals. In *Proceedings of UAI'98*, pages 121–128, 1998.
- [Dubois *et al.*, 2002] D. Dubois, H. Fargier, and P. Perny. On the limits of ordinality in decision making. In *Proceedings of KR'02*, pages 133–144, 2002.
- [Grahne, 1991] G. Grahne. Updates and counterfactuals. In *Proceedings of KR'91*, pages 269–276, 1991.
- [Herzig, 1998] A. Herzig. *Handbook of defeasible reasoning and uncertainty management*, volume 3, chapter Logics for belief base updating, pages 189–231. Kluwer Academic Publishers, 1998.
- [Katsuno and Mendelzon, 1991] H. Katsuno and A. Mendelzon. Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52:263–294, 1991.
- [Lehmann, 2001] D. Lehmann. Expected qualitative utility maximization. *Games and Economic Behavior*, 35(1-2):54–79, 2001.
- [Pearl, 1993] J. Pearl. From conditional ought to qualitative decision theory. In *Proceedings of the UAI'93*, pages 12–20, 1993.
- [Poole, 1992] D. Poole. Decision-theoretic defaults. In *Proceedings of the Ninth Biennial Canadian Artificial Intelligence Conference*, pages 190–197, 1992.
- [Rao and Georgeff, 1991] A. Rao and M. Georgeff. Modeling rational agents within a BDI architecture. In *Proceedings of KR'91*, pages 473–484, 1991.
- [Tan and Pearl, 1994] S.W. Tan and J. Pearl. Specification and evaluation of preferences for planning under uncertainty. In *Proceedings of KR'94*, 1994.
- [Thomason, 2000] R. Thomason. Desires and defaults: a framework for planning with inferred goals. In *Proceedings of KR'00*, pages 702–713, 2000.
- [van der Torre and Tan, 1997] L. van der Torre and Y. Tan. The many faces of defeasibility in defeasible deontic logic. In D. Nute, editor, *Defeasible Deontic Logic*, volume 263 of *Synthese Library*, pages 79–121. Kluwer, 1997.

Constitutive Rules for Agent Communication Languages

Jeremy Pitt

Intelligent & Interactive Systems Group, Department of Electrical & Electronic Engineering
South Kensington Campus, Exhibition Road, Imperial College London, SW7 2BT, England
Email: j.pitt@imperial.ac.uk URL: <http://www.iis.ee.imperial.ac.uk/~j.pitt>

Abstract

We follow Searle's contention that speaking a (natural) language is to engage in a rule-governed form of behaviour, and that those rules are conventional (institutional) rather than natural or physical. We show how this analysis can also be used to specify rules of interaction for systems of electronic agents communicating with an artificial language. We conclude that using constitutive rules to define the semantics of an agent communication language not only distinguishes agent communication from method invocation, but also offers significant computational advantages over using intentional states.

1 Introduction

We maintain that there is a distinct conceptual and functional difference between communication using an ACL (Agent Communication Language) and communication using an API (Application Programmer Interface). Method invocation (via an API) is essentially *perlocutionary*, that is, the 'speech act' (the method call) is completely definable in terms of its perlocutionary effect, i.e. the further consequences or effects on the receiving (remote) object. This property makes method invocation practical for client-server distributed systems, because the execution of the remote method is transparent to the (local) caller – the data and methods of the called object may just as easily have been on the same machine as on another connected to the network. It also explains why the call semantics is so important: the fundamental difference between idempotent and at-most-once call semantics is whether or not the server is maintaining state. The communicating language itself is just a conventional device for securing a natural response.

However, according to Searle's institutional theory of communication [Searle, 1969], speaking a language is to engage in a rule-governed form of behaviour. The semantic structure of the language is given by a conventional characterisation of sets of *constitutive* rules, and 'speech acts' are acts typically performed by uttering an expression in accordance with those rules. Thus the basic unit

of communication is the *illocutionary* act, which comprises the type of the act (its illocutionary force: stating, ordering, promising, etc.) and its propositional content. The meaning of each unit is given by the constitutive rules: these are rules that both define the forms of behaviour and determine what that behaviour *counts as*. Searle contended that it was not possible to reduce the analysis of illocutionary acts to perlocutionary effects. Therefore there *is* a significant difference between agent communication and object invocation: the aim of this paper is to make that difference clear and suggest how to leverage it for computational advantage.

In [Searle, 1969], the characterisation of the constitutive rules (in particular the preparatory and sincerity conditions) was expressed in terms of intentional states. While justifiable for natural language communication between humans, we believe that this is inappropriate for open systems of electronic agents and has misdirected standardisation efforts, e.g. FIPA (the Foundation for Intelligent Physical Agents). However, we believe that the conventional and institutional nature of communication can still be used to specify constitutive rules of interaction for such artificial systems.

The argument advanced in this paper to support that belief is as follows. Section 2 gives some preliminaries, introducing our methodology and notation. Section 3 gives some illustrative examples to motivate the specification, in Section 4, of some basic 'standard' communicative acts, and in Section 5 of a richer form of communication for conversations. Section 6 concludes with a summary, a brief review of further and related work, and draws some conclusions. In particular, the emphasis on the institutional perspective of Speech Act theory, rather than on agents' beliefs, desires, or intentions, indicates that a new paradigm in designing agent communication languages offers significant computational advantages.

2 Preliminaries

In this section, we briefly describe our methodological motivation for applying the ideas of Speech Act Theory to agent communication. We then describe the notation used in the rest of the paper, in particular, the abstract representation of agents and normative relations (in particular institutional power, permission and obligation).

2.1 Methodology

We operate on the assumption that there are aspects of human intelligence and society than can inform algorithms, architectures and data structures for the design and implementation of ‘better’ solutions to software engineering problems. The process begins with the analysis of human intelligence and the expression of a theory (the process of theory formation in philosophy, psychology, linguistics, etc. We don’t presume to undertake this step ourselves). The next step is a process we might call reification: the specification of that theory in a formal language. There are two problems with this process: firstly, information loss, as parts of the theory which are intractable are eliminated; and secondly, complexity gain, as there may be theoretical assumptions which are easy to state but much harder to formalise. As a consequence, the reified theory, or specification, is *not* a direct representation or literal specification of the original: instead it has been inspired by that theory.

The process is marked in this paper. We have been analysing both Speech Act Theory [Searle, 1969], Institutional Power [Jones and Sergot, 1996], and Conventional Signalling systems [Jones, 2003]. The resulting specification of – or rather, proposal for a way of specifying – an agent communication language owes something to both accounts of communication (in particular, we will refer to signals and agent’s signalling actions, rather than speech acts) but also looks to the extant computer implementations described in [Artikis *et al.*, 2002]. (Such implementation of the specification is step three of our methodology, while step four is to embed that implementation back in the human context in which it originated.)

2.2 Notation: Agent Description

We assume a domain with distinguished agents a, b , etc., facts (propositions) ϕ , and actions (procedures) π . An agent is a triple comprising (K, G, P) , respectively the agent’s knowledge base, goal base, and planned actions. For an individual agent a , we write:

Δ_a	represents the program state of agent a , which encapsulates the agent’s deliberative state;
$\Delta_a \vdash \phi$	a ’s current program state (knowledge base) proves ϕ ($\phi \in K$);
$\Delta_a \rightsquigarrow \phi$	a ’s current program state plans that in some future program state $\Delta_a \vdash \phi$ (currently $\phi \in G$);
$\Delta_a \mapsto \pi$	a ’s program state plans action π (not necessarily that a plans to do π itself) ($\pi \in P$);
$\tau(\Delta_a, \pi) = \Delta'_a$	denotes a transition of a ’s program state as a consequence of action π being performed.

Intuitively, $\Delta_a \vdash \phi$ can be thought of, if it is preferred, as agent a believes ϕ is true). We write $\Delta_a \vdash (\Delta_b \vdash \phi)$ to denote an actual ‘belief’ of a itself concerning the program state of b , which can be true (in a) independently of whether $\Delta_b \vdash \phi$ or $\Delta_b \vdash \neg\phi$. $\Delta_a \rightsquigarrow \phi$ is an abstraction

covering a future state of a , which may be thought of as a goal to achieve ϕ (say), and $\Delta_a \mapsto \pi$ is an abstraction for a plan to execute action π , either to be done itself or by another agent.

Note that, in a transition $\tau(\Delta_a, \pi)$ to Δ'_a , if a performed π , we require that $\Delta_a \mapsto \pi$ and would normally expect that $\Delta'_a \not\vdash \pi$ (i.e. we are not concerned with ‘accidents’: we are concerned with planned, purposeful action). If $\Delta_a \vdash \neg\phi$ and after (doing) π , ϕ is the case, we assume there is some process of belief revision which ensures that after the transition $\tau(\Delta_a, \pi)$, $\Delta'_a \vdash \phi$ and not $\Delta'_a \vdash \neg\phi$, i.e. that Δ'_a is *internally* consistent.

2.3 Notation: ‘Logical’ Description

The characterisation of the constitutive rules for an agent communication language requires the representation of action, ‘counts as’ and institutional power [Jones and Sergot, 1996], and deontic relations. The notation used is as follows.

For an agent’s actions, both signalling acts and (physical) acts on the environment, a relativised (to agents) $\mathcal{E}$ operator is used (note that π here is the proposition that π happened):

$\mathcal{E}_a\phi$	a sees to it that ϕ is true
$\mathcal{E}_a\pi$	a sees to it that π is performed

For reasons illustrated in the first example of the next section, we also introduce a second relativised action operator $\mathcal{C}$, for the idea of ‘capability to verify’, with intuitive reading (we consider the formal reading later):

$\mathcal{C}_a\phi$	means a is able to demonstrate (give evidence) that ϕ (is true)
$\mathcal{C}_a\pi$	means a is able to demonstrate physical capability to perform π

Central to the [Jones, 2003] analysis of conventional signalling systems and the [Jones and Sergot, 1996] analysis of institutional power is the idea of *counts as*. The idea here is that one agent seeing to it that ϕ , say, can count as, in certain contexts, as another agent – or even the institution itself – seeing to it that ψ . This notion was formalised in [Jones and Sergot, 1996] with a relativised (to institutions) conditional connective $\Rightarrow$:

$$\mathcal{E}_a\phi \Rightarrow_I \mathcal{E}_I\psi$$

Such a formula is called an *institutional power*, whereby a seeing to it that ϕ counts as (just as if) I sees to it that ψ . This is used to formalise Searle’s constitutive rules of the form “ X counts as Y in context C ”, where Searle’s context C will here be denoted by some institution I . We write:

$${}_I\mathbf{Pow}_{a\rho} \stackrel{\text{def}}{=} \mathcal{E}_a\phi \Rightarrow_I \mathcal{E}_I\psi$$

to identify a ‘named’ power ρ denoting a specific counts as formula.

An agent a ’s permissions and obligations with respect to an institution I are represented by:

${}_I\mathbf{Per}_a\mathcal{E}_a\pi$	a is permitted to perform π in (by) I
${}_I\mathbf{Per}_a\mathcal{E}_a\phi$	a is permitted to make ϕ true in (by) I
${}_I\mathbf{Obl}_a\mathcal{E}_a\pi$	a is obliged to (by) I to perform π
${}_I\mathbf{Obl}_a\mathcal{E}_a\phi$	a is obliged to (by) I that ϕ be true

3 Two Illustrative Examples

3.1 Example 1: TCP Slowstart

The standard TCP specification requires implementation of an algorithm called “slow start”, documented in IETF RFC2001 (cf. [Stevens, 1996]). The specification of the algorithm mandates that, on the sender’s side, the first transmission consists of just one packet. If this is acknowledged successfully, then the sender can transmit two packets. The sender then continues to increase the number of packets transmitted until the network capacity is reached (packets are not acknowledged successfully), and then throttles back. This ensures that no TCP sender overwhelms a network.

Let w be a Windows NT server, and let ϕ be the statement “ I (w) implement TCP slow start algorithm as specified in IETF RFC2001”. Let σ be the signalling system (i.e. agent communication language) in which the signal ς denotes ϕ . Assume σ is used for communication by institution IETF, represented by I , so that $\mathcal{E}_a\varsigma$ is the action of transmitting the signal denoting ϕ .

Now, suppose we want to allow agents, in particular servers (TCP senders), to assert or inform other agents that they implement the IETF (institution I) standard specifications. We also want I to be able to inspect and verify that servers do indeed implement the standard specifications correctly.

So, we can write $\Delta_w \vdash \phi$, meaning that, subjectively, from w ’s perspective, ϕ is true (w believes ϕ). Suppose then that w makes the assertion, in σ that ϕ . As per Searle (“Counts as an undertaking that $[\phi]$ represents an actual state of affairs” ([Searle, 1969]: p.66)) we want this to count as a commitment, in the context of the institution I , that it implements the slow start algorithm correctly. We interpret commitment here as an obligation to be able to verify that ϕ .

Now, to capture the idea of an assertion as a commitment, the institution I has the following as one of its constitutive rules, which also apply to *any* other server:

$$\mathcal{E}_w\varsigma \Rightarrow_I \mathcal{E}_I(I\mathbf{Obl}_w\mathcal{C}_w\phi)$$

Thus the assertion, in I , that the server w implements the slow start algorithm, counts as an obligation, from the server to the institution, that this is a true statement.

However, and more importantly, we also want to be able to make judgements of the form either:

$$\Delta_I \vdash \phi \quad \text{or} \quad \Delta_I \not\vdash \phi$$

to say that according to the institution, ϕ is objectively (at least according to the institution) true (or not).

Certainly this demands that $\Delta_w \vdash \phi$, as a required state, but also that w be capable of demonstrating that $\Delta_w \vdash \phi$. As it turns out, because of a widespread bug in the slow start implementation in Windows NT servers, ϕ is not a true (objectively true) statement [Stevens, 1996]. Being unable to verify (if demanded to do so) that ϕ , w would be in violation of its obligation if it made the assertion. (Note we have set aside the practical problem of a program verifying properties (correctness) of itself.)

3.2 Example 2: Football (Soccer)

This kind of example is common in the literature but worth examining again. Consider the institution F of football (soccer). We define a “goal” is scored when the ball has crossed a painted white line between two uprights and under a crossbar. Then, we can examine the situation at any given instant and determine whether or not a *goal* (according to F) has been scored. (In Searle’s terms, the fact that a ball has crossed a line is a brute fact, the fact that by crossing the line between some sticks a *goal* has been scored, is an institutional fact.)

In certain games, a referee is given the power to examine the situation and decide whether or not it fits the definition of a “goal”. If the referee signals that this is the case, then it is a *goal* (in the sense that the score of one team is incremented, etc.). Although a fairly coarse over-simplification, we could say that the referee is empowered by F to see to it that *goals* are scored. However, the referee is not permitted to signal “goal” (cannot exercise the institutional power) unless the world-to-word fit is such that the situation in the world matches the definition of the word. So if one of the players sees to it that the conditions for a *goal* are satisfied (implicitly, according to the rules of football), then this counts as, in F , as initiating the referee’s permission to signal “goal”.

Let us assume that the players *are* permitted to signal “goal” arbitrarily (of course this is not strictly the case), but because they are not empowered to create *goals*, if they do signal “goal”, this does not count as a *goal* (in effect, all they can do is ‘claim’).

So let us have agent r who has the power of referee (i.e. agent r occupies the role of referee, which is granted certain powers by institution F). Let ς_g be the action of transmitting a signal in the signalling system (ACL) used by F , which denotes “goal” (e.g., whistling, saying “goal”, etc.). Then, for the institution F , referee agent r , and player agents p , we have :

$$\begin{aligned} {}_F\mathbf{Pow}_r\text{referee} &\stackrel{\text{def}}{=} \{\mathcal{E}_r\varsigma_g \Rightarrow_F \mathcal{E}_F\text{goal}\} \\ \mathcal{E}_p\text{“goal”} &\Rightarrow_F \mathcal{E}_F({}_F\mathbf{Per}_r\mathcal{E}_r\varsigma_g) \\ \forall p.F\mathbf{Per}_p\mathcal{E}_p\varsigma_g & \end{aligned}$$

A situation arose in the 1966 World Cup Final between England and Germany, when the England players were claiming a goal had been scored. The England players were permitted to signal “goal”, but not being empowered, these signals did not count as *goals*. The referee was so empowered, so when he signalled “goal”, it did count as a *goal* – even though a “goal” had *not* been scored. The problem, according to the characterisation above, was that the referee was exercising a power that in fact he was not permitted to. At this point, the referee should have been sanctioned according to the rules of F : but this too is a subject for further work. However, as we see in Section 5, this is the kind of situation we expect to encounter in future multi-agents systems, where agents have powers to see to it that institutional facts are true, but have constraints on their permissions or occasions when to exercise that power.

4 Single Signalling Acts

The two illustrative examples of the previous section indicate typical examples of the kind of meaning we seek for agent's illocutionary acts. In the first example, it was that the assertion counted as an undertaking that the propositional content was true (and that the asserter could verify that it was true); in the second example, that the declaration counted as the institution seeing to that the corresponding institutional fact was true.

Motivated by this, in this section, we specify a set of constitutive rules for single, 'one off' signals, i.e. signalling actions not anticipating a reply. The specific context for this argument is as follows. For "an agent communicates by message passing", i.e. a communicative action performed by an agent, we are assuming a computational model involving the point-to-point transmission (from agent a to agent b) of 'information structures' representing these messages. 'Saying' and 'hearing' are the results of, respectively, writing to and reading from some channel (e.g. a TCP/IP socket connection), or the consequences of remote method invocation (RMI) using constitutive rules rather than a conventional device.

For the operational context, we mean an open system as indicated in [Artikis *et al.*, 2002], whereby: no single agent has universal knowledge of the entire system, agents may have conflicting goals, internal architectures are not known, local states are individually consistent but may be globally inconsistent, and there is no centralized authority or control (of knowledge and behaviour). On the issue of decentralization, we take the position of local autonomy over goals, decisions and state, but that all communication is institutional [Searle, 1999].

Assume for the application that the agents are engaged in information trading or web services. We assume that the application will include finding and requesting services, and finding and supplying information. Thus there will be some individual message exchanges (assertives, directives, etc.) and some structured exchanges (e.g. contract nets and auctions). For each individual signalling act, we give the required form (of propositional content and signalling content), the conventional interpretation of the signal, and the institutional powers (if any) associated with the signal itself. Note that this is an external specification: so the conventional interpretation indicates that, in the context of this particular institution, if an agent wants to convey a certain meaning, then (by convention) it uses a particular signal; and if an agent witnesses a particular signal, then (by convention) a certain meaning is conveyed. We consider four of the five types of illocutionary point identified in [Searle, 1999]: assertive (assert and inform), directive (command and request), commissive and declarative.

1. Assertive (assert)	
<i>propositional content</i>	proposition ϕ
<i>signal</i> ς	$\text{assert}(\phi)$
<i>conventional interpretation</i>	$\Delta_a \rightsquigarrow (\Delta_b \vdash \phi)$
<i>institutional condition</i>	$\mathcal{E}_a \varsigma \Rightarrow_I \mathcal{E}_I(I \mathbf{Obl}_a \mathcal{C}_a \phi)$

2. Assertive (inform)	
<i>propositional content</i>	proposition ϕ
<i>signal</i> ς	$\text{inform}(\phi)$
<i>conventional interpretation</i>	$\Delta_a \rightsquigarrow (\Delta_b \vdash \phi)$
<i>institutional condition</i>	none
3. Directive (request)	
<i>propositional content</i>	action π
<i>signal</i> ς	$\text{request}(\pi)$
<i>conventional interpretation</i>	$\Delta_a \rightsquigarrow (\Delta_b \mapsto \pi)$
<i>institutional condition</i>	none
4. Directive (command)	
<i>propositional content</i>	action π
<i>signal</i> ς	$\text{command}(\pi)$
<i>conventional interpretation</i>	$\Delta_a \rightsquigarrow (\Delta_b \mapsto \pi)$
<i>institutional condition</i>	$\mathcal{E}_a \varsigma \Rightarrow_I \mathcal{E}_I(I \mathbf{Obl}_b \mathcal{E}_b \pi)$
5. Commissive (promise)	
<i>propositional content</i>	action π
<i>signal</i> ς	$\text{promise}(\pi)$
<i>conventional interpretation</i>	$\Delta_a \rightsquigarrow (\Delta_b \vdash (\Delta_a \mapsto \pi))$
<i>institutional condition</i>	$\mathcal{E}_a \varsigma \Rightarrow_I \mathcal{E}_I(I \mathbf{Obl}_a \mathcal{E}_a \pi)$
6. Declarative (declare)	
<i>propositional content</i>	proposition ϕ
<i>signal</i> ς	$\text{declare}(\phi)$
<i>conventional interpretation</i>	$\Delta_a \rightsquigarrow (\Delta_b \vdash \phi)$
<i>institutional condition</i>	$\mathcal{E}_a \varsigma \Rightarrow_I \mathcal{E}_I \phi$

Note that there might be some minor objections to the characterisation of the constitutive rules for the given illocutionary act types. However, recall that this is an *artificial* system of communication, that is being *designed*. It is not intended to be a formal characterisation of a pre-existing system of communication, i.e. natural language, although it draws its inspiration from that source. If in turn the inspiration is not respecting the intuition, then it is easy enough to substitute *signal_act_1* for assert, *signal_act_2* for inform, and so on, as the software will process the messages in just the same way and according to the same constitutive rules.

Searle's [Searle, 1969] formulation of rules in speech act theory included what he called preparatory conditions and sincerity conditions. Preparatory conditions were extant circumstances required for a speech act to be a valid act of the associated illocutionary type; the sincerity condition was a 'psychological' expression associated with the speech act.

Our agents do not have psychological states, but given the notation of Section 2 it is not uninteresting to consider putative sincerity conditions expressed in the signals defined above. For example, if we added to *inform* the preparatory condition $\neg(\Delta_a \vdash (\Delta_b \vdash \phi))$, and the sincerity condition $\Delta_a \vdash \phi$, then these are effectively FIPA inform Feasibility Preconditions FP1 and FP2 [FIPA, 1997], while the conventional interpretation is essentially the intended RE (Rational Effect). Thus it is straightforward, within the framework of constitutive rules, to get the current FIPA ACL semantics to 'drop out' as a special case.

5 Conversations: An Auction Protocol

In this section, we apply the idea of constitutive rules to a conversation, based on a variant of the English Auction protocol [Venkatraman and Singh, 1999]. One observation from the analysis here is that specification of constitutive rules for an ACL must be sensitive to time and predication, cf. [Artikis *et al.*, 2002; Sergot, 2003].

An informal description of the auction is: the auctioneer opens the auction for goods g at price p ; bidders are empowered to bid for g at price p ; after receiving a bid, the auctioneer is empowered to announce a new price p' , $p' > p$; and if there is no bid at the new price, the auctioneer has the power to accept a bid at price p .

There are also other rules that govern the conduct of an auction, namely that: offering specific goods g for auction implies that the auctioneer either owns or is licensed to sell g ; making a bid of p for p implies that the bidder owns p ; bidding p more than once and bidding a price lower than the announced price p are meaningless actions; and the auctioneer is not empowered to accept two bids for the same g .

One way to formalise this protocol is to specify: the institutional powers of the auctioneer and the bidders; the assertions implicit in making announcements and bids; and the changing permissions of each participant as the protocol progresses which determine whether or not they are allowed to exercise their powers. There are three illocutionary acts: announce, bid and accept, with propositional content goods g (lot number) and price p . We assume, that all communications are broadcast, with the intended recipient b of accept messages explicitly identified in some way.

In general, the assertive force implicit in the illocutionary acts is given by:

$$\mathcal{E}_a \text{announce}(G, P) \Rightarrow_I \mathcal{E}_I(I \text{Obl}_a \mathcal{C}_a \text{own}(G))$$

$$\mathcal{E}_b \text{bid}(G, P) \Rightarrow_I \mathcal{E}_I(I \text{Obl}_b \mathcal{C}_b \text{own}(P))$$

The institutional conditions element of the constitutive rules are given by the following institutional powers. Note that these are unlike some of the institutional powers previously discussed in this paper, for two reasons. Firstly, these are ‘power schema’, in the sense that the actions of other agents will instantiate specific instances of each schema; and secondly, because they are transient rather than permanent, and each specific instance is either initiated or terminated under certain circumstances.

The power of a bidder is given by:

$$I \text{Pow}_b \text{bid}(G, P) \stackrel{\text{def}}{=} \mathcal{E}_b \text{bid}(G, P) \Rightarrow_I \mathcal{E}_I \text{bid}(b, G, P)$$

while the auctioneer has two powers: firstly the power to make announcements, by announcing an auction, and secondly the power to make contracts, by accepting a bid:

$$I \text{Pow}_a \text{announce}(G, P) \stackrel{\text{def}}{=} \mathcal{E}_a \text{announce}(G, P) \Rightarrow_I \mathcal{E}_I \text{announcement}(a, G, P)$$

$$I \text{Pow}_a \text{accept}(B, G, P) \stackrel{\text{def}}{=} \mathcal{E}_a \text{accept}(B, G, P) \Rightarrow_I \mathcal{E}_I \text{contract}(a, B, G, P)$$

Now the way it works is that if an agent has the power and performs the action, then it will *count as* seeing to it that the corresponding institutional fact is true. The institutional fact in turn initiates or terminate other powers (so each agent in effect has the power to empower other agents). If, initially, $I \text{Pow}_a \text{announce}(g, p)$, then:

signal	powers (initiated/terminated)	
a signals $\text{announce}(g, p)$ results in ...		
... initiating	$I \text{Pow}_b \text{bid}(g, p)$	for all b
... terminating	$I \text{Pow}_a \text{announce}(g, p)$	
b signals $\text{bid}(g, p)$ results in ...		
... initiating	$I \text{Pow}_a \text{announce}(g, p')$	$p' > p$
... terminating	$I \text{Pow}_{b'} \text{bid}(g, p)$	for all b'
... terminating	$I \text{Pow}_a \text{accept}(b'', g, p)$	some b''
... initiating	$I \text{Pow}_a \text{accept}(b, g, p)$	
a signals $\text{accept}(b, g, p)$ results in ...		
... terminating	$I \text{Pow}_a \text{accept}(b, g, p)$	
... terminating	$I \text{Pow}_{b'} \text{bid}(g, p')$	

Note [Artikis *et al.*, 2002] formalised a specification of the contract-net protocol in terms of which powers, etc., were initiated and which were terminated by specific actions (which allowed for an event calculus specification and direct implementation of the protocol). [Sergot, 2003] also contains an analysis of an auction protocol.

What’s missing from the formalisation of conversational illocutionary acts, as opposed to the signalling acts of the previous section, is the conventional interpretation (that these acts count as attempts to affect computational states in some way). It is not entirely clear what these should be (e.g. for announce, to inform b that g is for sale, to induce b to make a bid, or to indicate its intention to sell g once a buyer is found); or indeed that such rules are adding anything useful in this context. We leave this for further investigation.

In addition, we could specify that agents are only permitted to exercise their power, as an auctioneer or a bidder, if they either own the goods g or the bidding price p respectively. In this way, we could circumvent the implicit assertive force in the announce and bid signals, by requiring that agents seek permission to sell goods or offer bids, and are subject to sanction if they create contracts without permission, which they cannot honour.

6 Summary and Conclusions

Searle [Searle, 1969] defined the semantics of illocutionary acts in terms of the conventional realization of series of sets of constitutive rules, and that a meaningful speech act was performed by uttering an expression in accordance with those rules. The constitutive rule sets proposed ([Searle, 1969]:p66-67) contained several conditions, including the propositional content, preparatory, sincerity and essential conditions. However, a number of these rules were concerned with ‘psychological’ states, and the attempted formalization of such rules in multi-modal BDI logics are far from computationally tractable.

In this paper, we have proposed to use Searle’s theory of constitutive rules to define a semantics of illocutionary acts (signals) for an agent communication language.

The set of conditions, for a wide series of act types, includes the propositional content and form of a signalling act, the conventional interpretation (of intended perlocutionary effect), the institutional conditions (expressed as a counts as formula) requiring institutional powers, and (in conversations) what deontic states are initiated or terminated by the performance of the signalling act.

There is, however, more work that is required. We have characterised the formalism used in Section 2 as a ‘notation’. Ideally, the agent abstraction would define an operational model, which would be directly executable, and the ‘logical’ representation would be complete. For an executable operational model, we need an operational semantics, and for this purpose we are investigating labelled transition systems. A compiler for such rules, with an explicit link to (an implementation of) the agent abstraction, could then be considered.

For the logical model, we require (at least) a semantics of the $\mathcal{C}$ operator. We recognize this is problematic, but for the kind of applications in which we are interested (where assertions of the forms “I have this right”, “I represent this agency”, “I can perform this service” are anticipated) we feel something of this kind is required. We have interpreted Searle’s meaning of an assertion (“commitment to the truth of ϕ ”) as an obligation to be able to verify ϕ , and this is not a matter of merely seeing to it that ϕ ($\mathcal{E}_a\phi$). However, the work-around suggested in the previous section could eliminate the $\mathcal{C}$ operator.

Moreover, there are many open questions, some of which have already been indicated. The nature of the relationship between electronic institutions and deontic states of agents needs to be further elaborated, as this is central to definition of the communication model. Many problems in Section 5 stemmed from trying to analyse dynamic phenomena in an essentially static formalism, so an explicit treatment and representation of time, linking to [Artikis *et al.*, 2002; Sergot, 2003], is required. The representation of contracts, from where this work originated, also needs to be refined.

Defining ACL semantics based on Speech Act theory was first proposed for both KQML [Labrou and Finin, 1998] and FIPA [FIPA, 1997], and given in terms of an axiomatic characterisation of intentional BDI agents. We would argue that the emphasis on the intentional states rather than the institutional nature of the communication has omitted the most important component of the theory. [Greaves *et al.*, 2001] proposed an alternative model, using the idea of conversation policies, which define fine-grained constraints on ACL usage. Another alternative involves proposals primarily based on the idea of commitments, e.g. [Venkatraman and Singh, 1999; Colombetti, 2001], while Jones [Jones, 2003] proposes a conventional account of natural language speech acts based on the intentional exploitation of signalling systems. Motivated in part by the latter work, we have returned to Speech Act theory and attempted to characterise constitutive rules for a system of communication between ‘autonomous’ agents governed by an electronic institution. Our intuition is that the interaction between

this work and that of conversation policies offers some intriguing potential for further development.

In conclusion, we believe that we have a basis for designing ACLs, and giving a formal characterization of the semantics of communicative acts which is powerful, expressive and flexible. Most importantly, it demonstrates the fundamental conceptual and functional difference between language-based communication and method invocation, and justifies why agent-based systems offer a unique advantage over object-oriented ones.

Acknowledgements

We are grateful to the partners on the ALFEBIITE project (EU IST-1999-10298), especially Marek Sergot, Andrew Jones, Alexander Artikis and Lloyd Kamara; the members of FIPA TC Semantics; and several other commentators for their valuable input and insight.

References

- [Artikis *et al.*, 2002] A. Artikis, J. Pitt, and M. Sergot. Animated specifications of computational societies. In C. Castelfranchi and L. Johnson, eds., *Proceedings AAMAS’02*, pp1053–1062. ACM Press, 2002.
- [Colombetti, 2001] M. Colombetti. Semantic, normative and practical aspects of agent communication. In F. Dignum and M. Greaves, eds., *Issues in Agent Communication*, LNAI1916, pp17–30. Springer, 2001.
- [FIPA, 1997] FIPA. FIPA’97 specification part 2: Agent communication language. FIPA (Foundation for Intelligent Physical Agents), <http://www.fipa.org>, 1997.
- [Greaves *et al.*, 2001] M. Greaves, H. Holmback, and J. Bradshaw. What is a conversation policy. In F. Dignum and M. Greaves, eds., *Issues in Agent Communication*, LNAI1916, pp118–131. Springer, 2001.
- [Jones and Sergot, 1996] A. Jones and M. Sergot. A formal characterisation of institutionalized power. *Journal of the IGPL*, 4(3):429–455, 1996.
- [Jones, 2003] A. Jones. A logical framework. In J. Pitt (ed.): *The Open Agent Society*. Wiley, 2003 (to appear) [m/s at <http://alfebiite.ee.imperial.ac.uk/>].
- [Labrou and Finin, 1998] Y. Labrou and T. Finin. Semantics and conversations for an agent communication language. In M. Huhns and M. Singh, eds., *Readings in Agents*, pp235–242. Morgan Kaufmann, 1998.
- [Searle, 1969] J. Searle. *Speech Acts: An Essay in the Philosophy of Language*. CUP, 1969.
- [Searle, 1999] J. Searle. *Mind, Language and Society: Philosophy in the Real World*. Basic Books, 1999.
- [Sergot, 2003] M. Sergot. The language ($\mathcal{C}$ +)++. In J. Pitt (ed.): *The Open Agent Society*. Wiley, 2003 (to appear) [m/s at <http://alfebiite.ee.imperial.ac.uk/>].
- [Stevens, 1996] W. Richard Stevens. *TCP/IP Illustrated, Volume 3*. Addison-Wesley, 1996.
- [Venkatraman and Singh, 1999] M. Venkatraman and M. Singh. Verifying compliance with commitment protocols: Enabling open web-based multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 2(3):217–236, 1999.

MULTIAGENT SYSTEMS

MULTIAGENT REINFORCEMENT LEARNING
AND POMDPs

Simultaneous Adversarial Multi-Robot Learning

Michael Bowling and Manuela Veloso

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

Abstract

Multi-robot learning faces all of the challenges of robot learning with all of the challenges of multiagent learning. There has been a great deal of recent research on multiagent reinforcement learning in stochastic games, which is the intuitive extension of MDPs to multiple agents. This recent work, although general, has only been applied to small games with at most hundreds of states. On the other hand robot tasks have continuous, and often complex, state and action spaces. Robot learning tasks demand approximation and generalization techniques, which have only received extensive attention in single-agent learning. In this paper we introduce GraWoLF, a general-purpose, scalable, multiagent learning algorithm. It combines gradient-based policy learning techniques with the WoLF (“Win or Learn Fast”) variable learning rate. We apply this algorithm to an adversarial multi-robot task with simultaneous learning. We show results of learning both in simulation and on the real robots. These results demonstrate that GraWoLF can learn successful policies, overcoming the many challenges in multi-robot learning.

1 Introduction

Multi-robot learning is the challenge of learning to act in an environment containing other robots. These other robots, though, have their own goals and may be learning as well. Other adapting robots make the environment no longer stationary, violating the Markov property that traditional single-agent behavior learning relies upon. Multi-robot learning combines all of these multiagent learning challenges with the problems of learning in robots, such as continuous state and action spaces and minimal training data.

A great deal of recent work on multiagent learning has looked at the problem of learning in stochastic games [Littman, 1994; Singh *et al.*, 2000; Bowling and Veloso, 2002a; Greenwald and Hall, 2002]. Stochastic games are a natural extension of Markov decision processes (MDPs) to multiple agents and have been well studied in the field of game theory. The traditional solution concept for the problem of simultaneously finding optimal policies is that of Nash

equilibria. An equilibrium is simply a policy for all of the players where each is playing optimally with respect to the others. This concept is a powerful solution for these games even in a learning context, since no agent could learn a better policy when all the agents are playing an equilibrium.

Multiagent learning in stochastic games, thus far, has only been applied to small games with enumerable state and action spaces. Robot learning tasks though have continuous state and action spaces, and typically with more than just a couple dimensions. Discretizations of this space into an enumerable state set also do not typically perform well. In addition, data is considerably more costly to gather, and millions of training runs to learn policies is not feasible. Typical solutions to this robot learning problem is to use approximation to make the learning tractable and generalization for more efficient use of training experience.

In this paper we introduce a new algorithm, GraWoLF¹, that combines approximation and generalization techniques with the WoLF multiagent learning technique. We show empirical results of applying this algorithm to a problem of simultaneous learning in an adversarial robot task. In Section 2, we give a brief overview of key concepts from multiagent learning along with the formal model of stochastic games. In Section 3, we describe a particular adversarial robot task and its challenges for learning. In Section 4, we present the main components of GraWoLF: policy gradient ascent and the WoLF variable learning rate. In Section 5, we present experimental results of applying this algorithm to our adversarial robot task, and then conclude.

2 Multiagent Learning

Multiagent learning has focused on the game theoretic framework of stochastic games. A *stochastic game* is a tuple $(n, \mathcal{S}, \mathcal{A}_{1..n}, T, R_{1..n})$, where n is the number of agents, $\mathcal{S}$ is a set of states, $\mathcal{A}_i$ is the set of actions available to agent i (and $\mathcal{A}$ is the joint action space $\mathcal{A}_1 \times \dots \times \mathcal{A}_n$), T is a transition function $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$, and R_i is a reward function for the i th agent $\mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$. This looks very similar to the MDP framework except we have multiple agents selecting actions and the next state and rewards depend on the joint action of the agents. Another important difference is that each agent

¹GraWoLF is short for “Gradient-based Win or Learn Fast”, and the ‘a’ has the same sound as in “gradient”.

has its own separate reward function. The goal for each agent is to select actions in order to maximize its discounted future rewards with discount factor γ .

Stochastic games are a very natural extension of MDPs to multiple agents. They are also an extension of matrix games to multiple states. Each state in a stochastic game can be viewed as a matrix game with the payoffs for each joint action determined by the matrices $R_i(s, a)$. After playing the matrix game and receiving their payoffs the players are transitioned to another state (or matrix game) determined by their joint action. We can see that stochastic games then contain both MDPs and matrix games as subsets of the framework.

Stochastic Policies. Unlike in single-agent settings, deterministic policies in multiagent settings can often be exploited by the other agents. Consider the children’s game rock-paper-scissors. If one player were to play any action deterministically, the other player could win every time by selecting the action that defeats it. This fact requires the consideration of mixed strategies and stochastic policies. A stochastic policy, $\pi : \mathcal{S} \rightarrow PD(\mathcal{A}_i)$, is a function that maps states to mixed strategies, which are probability distributions over the player’s actions. We later show that stochastic policies are also useful for gradient-based learning techniques.

Nash Equilibria. Even with the concept of stochastic policies there are still no optimal policies that are independent of the other players’ policies. We can, though, define a notion of best-response. A policy is a *best-response* to the other players’ policies if it is optimal given their policies. The major advancement that has driven much of the development of matrix games, game theory, and even stochastic games is the notion of a best-response equilibrium, or *Nash equilibrium* [Nash, Jr., 1950].

A Nash equilibrium is a collection of strategies for each of the players such that each player’s strategy is a best-response to the other players’ strategies. So, no player can do better by changing strategies given that the other players also don’t change strategies. What makes the notion of equilibrium compelling is that all matrix games and stochastic games have such an equilibrium, possibly having multiple equilibria. Zero-sum, two-player games, like the adversarial task explored in this paper, have a single Nash equilibrium.²

Learning in Stochastic Games. Stochastic games have been the focus of recent research in the area of reinforcement learning. There are two different approaches being explored. The first is that of algorithms that explicitly learn equilibria through experience, independent of the other players’ policy, e.g., [Littman, 1994; Greenwald and Hall, 2002]. These algorithms iteratively estimate value functions, and use them to compute an equilibrium for the game. A second approach is that of best-response learners, e.g., [Claus and Boutilier, 1998; Singh *et al.*, 2000; Bowling and Veloso, 2002a]. These learners explicitly optimize their reward with respect to the

²There can actually be multiple equilibria, but they all have equal payoffs and are interchangeable.

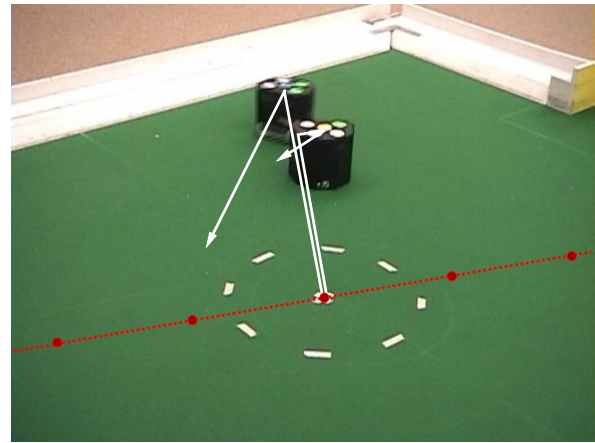


Figure 1: An adversarial robot task. The top robot is trying to get inside the circle while the bottom robot is trying to stop it. The lines show the state and action representation, which is described in Section 4.3.

other players’ possibly changing policies. This approach, too, has a strong connection to equilibria. If these algorithms converge when playing each other, then they must do so to an equilibrium [Bowling and Veloso, 2002a].

Neither of these approaches, though, have been scaled beyond games with a few hundred states. Games with a very large number of states, or games with continuous state spaces, make state enumeration intractable. Since previous algorithms in their stated form require the enumeration of states either for policies or value functions, this is a major limitation. In this paper we examine learning in an adversarial robot task, which can be thought of as a continuous state stochastic game. Specifically, we build on the idea of best-response learners using gradient techniques [Singh *et al.*, 2000; Bowling and Veloso, 2002a]. We first describe our robot task and then describe our algorithm and results.

3 An Adversarial Robot Task

Consider the adversarial robot task shown in Figure 1. The robot at the top of the figure, the attacker, is trying to reach the circle in the center of the field, while the robot closer to the circle, the defender, is trying to prevent this from happening. If the attacker reaches the circle, it receives a reward of one and the defender receives a reward of negative one. These are the only rewards in the task. When the attacker reaches the circle or ten seconds elapses, the trial is over and the robots reset to their initial positions, where the attacker is a meter from the circle and the defender half-way between. The robots simultaneously learn in this environment each seeking to maximize its own discounted future reward. For all of our experiments the discount factor used was 0.8 for each full second of delay.

The robots themselves are part of the CMDragons robot soccer team, which competes in the RoboCup small-size league. There are two details about the robot platform that are relevant to this learning task. First, the learning algorithm itself is situated within a large and complex architecture of

existing capability. The team employs a global vision system mounted over the field. This input is processed by an elaborate tracking module that provides accurate positions and velocities of the robots. These positions comprise the input state for the learning algorithm. The team also consists of robust modules for obstacle avoidance and motion control. The actions for the learning algorithm then involve providing target points for the obstacle avoidance module. Situating the learning within the context of this larger architecture focuses the learning. Rather than having the robot learn to solve well understood problems like path planning or object tracking, the learning is directed at the heart of the problem, the multi-robot interaction.

Second, the system control loop that is partially described above has inherent, though small, latency. Specifically, after an observed change in the world 100ms will elapse before the robot's response is executed. This latency is overcome for each robot's own position and velocity by predicting through this latency using knowledge of the past, but not yet executed, actions. Since the actions of the opponent robots are not known, this prediction is not possible for other robots. Latency effectively adds an element of partial observability to the problem, since the agents do not have the complete state of the world, and in fact have separate views of this state. Notice, that this also adds a tactical element to successful policies. A robot can "fake" the opponent robot by changing its direction suddenly, knowing the other robot will not be able to respond to this change for a full latency period.

This task involves numerous challenges for existing multi-agent learning techniques. These challenges include continuous state and action spaces, elements of partial observability due to system latency, and violation of the Markov assumption since many of the system components have memory, e.g., the tracking and the obstacle avoidance modules. In fact, these limitations may even make equilibria cease to exist altogether [Bowling and Veloso, 2002b]. This is a further reason for exploring best-response learning techniques, which don't directly seek to learn an equilibrium. We now present GraWoLF, a best-response learning algorithm that can handle the challenges of multi-robot learning.

4 GraWoLF

GraWoLF, or Gradient-based WoLF, combines two key ideas from reinforcement learning: policy gradient learning and the WoLF variable learning rate. Policy gradient learning is a technique to handle intractable or continuous state spaces. WoLF is a multiagent learning technique that encourages best-response learning algorithms to converge in situations of simultaneous learning. We briefly describe these techniques and how they are combined.

4.1 Policy Gradient Ascent

We use the policy gradient technique presented by Sutton and colleagues [2000]. Specifically, we define a policy as a Gibbs distribution over a linear combination of features of a candidate state and action pair. Let θ be a vector of the policy's parameters and let ϕ_{sa} be an identically sized feature vector for state s and action a , then the Gibbs distribution defines a

stochastic policy according to,

$$\pi_{\theta}(s, a) = \frac{e^{\theta \cdot \phi_{sa}}}{\sum_b e^{\theta \cdot \phi_{sb}}}.$$

Sutton and colleagues' main result was a convergence proof for the following policy iteration rule that updates a policy's parameters in a Gibbs distribution according to,

$$\theta_{k+1} = \theta_k + \delta_k \sum_s d^{\pi_k}(s) \sum_a \phi_{sa} \cdot \pi_{\theta_k}(s, a) f_{\mathbf{w}_k}(s, a) \quad (1)$$

$f_{\mathbf{w}_k}(s, a)$ is an independent approximation of $Q^{\pi_{\theta_k}}(s, a)$ with parameters $\mathbf{w}$, which is the expected value of taking action a from state s and then following the policy π_{θ_k} . For the Gibbs distribution, Sutton and colleagues showed that for convergence this approximation should have the following form,

$$f_{\mathbf{w}_k}(s, a) = \mathbf{w}_k \cdot \left[\phi_{sa} - \sum_b \pi_{\theta_k}(s, b) \phi_{sb} \right]. \quad (2)$$

As they point out, this amounts to $f_{\mathbf{w}}$ being an approximation of the advantage function, $A^{\pi}(s, a) = Q^{\pi}(s, a) - V^{\pi}(s)$, where $V^{\pi}(s)$ is the value of following policy π from state s . It is this advantage function that we estimate and use for gradient ascent.

Using this basic formulation we derive an on-line version of the learning rule, where the policy's weights are updated with each state visited. The summation over states can be removed by updating proportionately to that state's contribution to the policy's overall value. Since we are visiting states on-policy then we only need to weight later states by the discount factor to account for their smaller contribution. If t time has passed since the trial start, this turns Equation 1 into,

$$\theta_{k+1} = \theta_k + \gamma^t \delta_k \sum_a \phi_{sa} \cdot \pi_{\theta_k}(s, a) f_{\mathbf{w}_k}(s, a). \quad (3)$$

Since the whole algorithm is on-line, we do the policy improvement step (updating θ) simultaneously with the value estimation step (updating $\mathbf{w}$). We do value estimation using gradient-descent Sarsa(λ) [Sutton and Barto, 1998] over the same feature space as the policy. This requires maintaining an eligibility trace vector, $\mathbf{e}$. The update rule is then, if at time k the system is in state s and takes action a transitioning to state s' in time t and then taking action a' , we update the trace and the weight vector using,

$$\mathbf{e}_{k+1} = \lambda \gamma^t \mathbf{e}_k + \phi_{sa}, \quad (4)$$

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{e}_{k+1} \alpha_k \begin{pmatrix} r + \gamma^t Q_{\mathbf{w}_k}(s', a') \\ -Q_{\mathbf{w}_k}(s, a) \end{pmatrix}, \quad (5)$$

where λ is the Sarsa parameter and α_k is an appropriately decayed learning rate. The addition of raising γ to the power t allows for actions to take differing amounts of time to execute, as in semi-Markov decision processes [Sutton and Barto, 1998].

The policy improvement step then uses Equation 3 where s is the state of the system at time k and the action-value estimates from Sarsa, $Q_{\mathbf{w}_k}$, are used to compute the advantage term. We then get,

$$f_{\mathbf{w}_k}(s, a) = Q_{\mathbf{w}_k}(s, a) - \sum_a \pi_{\theta_k}(s, a) Q_{\mathbf{w}_k}(s, a). \quad (6)$$

This forms the crux of GraWoLF. What remains is the selection of the learning rate, δ_k . This is where the WoLF variable learning rate is used.

4.2 Win or Learn Fast

WoLF (“Win or Learn Fast”) is a method by Bowling and Veloso [2002a] for changing the learning rate to encourage convergence in a multiagent reinforcement learning scenario. Notice that the policy gradient ascent algorithm above does not account for a non-stationary environment that arises with simultaneous learning in stochastic games. All of the other agents actions are simply assumed to be part of the environment and unchanging. WoLF provides a simple way to account for other agents through adjusting how quickly or slowly the agent changes its policy.

Since only the rate of learning is changed, algorithms that are guaranteed to find (locally) optimal policies in non-stationary environments retain this property even when using WoLF. In stochastic games with simultaneous learning, WoLF has both theoretical evidence (limited to two-player, two-action matrix games), and empirical evidence (experiments in games with enumerated state spaces) that it encourages convergence in algorithms that don’t otherwise converge. The intuition for this technique is that a learner should adapt quickly when it is doing more poorly than expected. When it is doing better than expected, it should be cautious, since the other players are likely to change their policy. This implicitly accounts for other players that are learning, rather than other techniques that try to explicitly reason about the others’ action choices.

The WoLF principle naturally lends itself to policy gradient techniques where there is a well-defined learning rate, α_k . With WoLF we replace the original learning rate with two learning rates $\alpha_k^w < \alpha_k^l$ to be used when winning or losing, respectively. One determination of winning and losing that has been successful is to compare the value of the current policy $V^\pi(s)$ to the value of the average policy over time $V^{\bar{\pi}}(s)$. So, if $V^\pi(s) > V^{\bar{\pi}}(s)$ then the algorithm is considered winning, otherwise it is losing. With the policy gradient technique above, we cannot easily compute the average policy. Instead we examine the approximate value, using Q_w , of the current weight vector θ with the average weight vector over time $\bar{\theta}$. Specifically, we are “winning” if and only if,

$$\sum_a \pi_{\theta_k}(s, a) Q_{w_k}(s, a) > \sum_a \pi_{\bar{\theta}_k}(s, a) Q_{w_k}(s, a). \quad (7)$$

When winning in a particular state, we update the parameters for that state using α_k^w , otherwise α_k^l .

4.3 Our Task

Returning to the robot task shown in Figure 1, in order to apply GraWoLF we need to select a policy parameterization. Specifically we need to find a mapping from the continuous space of states and actions to a useful feature vector, i.e., to define ϕ_{sa} for a given s and a . The filtered positions and velocities of the two robots form the available state information, and the available actions are points on the field for navigation. By observing that the radial angle of the attacker with respect to the circle is not relevant to the task we arrive at a seven

dimensional input space. These seven dimensions are shown by the white overlaid lines in Figure 1.

We chose to use tile coding [Sutton and Barto, 1998], also known as CMACS, to construct our feature vector. Tile coding uses a number of large offset tilings to allow for both a fine discretization and large amount of generalization. We use 16 tiles whose size was 800mm in distance dimensions and 1600mm/s in velocity dimensions. We simplify the action space by requiring the attacker to select its navigation point along a perpendicular line through the circle’s center. This is shown by the dark overlaid line in Figure 1. This line, whose length is three times the distance of the attacker to the circle, is then discretized into seven points evenly distributed. The defender uses the same points for its actions but then navigates to the point that bisects the line from the attacker to the selected action point. The robot’s action is also included in the tiling as an eighth dimension with a tile size for that dimension of the entire line, so actions are distinguishable but there is also generalization. Agents select actions every tenth of a second, or after every three frames, unless the feature vector has not changed over that time. This keeps the robots from oscillating too much during the initial parts of learning.

5 Results

Before presenting results on applying GraWoLF to this problem we first consider some issues related to evaluation.

5.1 Evaluation

Evaluation of multi-robot learning algorithms present a number of challenges. The first is the problem of data gathering on a real robot platform. Learning often requires far more trials than is practical to execute in the physical world. We believe and demonstrate that the GraWoLF technique is practical for the time constraints of real robots. Yet, from a research standpoint, we want thorough and statistically significant evaluation, which requires far more trials than just those used for learning. We solve this problem by using both a simulator of our robot team as well as the robots themselves to show that the approach is both practical for robots while still providing an extensive analysis of the results.

The second challenge is the problem of evaluating the success of simultaneous learning. The traditional single-agent evaluation that shows performance over time converging to some optimal value is not applicable. Multiagent domains have no single optimal value independent of the other agents’ behavior. The optimal value is changing over time as the other agents also learn. This is especially true of self-play experiments where both agents are using an identical learning algorithm, and any performance success by one agent is necessarily a performance failure for the other.

On the other hand we would still want learning robots, even in self-play, to improve their policy over time. In this paper, our main evaluation compares the performance of the learned policy with the the performance of the initial policy before learning. The initial policy is a random selection of the available actions, and by design of the available actions is actually a fairly capable policy for both agents. We also use the evaluation technique of challengers, which was first examined by

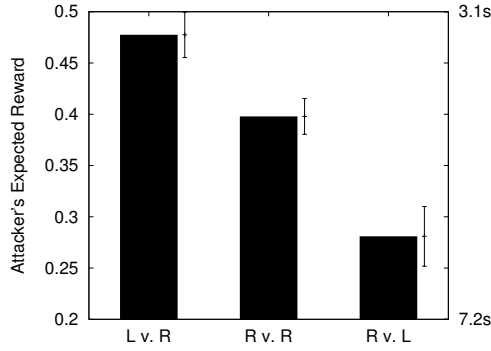


Figure 2: The value of learned policies compared to a random opponent in simulation. Lines to the right of the bars show standard deviations.

Littman [1994]. This technique trains a worst-case opponent, or challenger, for a particular policy to see the generality of the learned policy. In this paper we present challenger results demonstrating that the learned policies are indeed robust, and that the WoLF variable learning rate plays a critical role in keeping the learning away from easily exploited policies.

5.2 Experiments

In all of the performance graphs in this section, the y-axis shows the attacker's expected discounted reward, which roughly corresponds to the expected time it takes for the attacker to reach the circle. On the right of the graph the range is shown in seconds. All measurements of expected discounted reward are gathered empirically over the course of 1000 trials. All training occurred over 4000 trials, and takes approximately 6-7 hours of training time on the robot platform or in simulation. Unless otherwise noted, the training was always done with a simultaneously learning opponent, both using GraWoLF. The experiments in simulation were repeated nine times and the averages are shown in the graphs. We first examine the performance of the policies learned in simulation, and then examine the performance of learning on the robot.

Figure 2 shows the results of various learned policies when playing against an opponent following the random policy, which was the starting point for learning. The middle bar, “R v. R”, corresponds to the expected value of both players following the random policy. “L v. R” corresponds to the value of the attacker following the policy learned in self-play against a random defender. “R v. L” corresponds to the value of a random attacker against the defender policy learned in self-play.

Notice that, as was desired, learning does improve performance over the starting policy. The learned attacker policy against random does far better than the random attacker policy against the learned defender. These experiments demonstrate that GraWoLF improves considerably on its starting policy. The next experiment explores how robust the learned policy is and the effect of the WoLF component.

Figure 3 shows results of challenger experiments. Policies are trained using simultaneous learning. The policy is then fixed and a challenger policy is trained for 4000 trials,

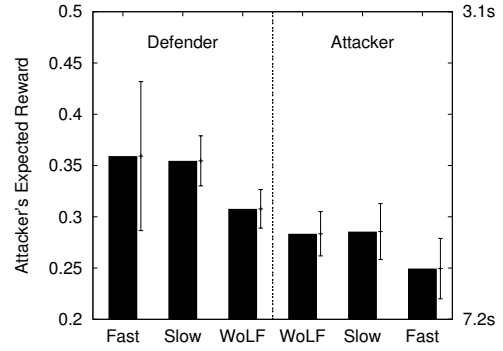


Figure 3: The value of learned policies playing their challengers in simulation. Lines to the right of the bars show standard deviations.

to specifically measure the policy's worst-case performance. The better the worst-case performance, the less exploitable and more robust the policy is to unknown opponents. We trained a challenger against the policies learned after 3000, 3500, and 4000 trials, averaging together the results. We used this experiment to investigate the robustness of the learned policy and the affect of the WoLF variable learning rate on the GraWoLF algorithm. The left side of the graph shows the worst-case performance of learned defender policies, while the right side shows the worst-case performance for attacker policies. “WoLF” corresponds to the described GraWoLF algorithm, “Slow” does not use a variable learning rate but rather always uses the WoLF's winning rate, while “Fast” always uses its losing rate.

First, notice that the distance between the worst-case performance of the defender and the worst-case performance of the attacker (the third and fourth column of Table 3, respectively) is quite small. This demonstrates that the learned policies are quite close to the equilibrium policy, if one exists. This also means that the learned policies are robust and difficult to exploit.

Second, notice that the WoLF learned defender policy performs better against its challenger, i.e., keeps its challenger to a lower reward, than either of the two learning rates it switches between. For the attacker, the learned policy performs better against its challenger than the fast learning rate, but is not significantly different than the slow learning rate. There are a couple of possible reasons for this. One explanation is due to the initialization of values. Since all values were initialized to zero for both sides, this amounts to an optimistic initialization for the defender, and a pessimistic one for the attacker (as all rewards are non-negative for the attacker.) This may mean the attacker considers itself winning far more often than the defender, causing the slower learning rate to be employed most of the time. There is evidence to this effect when examining the percentage of updates that use δ_w versus δ_l . During training, the attacker used the slower, winning rate for 92.3% of its updates, while the defender used the winning rate for only 84.2%. The effect of value initialization on GraWoLF is an interesting topic to be explored further. Overall, although the results are certainly not as dramatic as the unapproximated results [Bowling and Veloso, 2002a], WoLF still

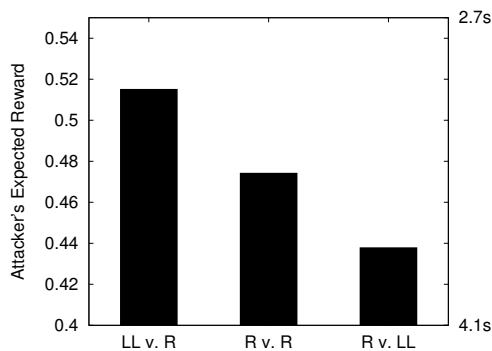


Figure 4: The value of learned policies playing the random policy on the real robots. Training was done with the first half of the trials in simulation and the last half on the robots.

seems to be providing a converging effect.

Finally, we examine results of using GraWoLF on the real robots. We took policies that were trained for 2000 trials in simulation and then did an additional 2000 trials of training on the robots. We evaluated the resulting policies against the random policy as we did with the simulator results in Figure 2. These results are shown in Figure 4

In the real robots the attacker is nearly impossible to keep from reaching the circle. The defender can at best only slow down its progress, and this is even true for a random attacker policy. This can be seen in Figure 4 by the much higher rewards as compared to the simulator results. Despite this, these results are qualitatively identical to the results produced in simulation. Simultaneous learning improves the performance for both the attacker and defender over their initial policies.

Finally, as an adversarial environment we would expect good policies to be stochastic in nature. This is true of all of the learned policies in both simulation and on the robots. For example, the most probable action in any given state has probability on average around 40% in the attacker's learned policy, and 70% in the defender's. In some states, though, the learned policies are nearly deterministic. So, the learned policies are indeed stochastic to avoid exploitation, while still learning optimal actions in states that don't depend on the other agent's behavior.

6 Conclusion

We have introduced, GraWoLF, a general-purpose multiagent learning algorithm capable of learning robot tasks in multi-robot, and even adversarial, environments. We showed results of this algorithm learning in one particular adversarial robot task, both in simulation and from actual robot experience. These results both demonstrated the effectiveness of the policy gradient learner, and the importance of the WoLF variable learning rate. It should be noted that the use of Sutton and colleagues' particular policy gradient formulation is not critical. It would be interesting to combine WoLF with other policy gradient techniques, such as [Williams, 1992; Baxter and Bartlett, 2000].

Acknowledgements

This research was sponsored by Grants F30602-00-2-0549 and DABT63-99-1-0013. The content of this publication does not necessarily reflect the position of the funding agencies and no official endorsement should be inferred. The authors also thank Brett Browning and James Bruce for the development of the CMDragons'02 robots used in this work.

References

- [Baxter and Bartlett, 2000] Johnathan Baxter and Peter L. Bartlett. Reinforcement learning in POMDP's via direct gradient ascent. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 41–48, Stanford University, June 2000. Morgan Kaufman.
- [Bowling and Veloso, 2002a] Michael Bowling and Manuela Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136:215–250, 2002.
- [Bowling and Veloso, 2002b] Michael Bowling and Manuela M. Veloso. Existence of multiagent equilibria with limited agents. Technical report CMU-CS-02-104, Computer Science Department, Carnegie Mellon University, 2002.
- [Claus and Boutilier, 1998] Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, Menlo Park, CA, 1998. AAAI Press.
- [Greenwald and Hall, 2002] Amy Greenwald and Keith Hall. Correlated Q-learning. In *Proceedings of the AAAI Spring Symposium Workshop on Collaborative Learning Agents*, 2002.
- [Littman, 1994] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 157–163. Morgan Kaufman, 1994.
- [Nash, Jr., 1950] John F. Nash, Jr. Equilibrium points in n -person games. *PNAS*, 36:48–49, 1950.
- [Singh et al., 2000] Satinder Singh, Michael Kearns, and Yishay Mansour. Nash convergence of gradient dynamics in general-sum games. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 541–548. Morgan Kaufman, 2000.
- [Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning*. MIT Press, 1998.
- [Sutton et al., 2000] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12*, pages 1057–1063. MIT Press, 2000.
- [Williams, 1992] R. Williams. Simple statistical gradient following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

Taming Decentralized POMDPs: Towards Efficient Policy Computation for Multiagent Settings

R. Nair and M. Tambe

Computer Science Dept.
University of Southern California
Los Angeles CA 90089
{nair,tambe}@usc.edu

M. Yokoo

Coop. Computing Research Grp.
NTT Comm. Sc. Labs
Kyoto, Japan 619-0237
yokoo@cslab.kecl.ntt.co.jp

D. Pynadath, S. Marsella

Information Sciences Institute
University of Southern California
Marina del Rey CA 90292
{pynadath, marsella}@isi.edu

Abstract

The problem of deriving joint policies for a group of agents that maximize some joint reward function can be modeled as a *decentralized partially observable Markov decision process* (POMDP). Yet, despite the growing importance and applications of decentralized POMDP models in the multiagents arena, few algorithms have been developed for efficiently deriving joint policies for these models. This paper presents a new class of locally optimal algorithms called “Joint Equilibrium-based search for policies (JESP)”. We first describe an exhaustive version of JESP and subsequently a novel dynamic programming approach to JESP. Our complexity analysis reveals the potential for exponential speedups due to the dynamic programming approach. These theoretical results are verified via empirical comparisons of the two JESP versions with each other and with a globally optimal brute-force search algorithm. Finally, we prove piece-wise linear and convexity (PWLC) properties, thus taking steps towards developing algorithms for continuous belief states.

1 Introduction

As multiagent systems move out of the research lab into critical applications such as multi-satellite control, researchers need to provide high-performing, robust multiagent designs that are as nearly optimal as feasible. To this end, researchers have increasingly resorted to decision-theoretic models as a framework in which to formulate and evaluate multiagent designs. Given a group of agents, the problem of deriving separate policies for them that maximize some joint reward can be modeled as a decentralized POMDP (**P**artially **O**bservable **M**arkov **D**ecision **P**rocess). In particular, the DEC-POMDP (**D**ecentralized POMDP) [Bernstein *et al.*, 2000] and MTDP (Markov Team Decision Problem [Pynadath and Tambe, 2002]) are generalizations of a POMDP to the case where there are multiple, distributed agents basing their actions on their separate observations. These frameworks allow a variety of multiagent analysis. Of particular interest here, they allow us to formulate what constitutes an optimal policy for a multiagent system and in principle derive that policy.

However, with a few exceptions, effective algorithms for deriving policies for decentralized POMDPs have not been developed. Significant progress has been achieved in efficient single-agent POMDP policy generation algorithms [Monahan, 1982; Cassandra *et al.*, 1997; Kaelbling *et al.*, 1998]. However, it is unlikely such research can be directly carried over to the decentralized case. Finding optimal policies for decentralized POMDPs is NEXP-complete [Bernstein *et al.*, 2000]. In contrast, solving a POMDP is PSPACE-complete [Papadimitriou and Tsitsiklis, 1987]. As Bernstein *et al.* [2000] note, this suggests a fundamental difference in the nature of the problems. The decentralized problem cannot be treated as one of separate POMDPs in which individual policies can be generated for individual agents because of possible cross-agent interactions in the reward, transition or observation functions. (For any one action of one agent, there may be many different rewards possible, based on the actions that other agents may take.) In some domains, one possibility is to simplify the nature of the policies considered for each of the agents. For example, Chadès *et al.* [2002] restrict the agent policies to be memoryless (reactive) policies. Further, as an approximation, they define the reward function and the transition function over observations instead of over states thereby simplifying the problem to solving a multi-agent MDP [Boutilier, 1996]. Xuan *et al.* [2001] describe how to derive decentralized MDP (not POMDP) policies from a centralized MDP policy. Their algorithm, which starts with an assumption of full communication that is gradually relaxed, relies on instantaneous and noise free communication. Such simplifications reduce the applicability of the approach and essentially side-step the question of solving decentralized POMDPs. Peshkin *et al.* [2000] take a different approach by using gradient descent search to find local optimum finite-controllers with bounded memory. Their algorithm finds locally optimal policies from a limited subset of policies, with an infinite planning horizon, while our algorithm finds locally optimal policies from an unrestricted set of possible policies, with a finite planning horizon.

Thus, there remains a critical need for new efficient algorithms for generating optimal policies in distributed POMDPs. In this paper, we present a new class of algorithms for solving decentralized POMDPs, which we refer to as Joint Equilibrium-based Search for Policies (JESP). JESP iterates through the agents, finding an optimal policy for each agent

assuming the policies of the other agents are fixed. The iteration continues until no improvements to the joint reward is achieved. Thus JESP achieves a local optimum similar to a Nash Equilibrium. We discuss Exhaustive-JESP which uses exhaustive search to find the best policy for each agent. Since this exhaustive search for even a single agent's policy can be very expensive, we also present DP-JESP which improves on Exhaustive-JESP by using dynamic programming to incrementally derive the policy. We conclude with several empirical evaluation that contrast JESP against a globally optimal algorithm that derives the globally optimal policy via a full search of the space of policies. Finally, we prove piece-wise linear and convexity (PWLC) properties, thus taking steps towards developing algorithms for continuous initial belief states.

2 Model

We describe the Markov Team Decision Problem (MTDP) [Pynadath and Tambe, 2002] framework in detail here to provide a concrete illustration of a decentralized POMDP model. However, other decentralized POMDP models could potentially also serve as a basis [Bernstein *et al.*, 2000; Xuan *et al.*, 2001].

Given a team of n agents, an MTDP [Pynadath and Tambe, 2002] is defined as a tuple: $\langle S, A, P, \Omega, O, R \rangle$. S is a finite set of world states $\{1, 2, \dots, m\}$. $A = \times_{1 \leq i \leq n} A_i$, where $A_1, \dots, A_n$, are the sets of action for agents 1 to n . A joint action is represented as $\langle a_1, \dots, a_n \rangle$. $P(s_i, \langle a_1, \dots, a_n \rangle, s_f)$, the transition function, represents the probability of the current state is s_f , if the previous state is s_i and the previous joint action is $\langle a_1, \dots, a_n \rangle$. $\Omega_1, \dots, \Omega_n$ are the set of observations for agents 1 to n . $O(s, \langle a_1, \dots, a_n \rangle, \omega)$, the observation function, represents the probability of joint observation ω , if the current state is s and the previous joint action is $\langle a_1, \dots, a_n \rangle$. The agents receive a single, immediate joint reward $R(s, a_1, \dots, a_n)$ which is shared equally.

Practical analysis using models like MTDP often assume that observations of each agent is independent of each other's observations. Thus the observation function can be expressed as $O(s, \langle a_1, \dots, a_n \rangle, \omega) = O_1(s, \langle a_1, \dots, a_n \rangle, \omega_1) \cdot \dots \cdot O_n(s, \langle a_1, \dots, a_n \rangle, \omega_n)$.

Each agent i chooses its actions based on its local *policy*, π_i , which is a mapping of its observation history to actions. Thus, at time t , agent i will perform action $\pi_i(\vec{\omega}_i^t)$ where $\vec{\omega}_i^t = \omega_i^1, \dots, \omega_i^t$. $\pi = \langle \pi_1, \dots, \pi_n \rangle$ refers to the joint policy of the team of agents. The important thing to note is that in this model, execution is distributed but planning is centralized. Thus agents don't know each other's observations and actions at run-time but they know each other's policies.

3 Example Scenario

For illustrative purposes it is useful to consider a familiar and simple example, yet one that is capable of bringing out key difficulties in creating optimal policies for MTDPs. To that end, we consider a multiagent version of the classic

tiger problem used in illustrating single agent POMDPs [Kaelbling *et al.*, 1998] and create an MTDP ($\langle S, A, P, \Omega, O, R \rangle$) for this example. In our modified version, two agents are in a corridor facing two doors: "left" and "right". Behind one door lies a hungry tiger and behind the other lies untold riches but the agents do not know the position of either. Thus, $S = \{SL, SR\}$, indicating behind which door the tiger is present. The agents can jointly or individually open either door. In addition, the agents can independently listen for the presence of the tiger. Thus, $A_1 = A_2 = \{OpenLeft', OpenRight', Listen'\}$. The transition function P , specifies that every time either agent opens one of the doors, the state is reset to SL or SR with equal probability, regardless of the action of the other agent, as shown in Table 1. However, if both agents listen, the state remains unchanged. After every action each agent receives an observation about the new state. The observation function, O_1 or O_2 (shown in Table 2) will return either HL or HR with different probabilities depending on the joint action taken and the resulting world state. For example, if both agents listen and the tiger is behind the left door (state is SL), each agent receives the observation HL with probability 0.85 and HR with probability 0.15.

Action/Transition	SL $\rightarrow$ SL	SL $\rightarrow$ SR	SR $\rightarrow$ SR	SR $\rightarrow$ SL
<OpenRight,*>	0.5	0.5	0.5	0.5
<OpenLeft,*>	0.5	0.5	0.5	0.5
<*,OpenLeft>	0.5	0.5	0.5	0.5
<*,OpenRight>	0.5	0.5	0.5	0.5
<Listen,Listen>	1.0	0.0	1.0	0.0

Table 1: Transition function

Action	State	HL	HR
<Listen,Listen>	SL	0.85	0.15
<Listen,Listen>	SR	0.15	0.85
<OpenRight,*>	*	0.5	0.5
<OpenLeft,*>	*	0.5	0.5
<*,OpenLeft>	*	0.5	0.5
<*,OpenRight>	*	0.5	0.5

Table 2: Observation function for each agent

If either of them opens the door behind which the tiger is present, they are both attacked (equally) by the tiger (see Table 3). However, the injury sustained if they opened the door to the tiger is less severe if they open that door jointly than if they open the door alone. Similarly, they receive wealth which they share equally when they open the door to the riches in proportion to the number of agents that opened that door. The agents incur a small cost for performing the 'Listen' action.

Clearly, acting jointly is beneficial (e.g., $A_1 = A_2 = OpenLeft'$) because the agents receive more riches and sustain less damage by acting together. However, because the agents receive independent observations (they do not share observations), they need to consider the observation histories of the other agent and what action they are likely to perform.

Action/State	SL	SR
<OpenRight,OpenRight>	+20	-50
<OpenLeft,OpenLeft>	-50	+20
<OpenRight,OpenLeft>	-100	-100
<OpenLeft,OpenRight>	-100	-100
<Listen,Listen>	-2	-2
<Listen,OpenRight>	+9	-101
<OpenRight,Listen>	+9	-101
<Listen,OpenLeft>	-101	+9
<OpenLeft,Listen>	-101	+9

Table 3: Reward function A

We also consider another case of the reward function, where we vary the penalty for jointly opening the door to the tiger (See Table 4).

Action/State	SL	SR
<OpenRight,OpenRight>	+20	0
<OpenLeft,OpenLeft>	-50	+20
<OpenRight,OpenLeft>	-100	-100
<OpenLeft,OpenRight>	-100	-100
<Listen,Listen>	-2	-2
<Listen,OpenRight>	+9	-101
<OpenRight,Listen>	+9	-101
<Listen,OpenLeft>	-101	+9
<OpenLeft,Listen>	-101	+9

Table 4: Reward function B

4 Optimal Joint Policy

When agents do not share all of their observations, they must instead coordinate by selecting policies that are sensitive to their teammates’ possible beliefs, of which each agent’s entire history of observations provides some information. The problem facing the team is to find the *optimal* joint policy, i.e. a combination of individual agent policies that produces behavior that maximizes the team’s expected reward.

One sure-fire method for finding the optimal joint policy is to simply search the entire space of possible joint policies, evaluate the expected reward of each, and select the policy with the highest such value. To perform such a search, we must first be able to determine the expected reward of a joint policy. We compute this expectation by projecting the team’s execution over all possible branches on different world states and different observations. We present here the 2-agent version of this computation, but the results easily generalize to arbitrary team sizes. At each time step, we can compute the expected value of a joint policy, $\pi = \langle \pi_1, \pi_2 \rangle$, for a team starting in a given state, S^t , with a given set of past observations, $\vec{\omega}_1^t$ and $\vec{\omega}_2^t$, as follows:

$$\begin{aligned}
V_\pi^t(S^t, \langle \vec{\omega}_1^t, \vec{\omega}_2^t \rangle) &= R(S^t, \langle \pi_1(\vec{\omega}_1^t), \pi_2(\vec{\omega}_2^t) \rangle) + \\
&\sum_{S^{t+1} \in S} P(S^t, \langle \pi_1(\vec{\omega}_1^t), \pi_2(\vec{\omega}_2^t) \rangle, S^{t+1}) \\
&\cdot \sum_{\omega_1^{t+1} \in \Omega_1} \sum_{\omega_2^{t+1} \in \Omega_2} O(S^{t+1}, \langle \pi_1(\vec{\omega}_1^t), \pi_2(\vec{\omega}_2^t) \rangle, \langle \omega_1^{t+1}, \omega_2^{t+1} \rangle) \\
&\cdot V_\pi^{t+1}(S^{t+1}, \langle \vec{\omega}_1^{t+1}, \vec{\omega}_2^{t+1} \rangle)
\end{aligned} \tag{1}$$

At each time step, the computation of V_π^t performs a summation over all possible world states and agent observations, so the time complexity of this algorithm is $O(|S| \cdot |\Omega_1| \cdot |\Omega_2|)^T$. The overall search performs this computation for each and every possible joint policy. Since each policy specifies different actions over possible histories of observations, the number of possible policies for an individual agent i is $O(|A_i|^{\frac{|\Omega_i|T-1}{|\Omega_i|-1}})$. The number of possible *joint* policies for n agents is thus $O\left(\left(|A_*|^{\frac{|\Omega_*|T-1}{|\Omega_*|-1}}\right)^n\right)$, where A_* and Ω_* correspond to the largest individual action and observation spaces, respectively, among the agents. The time complexity for finding the optimal joint policy by searching this space is thus: $O\left(\left(|A_*|^{\frac{|\Omega_*|T-1}{|\Omega_*|-1}}\right)^n \cdot (|S| \cdot |\Omega_*|^n)^T\right)$

5 Joint Equilibrium-based Search for Policies

Given the complexity of exhaustively searching for the optimal joint policy, it is clear that such methods will not be successful when the amount of time to generate the policy is restricted. In this section, we will present algorithms that are guaranteed to find a locally optimal joint policy. We refer to this category of algorithms as “JESP” (Joint Equilibrium-Based Search for Policies). Just like the solution in Section 4, the solution obtained using JESP is a Nash equilibrium. In particular it is a locally optimal solution to a partially observable identical payoff stochastic game (POIPSG) [Peshkin *et al.*, 2000]. The key idea is to find the policy that maximizes the joint expected reward for one agent at a time, keeping the policies of all the other agents fixed. This process is repeated until an equilibrium is reached (local optimum is found). The problem of which optimum the agents should select when there are multiple local optima is not encountered since planning is centralized.

5.1 Exhaustive approach(Exhaustive-JESP)

The algorithm below describes an exhaustive approach for JESP. Here we consider that there are n cooperative agents. We modify the policy of one agent at a time keeping the policies of the other $n - 1$ agents fixed. The function `best - Policy`, returns the joint policy that maximizes the expected joint reward, obtained by keeping $n - 1$ agents’ policies fixed and exhaustively searching in the entire policy space of the agent whose policy is free. Therefore at each iteration, the value of the modified joint policy will always either

increase or remain unchanged. This is repeated until an equilibrium is reached, i.e. the policies of all n agents remains unchanged. This policy is guaranteed to be a local maximum since the value of the new joint policy at each iteration is non-decreasing.

Algorithm 1 EXHAUSTIVE-JESP()

```

1: prev  $\leftarrow$  random joint policy, conv  $\leftarrow$  0
2: while conv  $\neq$   $n - 1$  do
3:   for  $i \leftarrow 1$  to  $n$  do
4:     fix policy of all agents except  $i$ 
5:     policySpace  $\leftarrow$  list of all policies for  $i$ 
6:     new  $\leftarrow$  bestPolicy( $i$ , policySpace, prev)
7:     if new.value = prev.value then
8:       conv  $\leftarrow$  conv + 1
9:     else
10:      prev  $\leftarrow$  new, conv  $\leftarrow$  0
11:   if conv =  $n - 1$  then
12:     break
13: return new

```

The best policy cannot remain unchanged for more than $n - 1$ iterations without convergence being reached and in the worst case, each and every joint policy is the best policy for at least one iteration. Hence, this algorithm has the same worst case complexity as the exhaustive search for a globally optimal policy. However, it could do much better in practice as illustrated in Section 6. Although the solution found by this algorithm is a local optimum, it may be adequate for some applications. Techniques like *random restarts* or *simulated annealing* can be applied to perturb the solution found to see if it settles on a different higher value.

The exhaustive approach to Steps 5 and 6 of the Exhaustive-JESP algorithm enumerates and searches the entire policy space of a single agent, i . There are $O\left(|A_i|^{\frac{|\Omega_i|^T - 1}{|\Omega_i| - 1}}\right)$ such policies, and evaluating each incurs a time complexity of $O(|S| \cdot |\Omega|^T)$. Thus, using the exhaustive approach incurs an overall time complexity in Steps 5 and 6 of: $O\left(|A_i|^{\frac{|\Omega_i|^T - 1}{|\Omega_i| - 1}} |S| \cdot |\Omega|^T\right)$. Since we incur this complexity cost in each and every pass through the JESP algorithm, a faster means of performing the bestPolicy function call of Step 6 would produce a big payoff in overall efficiency. We describe a dynamic programming alternative to this exhaustive approach for doing JESP next.

5.2 Dynamic Programming (DP-JESP)

If we examine the single-agent POMDP literature for inspiration, we find algorithms that exploit dynamic programming to incrementally construct the best policy, rather than simply search the entire policy space [Monahan, 1982; Cassandra *et al.*, 1997; Kaelbling *et al.*, 1998]. These algorithms rely on a *principle of optimality* that states that each sub-policy of an overall optimal policy must also be optimal. In other words, if we have a T -step optimal policy, then, given the history over the first t steps, the portion of that policy that covers the last $T - t$ steps must also be optimal over the remaining $T - t$

steps. In this section, we show how we can exploit an analogous optimality property in the multiagent case to perform more efficient construction of the optimal policy within our JESP algorithm.

To support such a dynamic-programming algorithm, we must define *belief states* that summarize an agent's history of past observations, so that they allow the agents to ignore the actual history of past observations, while still supporting construction of the optimal policy over the possible future. In the single-agent case, a belief state that stores the distribution, $B_{single}^t = \Pr(S^t | \vec{\omega}^t)$, is a *sufficient statistic*, because the agent can compute an optimal policy based on B_{single}^t without having to consider the actual observation sequence, $\vec{\omega}^t$ [Sondik, 1971].

In the multiagent case, an agent faces a complex but normal single-agent POMDP if the policies of all other agents are fixed. However, B_{single}^t is not sufficient, because the agent must also reason about the action selection of the other agents and hence on the observation histories of the other agents. Thus, at each time t , the agent i reasons about the tuple $e_i^t = \langle S^t, \vec{\omega}_{\neq i}^t \rangle$, where $\vec{\omega}_{\neq i}^t = \langle \vec{\omega}_1^t, \dots, \vec{\omega}_{i-1}^t, \vec{\omega}_{i+1}^t, \dots, \vec{\omega}_n^t \rangle$ is the joint observation histories of all the agents except i . By treating e_i^t to be the state of the agent i at time t , we can define the transition function and observation function for the single agent POMDP for agent i as follows:

$$\begin{aligned}
P'(e_i^t, a_i^t, e_i^{t+1}) &= \Pr(e_i^{t+1} | e_i^t, a_i^t) \\
&= P(S^t, (\pi_{\neq i}(\vec{\omega}_{\neq i}^t), a_i^t), S^{t+1}) \\
&\quad \cdot \prod_{j \neq i} O_j(S^{t+1}, (\pi_{\neq i}(\vec{\omega}_{\neq i}^t), a_i^t), \omega_j^{t+1}) \quad (2)
\end{aligned}$$

$$\begin{aligned}
O'(e_i^{t+1}, a_i^t, \omega_i^{t+1}) &= \Pr(\omega_i^{t+1} | e_i^{t+1}, a_i^t) \\
&= O_i(S^{t+1}, (\pi_{\neq i}(\vec{\omega}_{\neq i}^t), a_i^t), \omega_i^{t+1}) \quad (3)
\end{aligned}$$

where $\pi_{\neq i} = \langle \pi_1^t, \dots, \pi_{i-1}^t, \pi_{i+1}^t, \dots, \pi_n^t \rangle$ is the joint policy for all agents except i .

We now define the novel *multiagent* belief state for an agent i given the distribution over the initial state, $b(s) = \Pr(S^1 = s)$:

$$B_i^t = \Pr(e_i^t | \vec{\omega}_i^t, \vec{a}_i^{t-1}, b) \quad (4)$$

In other words, when reasoning about an agent's policy in the context of other agents, we maintain a distribution over e_i^t , rather than simply the current state. Figure 1 shows different belief states B^1 , B^2 and B'^2 for agent 1 in the tiger domain. For instance, B^2 , shows probability distributions over e_1^2 . In $e_1^2 = (SL, (HR))$, (HR) is the history of agent 2's observations while SL is the current state. Section 5.3 demonstrates how we can use this multiagent belief state to construct a dynamic program that incrementally constructs the optimal policy for agent i .

5.3 The Dynamic Programming Algorithm

Following the model of the single-agent value-iteration algorithm, our dynamic program centers around a value function over a T -step finite horizon. For readability, this section presents the derivation for the dynamic program in the

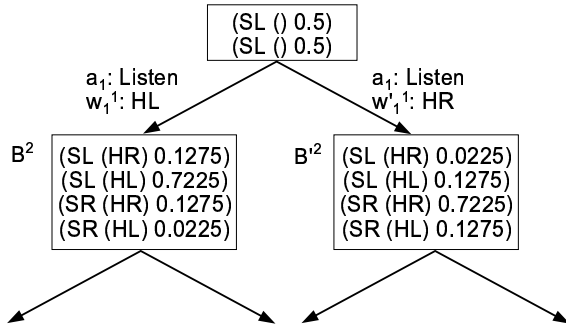


Figure 1: Trace of Tiger Scenario

two-agent case; the results easily generalize to the n -agent case. Having fixed the policy of agent 2, our value function, $V_t(B^t)$, represents the expected reward that the team will receive when agent 1 follows its optimal policy from the t -th step onwards when starting with a current belief state, B_1^t . We start at the end of the time horizon (i.e., $t = T$), and then work our way back to the beginning. Along the way, we construct the optimal policy by maximizing our value function over possible action choices:

$$V_t(B_1^t) = \max_{a_1 \in A_1} V_t^{a_1}(B_1^t) \quad (5)$$

We can define the action value function, $V_t^{a_1}$, recursively:

$$V_t^{a_1}(B_1^t) = ER(a_1, B_1^t) + \sum_{\omega_1^{t+1} \in \Omega_1} \Pr(\omega_1^{t+1} | B_1^t, a_1) \cdot V_{t+1}(B_1^{t+1}) \quad (6)$$

The first term in equation 6 refers to the expected immediate reward, while the second term refers to the expected future reward. B_1^{t+1} is the belief state updated after performing action a_1 and observing ω_1^{t+1} . In the base case, $t = T$, the future reward is 0, leaving us with:

$$V_T^{a_1}(B_1^T) = ER(a_1, B_1^T) \quad (7)$$

The calculation of expected immediate reward breaks down as follows:

$$ER(a_1, B_1^t) = \sum_{e_1^t = \langle S^t, \vec{\omega}_2^t \rangle} B_1^t(e_1^t) \cdot R(S^t, (a_1, \pi_2(\vec{\omega}_2^t))) \quad (8)$$

Thus, we can compute the immediate reward using only the agent's current belief state and the primitive elements of our given MTDP model (See Section 2).

Computation of the expected future reward (the second term in Equation 6) depends on our ability to update agent 1's belief state from B_1^t to B_1^{t+1} in light of the new observation, ω_1^{t+1} . For example, in Figure 1, the belief state B^1 is updated to B'^2 on performing action a_1 and receiving observation ω_1^{11} . We now derive an algorithm for performing such an update, as well as computing the remaining $\Pr(\omega | B, a)$ term from Equation 6. The initial belief state based on the distribution over initial state, b , is:

$$B^1(e_1^1) = b(S^1) \quad (9)$$

For $e_1^{t+1} = \langle S^{t+1}, \langle \vec{\omega}_2^t, \omega_2^{t+1} \rangle \rangle$, the updated $B_1^{t+1}(e_1^{t+1})$ is obtained using Equations 2 and 3 and Bayes Rule and is given as follows:

$$B_1^{t+1}(e_1^{t+1}) = \sum_{S^t} B_1^t(e_1^t) \cdot P(S^t, (a_1, \pi_2(\vec{\omega}_2^t)), S^{t+1}) \cdot O_1(S^{t+1}, (a_1, \pi_2(\vec{\omega}_2^t)), \omega_1^{t+1}) \cdot O_2(S^{t+1}, (a_1, \pi_2(\vec{\omega}_2^t)), \omega_2^{t+1}) / \Pr(\omega_1^{t+1} | B_1^t, a_1) \quad (10)$$

We treat the denominator of Equation 10 (i.e., $\Pr(\omega_1^{t+1} | B_1^t, a_1)$) as a normalizing constant to bring the sum of the numerator over all e_1^t to be 1. This result also enters into our computation of future expected reward in the second term of Equation 6. Thus, we can compute the agent's new belief state (and the future expected reward and the overall value function, in turn) using only the agent's current belief state and the primitive elements of our given MTDP model. Having computed the overall value function, V_t , we can also extract a form of the optimal policy, π_1 , that maps observation histories into actions, as required by Equations 8 and 10.

Algorithm 2 presents the pseudo-code for our overall dynamic programming algorithm. Lines 1–6 generate all of the belief states reachable from a given initial belief state, $b(s) = \Pr(S^1 = s)$. Since there is a possibly unique belief state for every sequence of actions and observations by agent 1, there are $O(|A_1| \cdot |\Omega_1|)^T$ reachable belief states. This reachability analysis uses our belief update procedure (Algorithm 3), which itself has time complexity $O(|S|^2 |\Omega_2|^{t+1})$ when invoked on a belief state at time t . Thus, the overall reachability analysis phase has a time complexity of $O(|S|^2 (|A_1| \cdot |\Omega_1| \cdot |\Omega_2|)^T)$. Lines 7–22 perform the heart of our dynamic programming algorithm, which also has a time complexity of $O(|S|^2 (|A_1| \cdot |\Omega_1| \cdot |\Omega_2|)^T)$. Lines 23–27 translate the resulting value function into an agent policy defined over observation sequences, as required by our algorithm (i.e., the π_2 argument). This last phase has a lower time and space complexity, $O(|S|^2 |\Omega_1|^T \cdot |\Omega_2|^T)$, than our other two phases, since it considers only optimal actions for agent 1. Thus, the overall time complexity of our algorithm is $O(|S|^2 (|A_1| \cdot |\Omega_1| \cdot |\Omega_2|)^T)$. The space complexity of the resulting value function and policy is essentially the product of the number of reachable belief states and the size of our belief state representation: $O(|S| (|A_1| \cdot |\Omega_1| \cdot |\Omega_2|)^T)$.

5.4 Piecewise Linearity and Convexity of Value Function

Algorithm 2 computes a value function over only those belief states that are reachable from a given initial belief state, which is a subset of all possible probability distributions over S^t and $\vec{\omega}_2^t$. To use dynamic programming over the entire set, we must show that our chosen value function is piecewise linear and convex (PWLC). Each agent is faced with solving a single agent POMDP if the policies of all other agents is fixed as shown in Section 5.2. Sondik [1971] showed that the value function for a single agent POMDP is PWLC. Hence the value function in Equation 5 is PWLC. Thus, in addition

Algorithm 2 OPTIMALPOLICYDP(b, π_2, T)

```
1:  $reachable(0) \leftarrow \{b\}$ 
2: for  $t \leftarrow 1$  to  $T$  do
3:   for all  $B^{t-1} \in reachable(t-1)$  do
4:      $reachable(t) \leftarrow \emptyset$ 
5:     for all  $a_1 \in A_1, \omega_1 \in \Omega_1$  do
6:        $reachable(t) \leftarrow \bigcup UPDATE(B^{t-1}, a_1, \omega_1)$ 
7:   for  $t \leftarrow T$  downto 1 do
8:     for all  $B^t \in reachable(t)$  do
9:        $V_t(B^t) \leftarrow -\infty$ 
10:      for all  $a_1 \in A_1$  do
11:         $V_t^{a_1}(B^t) \leftarrow 0$ 
12:        for all  $s \in S, \vec{\omega}_2 \in \Omega_2^t$  do {Equation 8}
13:           $V_t^{a_1}(B^t) \leftarrow B^t(s, \vec{\omega}_2) \cdot R(s, \langle a_1, \pi_2(\vec{\omega}_2) \rangle)$ 
14:          if  $t < T$  then {Compute future reward}
15:            for all  $\omega_1 \in \Omega_1$  do
16:               $prob \leftarrow 0$ 
17:              for all  $s^t, s^{t+1} \in S, \vec{\omega}_2 \in \Omega_2^t$  do
18:                 $act \leftarrow \langle a_1, \pi_2(\vec{\omega}_2) \rangle$ 
19:                 $prob \leftarrow B^t(s^t, \vec{\omega}_2) \cdot P(s^t, act, s^{t+1}) \cdot$ 
20:                   $O_1(s^{t+1}, act, \omega_1)$ 
21:                 $V_t^{a_1}(B^t) \leftarrow \max(V_t^{a_1}(B^t), prob)$ 
22:                 $V_{t+1}(UPDATE(B^t, a_1, \omega_1)) \leftarrow V_t^{a_1}(B^t)$  {Equation 6}
23:            if  $V_t^{a_1}(B^t) > V_t(B^t)$  then
24:               $V_t(B^t) \leftarrow V_t^{a_1}(B^t)$ 
25:          for all  $\vec{\omega}_1 \in \Omega_1^T$  do
26:             $B^0 \leftarrow b$ 
27:            for  $t \leftarrow 1$  to  $T$  do
28:               $B^t \leftarrow UPDATE(B^{t-1}, \pi_1(\vec{\omega}_1[1 \dots (t-1)]), \vec{\omega}_1[t])$ 
29:               $\pi_1(\vec{\omega}_1[1 \dots t]) \leftarrow \arg \max_{a_1} V_t^{a_1}(B^t)$ 
30: return  $\pi_1$ 
```

Algorithm 3 UPDATE(B^t, a_1, ω_1)

```
for all  $s^{t+1} \in S, \vec{\omega}_2 \in \Omega_2^{t+1}$  do
   $B^{t+1}(s^{t+1}, \vec{\omega}_2) \leftarrow 0$ 
   $act \leftarrow \langle a_1, \pi_2(\vec{\omega}_2[1 \dots t]) \rangle$ 
  for all  $s^t \in S$  do {Equation 10}
     $B^{t+1}(s^{t+1}, \vec{\omega}_2) \leftarrow \max(B^{t+1}(s^{t+1}, \vec{\omega}_2),$ 
     $P(s^t, act, s^{t+1}) \cdot B^t(s^t, \vec{\omega}_2[1 \dots t]) \cdot$ 
     $O_1(s^{t+1}, act, \omega_1))$ 
     $O_2(s^{t+1}, act, \vec{\omega}_2[t+1])$ 
  normalize  $B^{t+1}(s^{t+1}, \vec{\omega}_1)$ 
return  $B^{t+1}$ 
```

to supporting the more efficient dynamic programming of Algorithm 2, our novel choice of belief state space and value function can potentially support a dynamic programming algorithm over the entire continuous space of possible belief states.

6 Experimental Results

In this section, we perform an empirical comparison of the algorithms described in Sections 4 and 5 using the Tiger Scenario (See Section 3) in terms of time and performance. Figure 2, shows the results of running the globally optimal algorithm and the Exhaustive JESP algorithm for two different reward functions (Tables 3 and 4). Finding the globally optimal policy is extremely slow and is doubly exponential in the finite horizon, T and so we evaluate the algorithms only for finite horizons of 2 and 3. We ran the JESP algorithm for 3 different randomly selected initial policy settings and compared the performance of the algorithms in terms of the number of policy evaluations (on Y-axis using log scale) that were necessary. As can be seen from this figure, for $T = 2$ the JESP algorithm requires much fewer evaluations to arrive at an equilibrium. The difference in the run times of the globally optimal algorithm and the JESP algorithm is even more apparent when $T = 3$. Here the globally optimal algorithm performed > 4 million policy evaluations while the JESP algorithm did < 7000 evaluations. For the reward function A, JESP succeeded in finding the globally optimal policies for both $T = 2$ (expected reward = -4) and 3 (expected reward = -6). However, this is not always the case. Using reward function B for $T = 2$, the JESP algorithm sometimes settles on a locally optimal policy (expected reward = -4) that is different from the globally optimal policy (expected reward = 20). However, when random restarts are used, the globally optimal reward can be obtained.

Based on Figure 2, we can conclude that the exhaustive JESP algorithm performs better than an exhaustive search for the globally optimal policy but can sometimes settle on a policy that is only locally optimal. This could be sufficient for problems where the difference between the locally optimal policy's value and the globally optimal policy's value is small and it is imperative that a policy be found quickly. Alternatively, the JESP algorithm could be altered so that it doesn't get stuck in a local optimum via random restarts.

Table 5 compares presents experimental results from comparison of exhaustive JESP with our dynamic programming approach (DP-JESP). These results, also from the tiger domain, show run-time in milliseconds (ms) for the two algorithms with increasing horizon. DP-JESP is seen to obtain significant speedups over exhaustive-JESP. For time horizon of 2 and 3 DP-JESP run time is essentially 0 ms, compared to the significant run times of Exhaustive-JESP. As we increased the horizon to > 3 , we could not run exhaustive-JESP at all; while DP-JESP could be easily run up to horizon of 7.

7 Summary and Conclusion

With the growing importance of decentralized POMDPs in the multiagents arena, for both design and analysis, it is critical to develop efficient algorithms for generating joint poli-

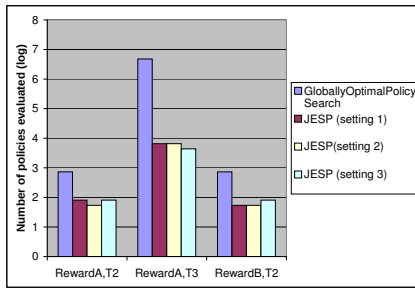


Figure 2: Evaluation Results

Method	2	3	4	5	6	7
Exhaustive-JESP	10	317,800				
DP-JESP	0	0	20	110	1,360	30,030

Table 5: Run time(ms) for various T with Pentium 4, 2.0GHz, 1GB memory, Linux Redhat 7.1, Allegro Common Lisp 6.0

cies. Yet, there is a significant lack of such efficient algorithms. There are three novel contributions in this paper to address this shortcoming. First, given the complexity of the exhaustive policy search algorithm — doubly exponential in the number of agents and time — we describe a class of algorithms called “Joint Equilibrium-based Search for Policies” (JESP) that search for a local optimum rather than a global optimum. In particular, we provide detailed algorithms for Exhaustive JESP and dynamic programming JESP (DP-JESP). Second, we provide complexity analysis for DP-JESP, which illustrates a potential for exponential speedups over exhaustive JESP. We have implemented all of our algorithms, and empirically verified the significant speedups they provide. Third, we provide a proof that the value function for individual agents is piece-wise linear and convex (PWLC) in their belief states. This key result could pave the way to a new family of algorithms that operate over continuous belief states, increasing the range of applications that can be attacked via decentralized POMDPs, and is now a major issue for our future work.

Acknowledgments

We thank Piotr Gmytrasiewicz for discussions related to the paper. This research was supported by NSF grant #0208580 and DARPA award no. F30602-98-2-0108.

References

- [Bernstein *et al.*, 2000] D. Bernstein, S. Zilberstein, and N. Immerman. The complexity of decentralized control of MDPs. In *UAI*, 2000.
- [Boutilier, 1996] C. Boutilier. Planning, learning & coordination in multiagent decision processes. In *TARK-96*, 1996.
- [Cassandra *et al.*, 1997] A. Cassandra, M. Littman, and N. Zhang. Incremental pruning: A simple, fast, exact method for partially observable markov decision processes. In *UAI*, 1997.

- [Chadès *et al.*, 2002] I. Chadès, B. Scherrer, and F. Charpillet. A heuristic approach for solving decentralized-pomdp: Assessment on the pursuit problem. In *SAC*, 2002.
- [Kaelbling *et al.*, 1998] L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101, 1998.
- [Monahan, 1982] G. Monahan. A survey of partially observable markov decision processes: Theory, models and algorithms. *Management Science*, 101(1):1–16, January 1982.
- [Papadimitriou and Tsitsiklis, 1987] C. Papadimitriou and J. Tsitsiklis. Complexity of markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.
- [Peshkin *et al.*, 2000] L. Peshkin, N. Meuleau, K. E. Kim, and L. Kaelbling. Learning to cooperate via policy search. In *UAI*, 2000.
- [Pynadath and Tambe, 2002] D. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *JAIR*, 2002.
- [Sondik, 1971] Edward J. Sondik. The optimal control of partially observable markov processes. *Ph.D. Thesis, Stanford*, 1971.
- [Xuan *et al.*, 2001] P. Xuan, V. Lesser, and S. Zilberstein. Communication decisions in multiagent cooperation. In *Agents-01*, 2001.

A Bayesian Approach to Imitation in Reinforcement Learning

Bob Price

University of British Columbia
Vancouver, B.C., Canada V6T 1Z4
price@cs.ubc.ca

Craig Boutilier

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

Abstract

In multiagent environments, forms of social learning such as teaching and imitation have been shown to aid the transfer of knowledge from experts to learners in reinforcement learning (RL). We recast the problem of imitation in a Bayesian framework. Our *Bayesian imitation model* allows a learner to smoothly pool prior knowledge, data obtained through interaction with the environment, and information inferred from observations of expert agent behaviors. Our model integrates well with recent Bayesian exploration techniques, and can be readily generalized to new settings.

1 Introduction

Reinforcement learning is a flexible, yet computationally challenging paradigm. Recent results demonstrating that under certain assumptions the sample complexity of reinforcement learning is polynomial in the number of problem states [Kearns and Singh, 1998] are tempered by the sober fact that the number of states is generally exponential in the number of the attributes defining a learning problem. With recent interest in building interacting autonomous agents, reinforcement learning is increasingly applied to multiagent tasks, a development which only adds to the complexity of learning [Littman, 1994; Hu and Wellman, 1998]. In this paper, we examine multi-agent reinforcement learning under the assumption that other agents in the environment are not merely arbitrary actors, but actors “like me”. That is, the other agents may have similar action capabilities and similar objectives. This assumption radically changes the optimal learning strategy. Information about other agents “like me” can give the learning agent additional information about its own capabilities and how these capabilities relate to its own objectives. A number of techniques have been developed to exploit this, including *imitation* [Demiris and Hayes, 1999; Matarić, 2002], *learning by watching* [Kuniyoshi et al., 1994], *teaching or programming by demonstration* [Atkeson and Schaal, 1997] *behavioral cloning* [Sammut et al., 1992], and *inverse reinforcement learning* [Ng and Russell, 2000].

Learning by observation of other agents has intuitive appeal; however, explicit communication about action capabilities between agents requires considerable infrastructure: a

communication channel, a sufficiently expressive representation language, a transformation between possibly different agent bodies, and an incentive to communicate. In dynamic, competitive domains, such as web-based trading, it is unrealistic to expect all agents to be designed with compatible representations and altruistic intentions. Observation-based techniques, in which the learning agent observes only the *outward* behaviors of another agent, can reduce the need for explicit communication. Implicit communication through passive observations has been implemented as *implicit imitation* [Price and Boutilier, 1999; 2001]. In this model, the effects of other agents’ action choices on the state of the environment can be observed, but the internal state of other agents and their action control signals are not observable. Independent exploration on the part of the observer is used to adapt knowledge implicit in observations of other agents to the learning agent’s own needs. Unlike classic imitation models, the learner is not required to explicitly duplicate the behavior of other agents.

In this paper, we recast implicit imitation in a Bayesian framework. This new formulation offers several advantages over existing models. First it provides a more principled, and more elegant approach to the smooth pooling of information from the agent’s prior beliefs, its own experience and the observations of other agents (e.g., it eliminates the need for certain ad hoc tuning parameters in current imitation models). Second, it integrates well with state-of-the-art exploration techniques, such as Bayesian exploration. Finally, the Bayesian imitation model can be extended readily to partially-observable domains, though the derivation and implementation are considerably more complex and are not reported here.

2 Background

We assume a reinforcement learning (RL) agent is learning to control a Markov decision processes (MDP) $\langle \mathcal{S}, \mathcal{A}_o, R_o, D \rangle$, with finite state and action sets $\mathcal{S}, \mathcal{A}_o$, reward function $R_o : \mathcal{S} \mapsto \mathbf{R}$, and dynamics D . The dynamics D refers to a set of transition distributions $\Pr(s, a, \cdot)$. The actions $\mathcal{A}_o$ and rewards R_o are subscripted to distinguish them from those of other agents (see below). We assume throughout that the agent knows R_o but not the dynamics D of the MDP (thus we adopt the “automatic programming” perspective), and has the objective of maximizing discounted reward over an infinite horizon. Any of a number of RL techniques can be used to learn an optimal policy $\pi : \mathcal{S} \mapsto \mathcal{A}_o$. We focus here on *model-*

based RL methods, in which the observer maintains an estimated MDP $\langle \mathcal{S}, \mathcal{A}_o, \widehat{R}_o, \widehat{D} \rangle$, based on the set of experiences $\langle s, a, r, t \rangle$ obtained so far. At each stage (or at suitable intervals) this MDP can be solved exactly, or approximately using techniques such as prioritized sweeping [Moore and Atkeson, 1993]. Since R_o is known, we focus on learning dynamics.

Bayesian methods in model-based RL allow agents to incorporate priors and explore optimally. In general, we employ a prior density P over possible dynamics D , and update it with each data point $\langle s, a, t \rangle$. Letting $H_o = \langle s_0, s_1, \dots, s_T \rangle$ denote the (current) *state history* of the observer, and $A_o = \langle a_0, a_1, \dots, a_{T-1} \rangle$ be the action history, we use the posterior $P(D|H_o, A_o)$ to update the action Q-values, which are used in turn to select actions. The formulation of Dearden *et al.* 1999 renders this update tractable by assuming a convenient prior: P is the product of local independent densities for each transition distribution $\Pr(s, a, \cdot)$; and each density $P(D^{s,a})$ is a Dirichlet with parameters $\mathbf{n}^{s,a}$. To model $P(D^{s,a})$ we require one parameter $n^{s,a,s'}$ for each possible successor state s' . Update of a Dirichlet is straightforward: given prior $P(D^{s,a}; \mathbf{n}^{s,a})$ and data vector $\mathbf{c}^{s,a}$ (where $c_t^{s,a}$ is the number of observed transitions from s to t under a), the posterior is given by parameters $\mathbf{n}^{s,a} + \mathbf{c}^{s,a}$. Thus the posterior in Eq. 1 can be factored into posteriors over local families:

$$P(D^{s,a}|H_o^{s,a}) = \alpha \Pr(H_o^{s,a}|D^{s,a})P(D^{s,a}) \quad (1)$$

where $H_o^{s,a}$ is the subset of history composed of transitions from state s due to action a , and the updates themselves are simple Dirichlet parameter updates.

The Bayesian approach has several advantages over other approaches to model-based RL. First, it allows the natural incorporation of priors over transition and reward parameters. Second, approximations to optimal Bayesian exploration can take advantage of this approach, and the specific structural assumptions on the prior discussed above [Dearden *et al.*, 1999].

3 Bayesian Imitation

In multiagent settings, observations of other agents can be used in addition to prior beliefs and personal experience to improve an agent’s model of its environment. These observations can have enormous impact when they provide information to an agent about parts of the state space it has not yet visited. The information can be used to bias exploration towards the most promising regions of state space and thereby reduce exploration costs and speed convergence dramatically.

The flexibility of the Bayesian formulation leads to an elegant and principled mechanism for incorporating these observations into the agent’s model updates. Following Price and Boutilier 1999, we assume two agents, a knowledgeable *mentor* m and a naïve *observer* o , acting simultaneously, but independently, in a fixed environment.¹ Like the observer, the mentor too is controlling an MDP $\langle \mathcal{S}, \mathcal{A}_m, R_m, D \rangle$ with the same underlying state space and dynamics (that is, for any action $a \in \mathcal{A}_o \cap \mathcal{A}_m$, the dynamics are identical). The assumption that the two agents have the same state space is not critical: more important is that there is some analogical mapping

¹We assume that the agents are performing non-interacting tasks.

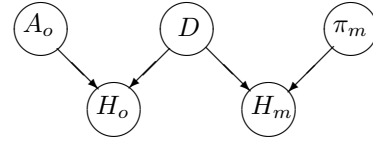


Figure 1: Dependencies among model and evidence sources

between the two [Nehaniv and Dautenhahn, 1998]. We assume full observability of the mentor’s state space; but we do not assume the observer can identify the actions taken by the mentor—it simply observes state transitions.

We make two additional assumptions regarding the mentor’s dynamics: the mentor implements a stationary policy π_m , which induces a Markov chain $\Pr_m(s, s') = \Pr(s, \pi_m^s, s')$; and for each action π_m^s taken by the mentor, there exists an action $a \in A_o$ such that the distributions $\Pr(\cdot|s, a)$ and $\Pr(\cdot|s, \pi_m^s)$ are the same. This latter assumption is the *homogeneous action assumption* and implies that the observer can duplicate the mentor’s policy.² As a consequence we can treat the dynamics D as the same for both agents. Note that we do not assume that the learner knows *a priori* which of its actions duplicates the mentor’s (for any given state s), nor that the observer *wants* to duplicate this policy (as the agents may have different objectives).

Since the learner can observe the mentor’s transitions (though not its actions directly), it can form estimates of the mentor’s Markov chain, along with estimates of its own MDP (transition probabilities and reward function). In [Price and Boutilier, 1999], this estimate is used to augment the normal Bellman backup, treating the observed distribution $\Pr(s, \cdot)$ as a model of an action available to the observer. Imitators using augmented backups based on their observations of a mentor often learn much more quickly, especially if the mentor’s reward function or parts of its policy overlap with that of the observer. Techniques like interval estimation [Kaelbling, 1993] can be used to suppress augmented backups where their value has low “confidence.”

In the Bayesian approach, the observer incorporates observations of the mentor directly into an *augmented model* of its environment. Let H_m denote the history of mentor state transitions observed by the learner. As above, H_o and A_o represents the observer’s state and action history respectively. Figure 1 illustrates the sources of information available to the imitator with which to constrain its beliefs about D , and their probabilistic dependence. While the observer knows its own action history, A_o , it has no direct knowledge of the actions taken by the mentor: at best it may have (often weak) prior knowledge about the mentor’s policy π_m . The learner’s beliefs over D can then be updated w.r.t. the joint observations:

$$\begin{aligned} P(D|H_o, A_o, H_m) \\ &= \alpha \Pr(H_o, H_m|D, A_o)P(D) \\ &= \alpha \Pr(H_o|D, A_o) \Pr(H_m|D)P(D). \end{aligned} \quad (2)$$

²The homogeneous action assumption can be relaxed [Price and Boutilier, 2001]. Essentially, the observer hypothesizes that violations can be “repaired” using a local search for a short sequence of actions that roughly duplicates a short subsequence of the mentor’s actions. If a repair cannot be found, the observer discards the mentor influence (at this point in state space).

We assume that the prior $P(D)$ has the factored Dirichlet form described above. Without mentor influence, a learner can maintain its posterior in the same factored form, updating each component of the model $P(D^{s,a})$ independently. Unfortunately, complications arise due to the unobservability of the mentor's actions. We show, however, that the model update in Eq. 2 can still be factored into convenient terms.

We derive a factored update model for $P(D^{s,a})$ describing the dynamics at state s under action a by considering two cases. In case one, the mentor's unknown action π_m^s could be different than the action a . In this case, the model factor $D^{s,a}$ would be independent of the mentor's history, and we can employ the standard Bayesian update Eq. 1 without regard for the mentor. In case two, the mentor action π_m^s is in fact the same as the observer's action a . Then the mentor observations are relevant to the update of $P(D^{s,a})$:

$$\begin{aligned} P(D^{s,a} | H_o^{s,a}, H_m^s, \pi_m^s = a) \\ &= \alpha \Pr(H_o^{s,a}, H_m^s | D^{s,a}, \pi_m^s = a) P(D^{s,a} | \pi_m^s = a) \\ &= \alpha \Pr(H_o^{s,a} | D^{s,a}) \Pr(H_m^s | D^{s,a}, \pi_m^s = a) P(D^{s,a}). \end{aligned}$$

Let $\mathbf{n}^{s,a}$ be the prior parameter vector for $P(D^{s,a})$, and $\mathbf{c}_o^{s,a}$ denote the counts of observer transitions from state s via action a , and $\mathbf{c}_m^s$ the counts of the mentor transitions from state s . The posterior *augmented model* factor density $P(D^{s,a} | H_o^{s,a}, H_m^s, \pi_m^s = a)$ is then a Dirichlet with parameters $\mathbf{n}^{s,a} + \mathbf{c}_o^{s,a} + \mathbf{c}_m^s$; that is, we simply update with the sum of the observer and mentor counts:

$$P(D^{s,a} | H_o^{s,a}, H_m^s, \pi_m(s) = a) = P(D^{s,a}; \mathbf{n}^{s,a} + \mathbf{c}_o^{s,a} + \mathbf{c}_m^s).$$

Since the observer does not know the mentor's action we compute the expectation w.r.t. these two cases:

$$\begin{aligned} P(D^{s,a} | H_o^{s,a}, H_m^s) \\ &= \Pr(\pi_m^s = a | H_o^{s,a}, H_m^s) P(D^{s,a}; \mathbf{n}^{s,a} + \mathbf{c}_o^{s,a} + \mathbf{c}_m^s) \\ &\quad + \Pr(\pi_m^s \neq a | H_o^{s,a}, H_m^s) P(D^{s,a}; \mathbf{n}^{s,a} + \mathbf{c}_o^{s,a}). \end{aligned} \quad (3)$$

This allows a factored update of the usual conjugate form, but where the mentor counts $\mathbf{c}_m^s$ are distributed across all actions, weighted by the posterior probability that the mentor's policy chooses that action at state s .³

With a mechanism to calculate the posterior over the mentor's policy, Eq. 3 provides a complete *factored* update rule for incorporating evidence from observed mentors by a Bayesian model-based RL agent. To tackle this last problem—that of updating our beliefs about the mentor's policy—we have:

$$\begin{aligned} \Pr(\pi_m | H_m, H_o) \\ &= \alpha \Pr(H_m | \pi_m, H_o) \Pr(\pi_m | H_o) \\ &= \alpha \Pr(\pi_m) \int_{D \in \mathcal{D}} \Pr(H_m | \pi_m, D) P(D | H_o). \end{aligned} \quad (4)$$

If we assume that the prior over the mentor's policy is factored in the same way as the prior over models—that is, we

³This assumes that *at least* one of the observer's actions is equivalent to the mentor's, but our model can be generalized to the heterogeneous case. An additional term is required to represent "none of the above".

have independent distributions $\Pr(\pi_m^s)$ over $\mathcal{A}_m$ for each s —this update can be factored as well, with history elements at state s being the only ones relevant to computing the posterior over $\pi_m(s)$. We still have the difficulty of evaluating the integral over models. Following Dearden *et al.* 1999, we tackle this by sampling models to estimate this quantity. Specifically, we sample models $\dot{D}^{s,a}$ from the factored Dirichlet $P(D^{s,a} | H_o^{s,a})$ over $\mathcal{D}$.⁴ Given a specific sample $\dot{D}^{s,a}$, with parameter vector $\mathbf{n}^{s,a}$, and observed counts $\mathbf{c}_m^s$, the likelihood of $\dot{D}^{s,a}$ is:

$$\Pr(H_m^s | \pi_m, \dot{D}^{s,a}) = \prod_{t \in \mathcal{S}} (n^{s,a,t})^{(c_m^{s,t})}. \quad (5)$$

We can combine the expression for expected model factor probability in Eq. 3 with our expression for mentor policy likelihood in Eq. 5 to obtain a tractable algorithm for updating the observer's beliefs about the dynamics model D based on its own experience, and observations of the mentor.⁵

A Bayesian imitator thus proceeds as follows. At each stage, it observes its own state transition and that of the mentor, using each to update its density over models as just described. Efficient methods are used to update the agent's value function. Using this updated value function, it selects a suitable action, executes it, and repeats the cycle.

Like any RL agent, an imitator requires a suitable exploration mechanism. In the *Bayesian exploration* model [Dearden *et al.*, 1999], the uncertainty about the effects of actions is captured by a Dirichlet, and is used to estimate a distribution over possible Q-values for each state-action pair.⁶ Notions such as value of information can then be used to approximate the optimal exploration policy. This method is computationally demanding, but total reward including reward captured during training is usually much better than that provided by heuristic techniques. Bayesian exploration also eliminates the parameter tuning required by methods like ϵ -greedy, and adapts locally and instantly to evidence. These facts makes it a good candidate to combine with imitation.

4 Experiments

In this section we attempt to empirically characterize the applicability and expected benefits of Bayesian imitation through several experiments. Using domains from the literature and two unique domains, we compare Bayesian imitation to non-Bayesian imitation [Price and Boutilier, 1999], and to several standard model-based RL (non-imitating) techniques, including Bayesian exploration, prioritized sweeping and complete Bellman backups. We also investigate how Bayesian exploration combines with imitation.

First, we describe the agents used in our experiments. The Oracle employs a fixed policy optimized for each domain,

⁴Sampling is efficient as only one local model needs to be resampled at any time step.

⁵Scaling techniques such as those used in HMM's may be required to prevent underflow in the term $(n^{s,a,t})^{(c_m^{s,t})}$ in Eq. 5.

⁶The Q-value distribution changes very little with each update and can be repaired efficiently using prioritized sweeping. In fact, the Bayesian learner is cheaper to run than a full Bellman backup over all states.

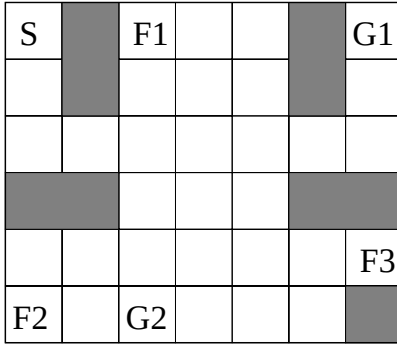


Figure 2: Flagworld Domain

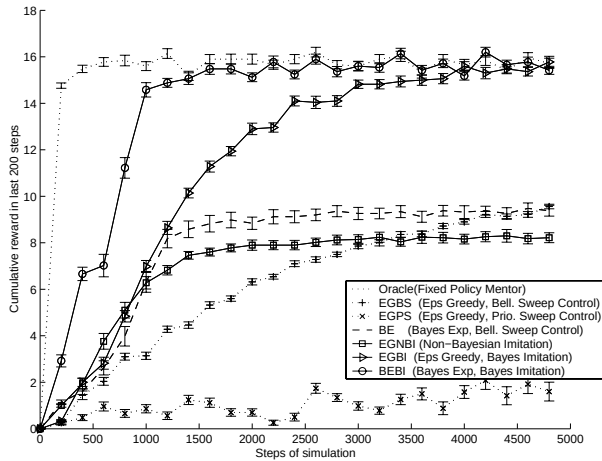


Figure 3: Flag world results (50 runs)

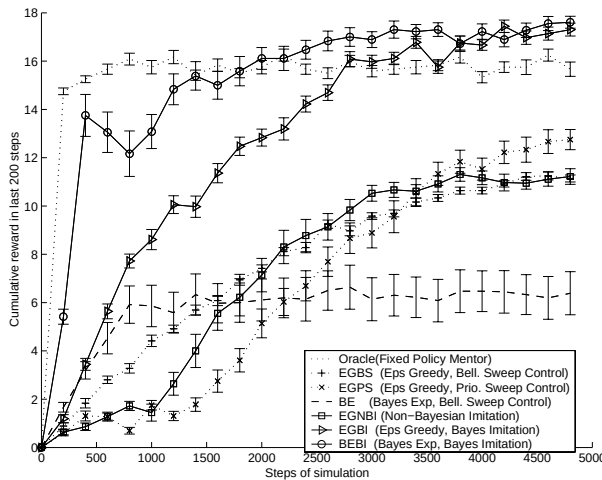


Figure 4: Flag World Moved Goal (50 runs)

providing both a baseline and a source of expert behavior for the observers. The EGBS agent combines ϵ -greedy exploration (EG) with a full Bellman backup (i.e., sweep) at each time step. It provides an example of a generic model-based approach to learning. The EGPS agent is a model-based RL agent, using ϵ -greedy (EG) exploration with prioritized sweeping (PS). EGPS use fewer backups, but applies them where they are predicted to do the most good. EGPS does not have a fixed backup policy, so it can propagate value *multiple* steps across the state space in situations where EGBS would not. The BE agent employs Bayesian exploration (BE) with prioritized sweeping for backups. BEBI combines Bayesian exploration (BE) with Bayesian imitation (BI). EGBI combines ϵ -greedy exploration (EG) with Bayesian imitation (BI). The EGNBI agent combines ϵ -greedy exploration with non-Bayesian imitation.

In each experiment, agents begin at the start state. The agents do not interact within the state space. When an agent achieves the goal, it is reset to the beginning. The other agents continue unaffected. Each agent has a fixed number of steps (which may be spread over varying numbers of runs) in each experiment. In each domain, agents are given locally uniform priors (i.e., every action has an equal probability of resulting in any of the local neighbouring states; e.g., in a grid world there are 8 neighbours). Imitators observe the expert oracle agent concurrently with their own exploration. Results are reported as the total reward collected in the last 200 steps. This sliding window integrates the rewards obtained by the agent making it easier to compare performance of various agents. During the first 200 steps, the integration window starts off empty causing the oracle's plot to jump from zero to optimal in the first 200 steps. The Bayesian agents use 5 sampled MDPs for estimating Q-value distributions and 10 samples for estimating the mentor policy from the Dirichlet distribution. Exploration rates for ϵ -greedy agents were tuned for each experimental domain.

Our first test of the agents was on the “Loop” and “Chain” examples (designed to show the benefits of Bayesian exploration), taken from [Dearden *et al.*, 1999]. In these experiments, the imitation agents performed more or less identically to the optimal oracle agent and no separation could be seen amongst the imitators.

Using the more challenging “FlagWorld” domain [Dearden *et al.*, 1999], we see meaningful differences in performance amongst the agents. In FlagWorld, shown in Figure 2, the agent starts at state *S* and searches for the goal state *G1*. The agent may pick up any of three flags by visiting states *F1*, *F2* and *F3*. Upon reaching the goal state, the agent receives 1 point for each flag collected. Each action (N,E,S,W) succeeds with probability 0.9 if the corresponding direction is clear, and with probability 0.1 moves the agent perpendicular to the desired direction. Figure 3 shows the reward collected in over the preceding 200 steps for each agent. The Oracle demonstrates optimal performance. The Bayesian imitator using Bayesian exploration (BEBI) achieves the quickest convergence to the optimal solution. The ϵ -greedy Bayesian imitator (EGBI) is next, but is not able to exploit information locally as well as BEBI. The non-Bayesian imitator (EGNBI) does better than the unassisted agents early on but fails

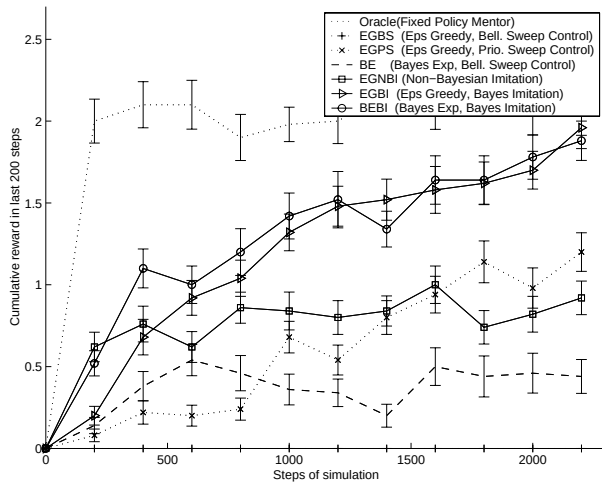


Figure 5: Tutoring domain results (50 runs)

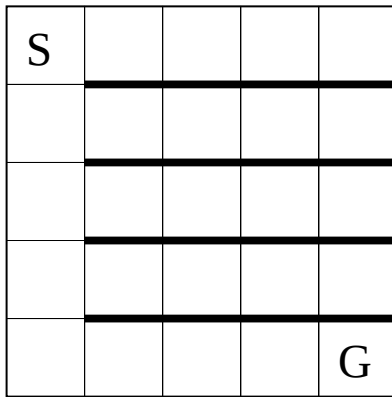


Figure 6: No-south domain

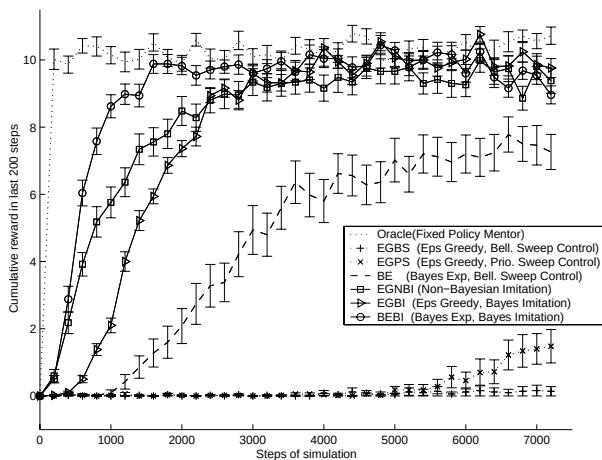


Figure 7: No South results (50 runs)

to find the optimal policy in this domain. A slower exploration rate decay would allow the agent to find the optimal policy, but would also hurt its early performance. The non-imitating Bayesian explorer fares poorly compared to the Bayesian imitators, but outperforms the remaining agents, as it exploits prior knowledge about the connectivity of the domain. The other agents show poor performance (though with high enough exploration rates they would converge eventually). We conclude that Bayesian imitation makes the best use of the information available to the agents, particularly when combined with Bayesian exploration.

We altered the FlagWorld domain so that the mentor and the learners had different objectives. The goal of the expert Oracle remained at location G1, while the learners had goal location G2 (Figure 2). Figure 4 shows that transfer due to imitation is qualitatively similar to the case with identical rewards. We see that imitation transfer is robust to modest differences in mentor and imitator objectives. This is readily explained by the fact that the mentor's policy provides model information over most states in the domain, which can be employed by the observer to achieve its own goals.

The *tutoring domain* requires agents to schedule the presentation of simple patterns to human learners in order to minimize training time. To simplify our experiments, we have the agents teach a simulated student. The student's performance is modeled by independent, discretely approximated, exponential forgetting curves for each concept. The agent's action will be its choice of concept to present. The agent receives a reward when the student's forgetting rate has been reduced below a predefined threshold for all concepts. Presenting a concept lowers its forgetting rate, leaving it un-presented increases its forgetting rate. Our model is too simple to serve as a realistic cognitive model of a student, but provides a qualitatively different problem to tackle. We note that the action space grows linearly with the number of concepts, and the state space exponentially.

The results presented in Figure 5 are based on the presentation of 5 concepts to a student. (EGBS has been left out as it is time-consuming and generally fares poorly.) We see that all of the imitators learn quickly, but with the Bayesian imitators BEBI and EGBI outperforming EGNBI (which converges to a suboptimal policy).⁷ The generic Bayesian agent (BE) also chooses a suboptimal solution (which often occurs in BE agents if its priors prevent adequate exploration). Thus, we see that imitation mitigates one of the drawbacks of Bayesian exploration: mentor observations can be used to overcome misleading priors. We see also that Bayesian imitation can also be applied to practical problems with factored state and action spaces and non-geometric structure.

The next domain provides further insight into the combination of Bayesian imitation and Bayesian exploration. In this grid world (Figure 6), agents can move south only in the first column. In this domain, the optimal Oracle agent proceeds due south to the bottom corner and then east across to the goal. The Bayesian explorer (BE) chooses a path based on its prior beliefs that the space is completely connected. The agent can

⁷Increasing exploration allows EGNBI to find the optimal policy, but further depresses short term performance.

easily be guided down one of the long “tubes” in this scenario, only to have to retrace its steps. The results in this domain, shown in Figure 7, clearly differentiate the early performance of the imitation agents (BEBI, EGBI and EGNBI) from the Bayesian explorer (BE) and other independent learners. The initial value function constructed from the learner’s prior beliefs about the connectivity in the grid world lead it to over-value many of the states that lead to a dead end. This results in a costly misdirection of exploration and poor performance. We see that the ability of the Bayesian imitator BEBI to adapt to the local quality of information allows it to exploit the additional information provided by the mentor more quickly than agents using generic exploration strategies like ϵ -greedy. Again, mentor information is used to great effect to overcome misleading priors.

5 Conclusions

Bayesian imitation, like the non-Bayesian implementation of implicit imitation, accelerates reinforcement learning in the presence of other agents with relevant knowledge without requiring either explicit communication with or the cooperation of these other agents. The Bayesian formulation is built on an elegant pooling mechanism which optimally combines prior knowledge, model observations from the imitator’s own experience and model observations derived from other agents. The combination of Bayesian imitation with Bayesian exploration eliminates parameter tuning and yields an agent that rapidly exploits mentor observations to reduce exploration and increase exploitation. In addition, imitation often overcomes one of the drawbacks of Bayesian exploration, the possibility of converging to a suboptimal policy due to misleading priors. Bayesian imitation can easily be extended to multiple mentors, and though we did not present the derivation here, it can also be extended to partially observable environments with known state spaces. Though the Bayesian formulation is difficult to implement directly, we have shown that reasonable approximations exist that result in tractable algorithms.

There are several very promising areas of future research that can benefit from the current formulation of Bayesian imitation. One obvious need is to extend the model to the heterogeneous action setting by incorporating the notions of feasibility testing and repair described in [Price and Boutilier, 2001]. We are particularly excited by the prospects of its generalization to richer environmental and interaction models. We have also derived one possible mechanism for using the Bayesian approach in domains with continuous attributes. We hope to extend this work to include methods for discovering correspondences between the state and action spaces of various agents. We also plan to introduce game-theoretic considerations into imitation so that agents can learn solutions to interacting tasks from experts and reason about both the reward-oriented aspects of their action choices as well as the information it reveals to others.

Acknowledgements

This research was supported by the Natural Sciences and Engineering Research Council and IRIS.

References

- [Atkeson and Schaal, 1997] C. G. Atkeson and S. Schaal. Robot learning from demonstration. In *Proc. Fourteenth Intl. Conf. on Machine Learning*, pp.12–20, Nashville, TN, 1997.
- [Dearden *et al.*, 1999] R. Dearden, N. Friedman, and D. Andre. Model-based Bayesian exploration. In *Proc. Fifteenth Conf. on Uncertainty in Artificial Intelligence*, pp.150–159, Stockholm, 1999.
- [Demiris and Hayes, 1999] J. Demiris and G. Hayes. Active and passive routes to imitation. In *Proc. AISB’99 Symposium on Imitation in Animals and Artifacts*, pp.81–87, Edinburgh, 1999.
- [Hu and Wellman, 1998] J. Hu and M. P. Wellman. Multiagent reinforcement learning: theoretical framework and an algorithm. In *Proc. Fifteenth Intl. Conf. on Machine Learning*, pp.242–250, Madison, Wisconsin, 1998.
- [Kaelbling, 1993] L. Pack Kaelbling. *Learning in Embedded Systems*. MIT Press, Cambridge, MA, 1993.
- [Kearns and Singh, 1998] M. Kearns and S. Singh. Finite sample convergence rates for Q-learning and indirect algorithms. In *Eleventh Conf. on Neural Information Processing Systems*, pp.996–1002, Denver, Colorado, 1998.
- [Kuniyoshi *et al.*, 1994] Y. Kuniyoshi, M. Inaba, and H. Inoue. Learning by watching: Extracting reusable task knowledge from visual observation of human performance. *IEEE Transactions on Robotics and Automation*, 10(6):799–822, 1994.
- [Littman, 1994] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proc. Eleventh Intl. Conf. on Machine Learning*, pp.157–163, New Brunswick, NJ, 1994.
- [Matarić, 2002] M. J. Matarić. Visuo-motor primitives as a basis for learning by imitation: Linking perception to action and biology to robotics. In *Imitation in Animals and Artifacts*, pp.392–422, Cambridge, MA, 2002. MIT Press.
- [Moore and Atkeson, 1993] A. W. Moore and C. G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, 13(1):103–30, 1993.
- [Nehaniv and Dautenhahn, 1998] C. Nehaniv and K. Dautenhahn. Mapping between dissimilar bodies: Affordances and the algebraic foundations of imitation. In *Proc. Seventh European Workshop on Learning Robots*, pp.64–72, Edinburgh, 1998.
- [Ng and Russell, 2000] A. Y. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *Proc. Seventeenth Intl. Conf. on Machine Learning*, pp.663–670. Morgan Kaufmann, San Francisco, CA, 2000.
- [Price and Boutilier, 1999] B. Price and C. Boutilier. Implicit imitation in multiagent reinforcement learning. In *Proc. Sixteenth Intl. Conf. on Machine Learning*, pp.325–334, Bled, SI, 1999.
- [Price and Boutilier, 2001] B. Price and C. Boutilier. Imitation and reinforcement learning in agents with heterogeneous actions. In *Proc. Fourteenth Canadian Conf. on Artificial Intelligence*, pp.111–120, Ottawa, 2001.
- [Sammur *et al.*, 1992] C. Sammur, S. Hurst, D. Kedzier, and D. Michie. Learning to fly. In *Proc. Ninth Intl. Conf. on Machine Learning*, pp.385–393, Aberdeen, UK, 1992.

MULTIAGENT SYSTEMS

MULTIAGENT SYSTEMS

Detecting & Avoiding Interference Between Goals in Intelligent Agents

John Thangarajah and Lin Padgham and Michael Winikoff

RMIT University

Melbourne, AUSTRALIA

{johthan,linpa,winikoff}@cs.rmit.edu.au

Abstract

Pro-active agents typically have multiple simultaneous goals. These may interact with each other both positively and negatively. In this paper we provide a mechanism allowing agents to detect and avoid a particular kind of negative interaction where the effects of one goal undo conditions that must be protected for successful pursuit of another goal. In order to detect such interactions we maintain summary information about the definite and potential conditional requirements and resulting effects of goals and their associated plans. We use these summaries to guard protected conditions by *scheduling* the execution of goals and plan steps. The algorithms and data structures developed allow agents to act rationally instead of blindly pursuing goals that will conflict.

1 Introduction

Pro-active intelligent agents are those agents that pursue goals [Thangarajah *et al.*, 2002a]. These agents typically have multiple goals that are active simultaneously. These goals can interact with each other both positively and negatively. For example consider a Mars rover agent that has a goal to perform soil analysis at location *A* and another goal to perform rock analysis at location *B*. Clearly the agent cannot pursue these goals in parallel since they require the agent to be at different locations. This is an example of negative interaction.

An intelligent agent should not blindly pursue conflicting goals or unintentionally make its own goals unattainable if such is easily avoidable. In earlier work [Thangarajah *et al.*, 2003] we provided mechanisms for identifying and facilitating positive interactions with respect to the effects of goals. Horthy and Pollack's work on evaluating an agent's options in the context of its existing plans [Horthy and Pollack, 2001] and Pollack's work on overloading intentions [Pollack, 1991] also contribute to managing positive interactions.

In pursuing their goals intelligent agents often use intermediate steps which enable later steps to achieve the goal. This is often managed using a series of plan steps where the effect of an earlier step achieves the pre-condition for a later step. A rational agent engaged in this kind of pro-active behaviour should not allow the pre-conditions that have been achieved

by an earlier step to be interfered with before the later step that relies on the pre-condition is done. For example suppose an agent is using a plan to cook dinner, involving going to the store, buying some items, coming home and preparing the meal. Having arrived at the location of the store, a rational agent would not allow a plan which took the agent elsewhere to intervene, prior to actually purchasing the items at the store.

This paper focuses on recognising situations where these linkages exist in a plan and controlling plan interleaving to ensure that an agent does not allow negative interference in the pursuit of separate parallel goals. We provide detailed mechanisms that can be easily implemented in agent platforms such as PRS [Ingrand *et al.*, 1992], JAM [Huber, 1999], dMARS [d'Inverno *et al.*, 1998], and JACK [Busetta *et al.*, 1998].

There has been significant work in the area of conflicts in agent systems [Tessier *et al.*, 2000]. However, the focus has been on multi-agents and identifying various types of conflicts. Our work focuses rather on providing algorithms and representations that allow detection and resolution of conflicts due to multiple parallel goals in a single agent.

There is also work in plan scheduling to avoid conflicts such as the work of [Boutilier and Brafman, 1997] and [Clement and Durfee, 1999b; 1999a]. Boutilier and Brafman provide a representation of concurrent interacting actions by extending the STRIPS action representation language, and provide an algorithm for planning with concurrent plans. However this work, like traditional planning approaches, requires the agent to have a completed plan, and perform the scheduling prior to execution (i.e. off-line). This is not useful for the type of agents that we deal with which are situated in highly dynamic environments where it is generally not possible to determine which plans to use in advance.

Clement *et al.* use the notion of summary information to co-ordinate plans at abstract levels. They have provided a formalism which defines correct interleaved plan execution [Clement and Durfee, 1999b] and a scheduling algorithm which uses this information to appropriately interleave plans to avoid resource conflicts [Clement *et al.*, 2001]. The work presented here focusses on recognising the way in which early steps in a plan typically pave the way for later steps, setting up "dependency links" and provides mechanisms to ensure that an agent recognizes these in managing its execu-

tion of parallel goals. We also provide mechanisms for continually updating summary information so that agents are also able to reason about partially completed goals.

In [Thangarajah *et al.*, 2002b] we provided means for an agent to detect and avoid resource conflicts by using summary information about resources.¹ In the current paper we use a notion of interaction summaries, based on the effect summaries developed by Clement and Durfee [Clement and Durfee, 1999b], containing information about effects, pre-conditions and in-conditions of goals and plans. Effects are the conditions that a goal or plan aims to achieve, pre-conditions are conditions that must be true for a plan to begin execution, and in-conditions are conditions that must remain true while the relevant goal or plan is active.

Specifically, we protect in-conditions and pre-conditions that have been achieved by earlier steps in a plan. We do not protect all pre-conditions: if the pre-condition has not been achieved by a previous step then there is no clear justification for protecting it. In particular, when defining the notion of *preparatory effects* in section 3, we only consider dependencies between nodes that are part of the same top-level goal.

In this work we consider only “achievement goals”², that is goals where the agent attempts to achieve some condition, and when this condition is achieved the goal is removed from the agent’s mental state.

2 Plan and Goal representation

This work builds on our previous work presented in [2002a; 2002b; 2003]. The representation of goals and plans are therefore similar to what we presented in [Thangarajah *et al.*, 2003]. To summarise these agents have a pre-defined set of plans which are used to satisfy goals at run-time. A goal-type is a template for a goal which consists of (i) a label that indicates the type of the goal, e.g. TransmitResultsGoal (ii) in-conditions of the goal³ (iii) the effects of the goal⁴ and (iv) the possible set of plan-types that can satisfy the goal [Winikoff *et al.*, 2002].

Similarly a plan-type is a template for a plan and consists of (i) a label for the type of plan (ii) pre-conditions for the plan to begin (iii) in-conditions that must be maintained during plan execution (iv) the direct effects of the plan and (v) a plan body which specifies what the plan does. Plan bodies can contain subgoals and actions. These are combined by either sequencing them (e.g. “achieve goal G_1 and then perform action A ” written as “ $G_1; A$ ”) or by performing them in parallel (e.g. “achieve goals G_1 and G_2 ” written as “ $G_1 \parallel G_2$ ”).

The pre-conditions include in-conditions⁵. The reason for this is that in-conditions are stronger than pre-conditions: if

¹This work was done independently from, but is quite similar to that of [Clement and Durfee, 1999b].

²Another type of goal is a “maintenance goal” where the agent ensures that a particular condition is never violated.

³Pre-conditions, in-conditions and effects are all represented as logical conditions.

⁴These effects are the direct, desired effects of the goal, similar to the notion of primary expectations of [Pollack, 1991].

⁵Formally, if i is the in-conditions then we consider the pre-conditions to be $i \wedge p$.

an in-condition fails to hold when a plan begins executing then the plan fails immediately.

Each goal-type G also has a *goal-plan tree* template where a goal-plan tree is a tree structure of goals whose children are the plans that achieve it and the children of a plan are the subgoals of the plan, with G as the root. All of the subgoals of a plan must be executed in order for the plan to succeed (“AND”). However, it is not necessary that all plans of a goal be executed (“OR”).

When the agent decides to pursue a goal, a goal-plan tree instance is created and updated at run-time to contain goal and plan instances. In addition to the above properties, goal and plan instances will also have a label (instance-name) that gives a unique handle to it at run-time. This is basically a path expression in a goal-plan tree instance (e.g. in figure 1 $G \rightarrow P \rightarrow SG3 \rightarrow P4$ is the instance-name for plan $P4$).

The execution cycle of the agent is similar to the well-known and developed BDI (Belief Desire Intention) [Rao and Georgeff, 1995] style of agents that map to agent implementation systems such as PRS, JAM, dMARS, and JACK, where a plan is selected from an applicable plan set and if it fails an alternative applicable plan is tried if available. We extend this model by requiring that a goal with an in-condition that is false is delayed until the in-condition becomes true⁶.

3 Preparatory Effects and Dependency links

Typically, when developing a set of plans to achieve a goal, a developer is not able to consider all the ways in which the pursuit of other goals may interfere with this goal. There are two important ways in which parallel goals can interfere, which we try to capture and reason about in this work. Considering the logical outcomes of goals and plans, interference can occur:

1. When an in-condition is made false while a plan or goal is executing, causing the plan or goal to fail.
2. When a previously achieved effect is made false before a plan or goal that relies on it begins executing, preventing the goal or plan from being able to execute.

We thus define a preparatory-effect (*p-effect*) of a goal G as follows. Consider two plans P_1 and P_2 which are utilised to satisfy G and P_1 is executed before P_2 as shown in figure 1. If P_1 brings about an effect e which is a pre-condition of P_2 , then there is a *dependency-link* between the *preparatory-effect* e and the *dependent-plan* P_2 ($e \rightarrow \{P_2\}$). We shall term this a *dependency-planlink*. In the event that an effect of a plan serves as a pre-condition to all the plans that satisfy a sub-goal (as is the case with effect e and sub-goal SG_3) then there is a dependency-link between the p-effect and the *dependent-subgoal* ($e \rightarrow \{SG_3\}$). We shall term this a *dependency-subgoal link*.

A dependency-planlink is *complete* when the dependent plan begins execution. A dependency-subgoal link is complete when either (a) the dependent-subgoal is complete or (b) the last possible plan to satisfy the sub-goal begins execution (all other plans have been tried but failed).

⁶Some of the systems will begin executing the goal and have it immediately fail.

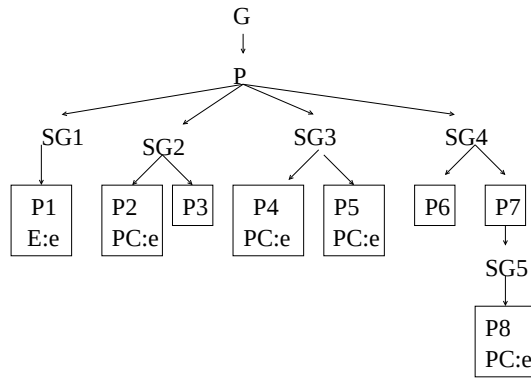


Figure 1: Dependency-links example

When a p-effect is achieved the agent should protect the effect from being undone until all the dependency-links associated with the effect are complete.

All dependency-planlinks are *potential* by definition because the agent has other means of achieving the dependent-plan's parent sub-goal.

A dependency-subgoalink is either

- *definite* if the dependent-subgoal will definitely be pursued as is the case with SG_3 (figure 1); or
- *potential* if the dependent-subgoal is not definite. SG_5 is an example of such a sub-goal as it depends on the path chosen for SG_4 .

The above categorisations leads to a ranking of importance with respect to guarding a p-effect. A definite dependency-subgoalink is strongest because the dependent-subgoal will definitely be pursued and all ways of achieving it require the p-effect as a pre-condition. A potential dependency-subgoalink is stronger than a dependency-planlink because although they are both potential the dependent-subgoal of the dependency-subgoalink will definitely fail if the p-effect is not true. In contrast the parent sub-goal of the dependent-plan of a dependency-planlink has other means of being achieved. This ranking is useful for defining agents that are more or less careful with respect to protecting preparatory effects from interference.

Dependency-entry:

In order to reason about and protect p-effects we attach dependency information at each plan-node of a goal-plan tree. This is done at compile time⁷ and each entry which we shall term a *dependency-entry* consists of a set of dependency-links, (*p-effect*, *dependent-plan/subgoal*), where the p-effect is an effect of the plan. We attach dependency-entries only to plans because p-effects are achieved by plans, goals only declare what the plans will achieve.

4 Interaction Tree

In order to detect if goals can be executed in parallel without any interference from each other we define the notion of inter-

⁷We do not describe the computation method in detail – the definitions above form a specification that can be easily realised.

action summaries. When there is potential interference these interaction-summaries help us to schedule appropriately.

Interaction-summaries are similar to effect-summaries developed in [Thangarajah *et al.*, 2003] and based on the work of Clement and Durfee [Clement and Durfee, 1999b] on condition summaries. The effect-summaries of a goal summarise the effects that will definitely be achieved irrespective of the path chosen to satisfy the goal and those that may be achieved depending on the path chosen. These were used in [Thangarajah *et al.*, 2003] to detect situations where the plans of two goals that bring about the same effect could be merged and to facilitate this merging.

We use interaction summaries to ensure that the effects of one goal do not interfere with active in-conditions and dependency-links of other goals. Therefore we need to maintain summary information of (a) effects as they could cause the interference, and (b) in-conditions and preparatory-effects as these are what need to be protected when active. Only those pre-conditions that are set-up by p-effects are protected so we do not need to maintain pre-condition summaries separate from the information in the dependency-entries.

For the discussion ahead the general term *condition* refers to either in-conditions or effects (including p-effects). Similarly to [Thangarajah *et al.*, 2003], conditions can be either *definite* or *potential*. A **definite condition** will *definitely* be achieved/required (achieved for effects and required for in-conditions) at some point along every possible path of achieving the goal (i.e. this condition need not be required by all plans, but is required by at least one plan in every possible way of achieving the goal). A **potential condition** will *possibly* be achieved/required at some point in the pursuit of the goal (i.e. this condition is part of at least one plan in at least one path of achieving the goal but **not** achieved/required in all possible paths of achieving the goal). The set of potential conditions and definite conditions are exclusive of each other.

Interaction Summary:

An *interaction-summary* of a goal/plan contains the definite and potential in-conditions, effects and p-effects. Formally the interaction-summary of a goal/plan n is a 3 tuple.

$$I_{summary}(n) = \langle S_i(n), S_e(n), S_{pe}(n) \rangle$$

where S_i , S_e and S_{pe} are the summaries of in-conditions, effects and preparatory-effects respectively. The in-conditions summary contains the definite and potential in-conditions of the node. $S_i(n) = \langle D_i, P_i \rangle$ and each of D_i and P_i are simply sets of conditions, $D_i = \{ic_1, ic_2, \dots, ic_n\}$. Similarly for the effects and p-effects summaries. We shall see how we can derive interaction-summaries of a goal/plan in section 4.1.

Interaction-Tree:

The *interaction-tree* structure is a goal-plan tree where attached at each node is the interaction-summary and in addition for plan-nodes the dependency-entry⁸ of the plan. This tree is built at compile time for each goal-type and instantiated at run-time for each goal-instance. Figure 2 shows the interaction tree of a simple goal of a Mars rover agent to perform soil experiments at location A .

The interaction-tree for any active goal is updated dynamically as the goal is pursued. When a plan/sub-subgoal com-

⁸as described at the end of section 3.

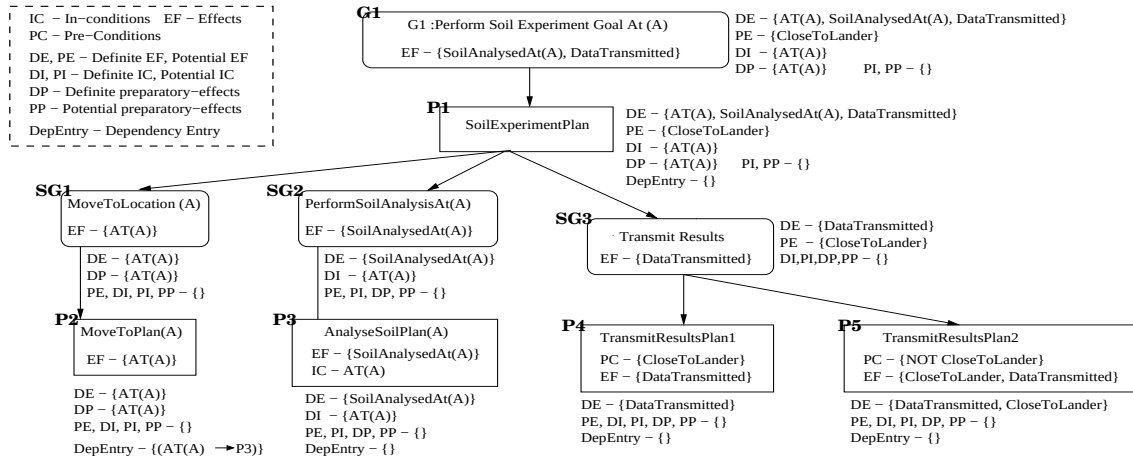


Figure 2: Interaction Tree example

pletes the corresponding node is removed and the interaction summary of the parent-node is re-computed. Any changes to the parent node are propagated up the tree. Dependency-entries are updated when plans (dependent-plans) begin execution.

4.1 Deriving Interaction Summaries

Interaction summaries are derived at compile time and updated at runtime, similarly to the mechanism used in [Thangarajah *et al.*, 2002b]. In this section we use the following notation: $G(P)$ is the set of subgoals of the plan P , and $in-cond(n)$ is the in-conditions of n (with n being either a goal or a plan).

The interaction-summary of a node of a goal-plan tree is derived by combining the local conditions of that node with the interaction-summaries of all its child nodes. Each piece of the summary (the in-condition summary, the effect summary and the p-effect summary) is calculated by combining the relevant pieces of the local and children nodes. The interaction summaries of goal nodes and plan nodes are calculated slightly differently and the details are as follows.

The **effects summary** of a plan is computed by taking the union of the local effects of the plan with the effects of each of the goals within the plan, using the addition operator $\oplus$ defined as follows:

$$\langle D_1, P_1 \rangle \oplus \langle D_2, P_2 \rangle = \langle (D_1 \cup D_2), (P_1 \cup P_2) \rangle$$

where D and P are respectively sets of definite and potential conditions (either effects or in-conditions).

The effects summary of a plan is then:

$$S_e(P) = \langle effects-of(P), \{\} \rangle \oplus \bigoplus_{g \in G(P)} S_e(g)$$

The **p-effect summaries** are analogous to effect summaries.

The **in-condition summary** of a plan is similar to the above and is defined as:

$$S_i(P) = \langle in-cond(P), \{\} \rangle \oplus \bigoplus_{g \in G(P)} S_i(g)$$

The summaries of goals however need to take into account that the plans whose conditions are being combined, are alternative ways to achieve a goal. The agent will typically⁹ only

⁹More than one plan may be executed in the case of plan failure. This does not change the rationale for the combination operators.

execute one of these alternative plans and thus the combined definite conditions are those that are definite for all plans, while the combined potential conditions are those potential for some plan plus those that are definite for some plan, but not for all. The conditions of plans are combined using a merge operator $\otimes$, defined as follows:

$$\langle D_1, P_1 \rangle \otimes \langle D_2, P_2 \rangle = \langle (D_1 \cap D_2), ((D_1 \cup D_2 - D_1 \cap D_2) \cup P_1 \cup P_2) \rangle$$

Goals do not have effects other than the effects of their plans, so the effect summary¹⁰ of a goal is:

$$S_e(G) = \bigotimes_{p \in plans-of(G)} S_e(p)$$

The in-condition summary of a goal is then defined as the merged in-conditions of its plan nodes, combined with the local in-conditions of the goal as follows:

$$S_i(G) = \langle in-cond(G), \{\} \rangle \oplus \bigotimes_{p \in plans-of(G)} S_i(p)$$

In order to determine the interaction-summary (IS) of a set of goals, the $\uplus$ operator is applied to all of the goals in the set as follows

$$IS(GoalSet) = \uplus_{g \in GoalSet} IS(g)$$

$$\text{where } IS(g_1) \uplus IS(g_2) = \langle S_i(g_1) \oplus S_i(g_2), S_e(g_1) \oplus S_e(g_2), S_{pe}(g_1) \oplus S_{pe}(g_2) \rangle.$$

5 Executing goals simultaneously

Having calculated interaction summaries, we then use these to detect and avoid potential interference. We will first look at ways of determining whether two goals will definitely not interfere with each other, in which case they can be pursued in parallel without any monitoring or scheduling of plans. Then we will discuss how goals that may have potential interference can be scheduled to avoid such conflicts whilst being pursued simultaneously.

As we have discussed in section 3, when two goals are executed in parallel, we track and protect the following two constraints: **(a)** the in-conditions of each goal and its active plan instances must not be violated and **(b)** the preparatory-effects (p-effects) that have been achieved must not be undone until the dependency-links are complete.

¹⁰The p-effect summary is similar

We can determine that a new goal G_{new} will not interfere with an existing set of goals $GSet$ if all (i.e. both definite and potential) the effects of G_{new} are compatible with all the derived in-conditions and p-effects of $GSet$. Two conditions¹¹ are compatible if it is possible for them to be simultaneously true. The details depend on the specific representation of conditions.

5.1 Scheduling

If there is interference between two goals this can be avoided by pursuing the goals in sequence. However, this restriction is too strong since it requires that the agent never have more than one goal that is being pursued. A more reasonable restriction is to pursue the goals in parallel, but monitor these goals for the steps which cause conflict and schedule them such that they do not interfere with each other. We do this by *guarding* the in-conditions and dependency-links that are active when adopting new goals and executing new plans.

In order to do this we use the interaction-trees and we further require a data structure that we shall call the *GuardedSet*¹² (GS) which includes in-conditions and dependency-links that are currently active and need to be guarded by the agent. Dependency-links are obtained from the dependency-entry of the associated plan-node from the interaction-tree. In-conditions are placed together with the name of the associated goal or plan instance. The GuardedSet can be defined by the following simple grammar:

$GuardedSet \rightarrow dependency\text{-}entry^* InCondEntry^*$

$dependency\text{-}entry \rightarrow p\text{-}effect PGName$

$InCondEntry \rightarrow in\text{-}condition PGName$

$PGName \rightarrow planInstanceName \mid goalInstanceName$

We avoid interference between goals by (i) protecting the in-conditions while a plan or goal is executing, and (ii) protecting the p-effects that are achieved by a plan from the effects of new plans until the relevant dependency-links complete. However, in doing this, we must be careful to avoid deadlocks. For example, imagine we have the following two goals:

$G_1.incondition(x), G_1 \rightarrow \dots \rightarrow subgoal1.effect(NOT y),$

$G_2.incondition(y), G_2 \rightarrow \dots \rightarrow subgoal2.effect(NOT x).$

Assume that the agent is pursuing G_1 , but has not yet reached *subgoal1*. The GS contains x , as it is an in-condition of G_1 . In considering G_2 , if the agent checks only the conditions in the GS there appears to be no problem and G_2 could be pursued, placing y also in the GS . However when *subgoal1* and *subgoal2* are respectively reached, they would be suspended as they affect y and x respectively, both of which are in the GS , thus creating a deadlock. Consequently, before adopting a goal/plan, we look ahead at effects that can be achieved by existing goals (using their interaction summaries) and ensure they don't interfere with the in-conditions of the new goal/plan to be executed.

Interference can then be avoided between goals by incorporating the following steps into the agent execution cycle.

1. When the agent **begins the pursuit of a goal** the in-conditions of the goal (i.e. the direct in-conditions, not

the derived in-conditions) are added to the GS .

2. When the agent **satisfies a goal** or the **goal fails** any in-conditions of the goal in the GS are removed. If the goal is part of a dependency-subgoallink in the GS this entry is also removed.
3. When a **plan begins execution**
 - (a) In addition to placing the in-conditions into the GS , if the plan has any p-effects then the relevant dependency-entry of the p-effects are placed in the *GuardedSet*. The reason we do this at the *start* of the plan, instead of the end (when the p-effect is achieved) is because the plan step that achieves the effect is not known in advance; and the agent should not begin another plan that could possibly undo the p-effect while the plan is executing.
 - (b) If the plan either completes a dependency-planlink or a dependency-subgoallink¹³ that is protected in the GS , then delete that dependency entry.
4. When a **plan completes or fails** execution, the in-conditions of the plan that are in the GS are removed. Further, in the event of plan failure any dependency-entries associated with p-effects of the plan are removed from the GS .

5. **Before the agent adopts a goal G_{new}** , it should perform the following to avoid deadlocks. Check for each goal G that has an associated condition in the GS , whether the in-conditions of G_{new} are compatible with the **combined definite and potential effects** of G .

If they are compatible then G_{new} can be pursued in parallel with G . Else check for compatibility with the **definite effects** of G .

If they are *not* compatible then unless G_{new} completes before G they will conflict. Since we have no notion of time G_{new} should not be pursued in parallel with G , but should be re-considered when the incompatible effects of G have been achieved.

Else the agent can either (a) choose to be cautious and not allow G_{new} to execute until the incompatible effects in G are achieved, or (b) since potential conditions depend on the path chosen, be optimistic and allow G_{new} to execute hoping that the incompatible effects (and resulting goal/plan failure) will be avoided.

6. **Before the agent executes a plan P_{new}** , it should check whether the effects of P_{new} are compatible with the conditions protected in the GS . If they are not compatible then the plan must wait until they are compatible. Also in order to prevent deadlocks the in-conditions and p-effects of the plan must be checked against the derived effects of existing goals that have a guarded condition as done before adopting a new goal.

Above we have given priority to the goal that is already being pursued as opposed to the new goal. This however is totally dependent on the priority mechanism of the application. If the new goal is more important than existing goals

¹¹More generally, sets of conditions

¹²Actually, this is a pair of multisets.

¹³By being the last plan to be attempted.

can be preempted. However goal priorities and mechanisms for goal preemption are beyond the scope of this paper.

6 Conclusion

In this paper we argued that intelligent agents often have plans involving several steps, where the results of one step set up conditions for a later step to be successful. Intelligent agents typically pursue multiple goals in parallel, but if intelligent or rational, they should manage this parallelism in a way which avoids undoing of preparatory effects too early. They should also not do something intentionally which causes immediate failure of another currently executing plan/goal.

We defined a notion of preparatory effects and dependency links and described how summary information could be computed and then used to protect both preparatory effects (before a plan/goal executes) and in-conditions (while a plan/goal executes). These mechanisms allow an intelligent agent to pursue multiple goals in parallel without stupidly and unnecessarily acting in a way that causes failure of some goal or plan. These mechanisms capture the kind of reasoning that is done effortlessly by humans, but which is not yet evident in intelligent computer agents.

Our analysis and experimentation indicates that this additional reasoning is not computationally expensive. However we plan to also test the costs and benefits empirically in further work. We also plan to do work to clarify the relationship between this work and earlier work by Clement and Durfee et al.

Acknowledgements

We would like to acknowledge the support of Agent Oriented Software Pty. Ltd. and of the Australian Research Council (ARC) under grant CO0106934. We would also like to thank Ed Durfee for bringing to our attention the similarities between this work and work previously published by his group.

References

- [Boutilier and Brafman, 1997] Craig Boutilier and Ronen I. Brafman. Planning with concurrent interacting actions. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97)*, 1997.
- [Busetta et al., 1998] Paolo Busetta, Ralph Rönquist, Andrew Hodgson, and Andrew Lucas. JACK Intelligent Agents - Components for Intelligent Agents in Java. Technical report, Agent Oriented Software Pty. Ltd, Melbourne, Australia, 1998.
- [Clement and Durfee, 1999a] Bradley J. Clement and Edmund H. Durfee. Identifying and resolving conflicts among agents with hierarchical plans. In *AAAI Workshop on Negotiation: Settling Conflicts and Identifying Opportunities*, AAAI Technical Report WS-99-12, 1999.
- [Clement and Durfee, 1999b] Bradley J. Clement and Edmund H. Durfee. Theory for coordinating concurrent hierarchical planning agents using summary information. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-99)*, pages 495–502, July 1999.
- [Clement et al., 2001] Bradley J. Clement, Anthony C. Barrett, Gregg R. Rabideau, and Edmund H. Durfee. Using abstraction in planning and scheduling. In *Proceedings of the Sixth European Conference on Planning (ECP-01)*, September 2001.
- [d’Inverno et al., 1998] Mark d’Inverno, David Kinny, Michael Luck, and Michael Wooldridge. A formal specification of dMARS. In *Intelligent Agents IV: Proceedings of the Fourth International Workshop on Agent Theories, Architectures, and Languages*, 1998.
- [Horty and Pollack, 2001] John F. Horty and Martha E. Pollack. Evaluating new options in the context of existing plans. *Artificial Intelligence*, 127:199–220, 2001.
- [Huber, 1999] Marcus J. Huber. JAM: A BDI-theoretic mobile agent architecture. In *Proceedings of the Third International Conference on Autonomous Agents (Agents’99)*, pages 236–243, May 1999.
- [Ingrand et al., 1992] F. F. Ingrand, M. P. Georgeff, and A. S. Rao. An architecture for real-time reasoning and system control. *IEEE Expert*, 7(6), 1992.
- [Pollack, 1991] M. E. Pollack. Overloading intentions for efficient practical reasoning. *Noûs*, 25(4):513–536, 1991.
- [Rao and Georgeff, 1995] Anand S. Rao and Michael P. Georgeff. BDI-Agents from theory to practice. In *Proceedings of the International Conference on Multi-Agent Systems ICMAS-95*, San Francisco, USA, 1995.
- [Tessier et al., 2000] Catherine Tessier, Laurent Chaudron, and Heinz-Jürgen Müller, editors. *Conflicting Agents: Conflict Management in Multi-Agent Systems*. Kluwer Academic Publishers, 2000. ISBN 0-7923-7210-7.
- [Thangarajah et al., 2002a] John Thangarajah, Lin Padgham, and James Harland. Representation and reasoning for goals in BDI agents. In *Proceedings of the Twenty-Fifth Australasian Computer Science Conference (ACSC 2002)*, Melbourne, Australia, 2002.
- [Thangarajah et al., 2002b] John Thangarajah, Michael Winikoff, Lin Padgham, and Klaus Fischer. Avoiding resource conflicts in intelligent agents. In *Proceedings of the 15th European Conference on Artificial Intelligence 2002 (ECAI 2002)*, Lyon, France, 2002.
- [Thangarajah et al., 2003] John Thangarajah, Lin Padgham, and Michael Winikoff. Detecting and exploiting positive goal interaction in intelligent agents. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2003)*, Melbourne, Australia, 2003.
- [Winikoff et al., 2002] Michael Winikoff, Lin Padgham, James Harland, and John Thangarajah. Declarative & procedural goals in intelligent agent systems. In *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR2002)*, Toulouse, France, April 2002.

Behavior Bounding: Toward Effective Comparisons of Agents & Humans*

Scott A. Wallace and John E. Laird

Artificial Intelligence Laboratory

University of Michigan

Ann Arbor, MI 48109

Abstract

In this paper, we examine methods for comparing human and agent behavior. The results of such a comparison can be used to validate a computer model of human behavior, score a Turing test, or guide an intelligent tutoring system. We introduce behavior bounding, an automated model-based approach for behavior comparison. We identify how this approach can be used with both human and agent behavior. We demonstrate that it requires minimal human effort to use, and that it is efficient when working with complex agents. Finally, we show empirical results indicating that this approach is effective at identifying behavioral problems in certain types of agents and that it has superior performance when compared against two benchmarks.

1 Introduction

Over the past twenty years, AI research has successfully demonstrated a number of techniques for constructing agents that exhibit intelligent behavior. Many applications have the additional requirement that the agent's behavior be consistent with that of a human expert. This is especially true for tasks in which the agent must simulate a human such as in training situations or in virtual social experiences such as on-line gaming.

For these tasks, the standard approach to developing expert level agents begins with knowledge acquisition. Unfortunately, knowledge acquisition is usually imperfect. As a result, significant resources must be spent on validation, which often requires both a knowledge engineer and domain expert to monitor the agent's behavior in a large number of test scenarios. In this paper, we present a method for automatically comparing two actors' behavior that could be used to overcome this validation bottleneck.

The potential uses of automated behavior comparison extend well beyond knowledge-base validation. For example, a generalized approach for comparing two actors' behavior can be used to objectively score a Turing test. A perfect result would be indicated if no detectable differences between the

human and its computer counterpart were identified. In addition, automated behavior comparison can serve as the core of an intelligent tutoring system, where the roles are reversed. A novice human's behavior is compared to a computer agent's behavior which serves as a gold standard and information about the student's errors is used to guide the lesson. In all of these applications, the basic process for comparing behavior is identical. The differences only stem from the source of behavior (e.g. human or machine, expert or novice) and how the results of the comparison are used (to identify programming errors, to score a test or to guide a lesson). For simplicity and cohesiveness, this paper will focus on using behavior comparisons to aid the knowledge-base validation problem, but the discussion and results can also be applied to the other tasks as well.

2 Interactive Human-Level Agents

The need for behavior comparisons is particularly pronounced when the agent must masquerade as its human counterpart (the expert). These agents, which we term *interactive human-level agents*, are distinguished by two properties. First, the agent's performance is judged based on its ability to behave as the human expert would behave. Secondly, like humans themselves, interactive human-level agents must interact with an external environment in order to perform many of their tasks.

A good example of an interactive human-level agent is TacAir-Soar [Jones *et al.*, 1999]. TacAir-Soar flies virtual military planes as part of a simulated training exercise. Teammates may be other TacAir-Soar agents or human counterparts. Because the agents are intended to model expert level behavior, it is not acceptable just to achieve the final states (e.g. shooting down the enemy planes). Instead, the agent must generate the same behavior as the expert. Meeting this requirement is challenging because the expert may perform the task differently on different occasions.

In the remainder of this paper, we first begin by examining a simple method of comparing a computer agent's behavior to a human expert's behavior. Deficiencies with this method lead us to examine more sophisticated model-based approaches. In Section 4 we summarize desirable features of such an approach. Then, beginning in Section 5, we present our method of model-based behavior comparison.

*This work was supported by the Office of Naval Research, contract N61339-99-C-0104

3 Toward Automated Behavior Comparison

Before two actors' behavior can be compared, the behavior must be represented in a form that can be processed by the comparison algorithm. We can do this most easily by storing individual instances of behavior, or behavior traces. A behavior trace is a sequence of tuples $B = ((s, G, a)_0, (s, G, a)_1, \dots, (s, G, a)_n)$ in which each tuple $(s, G, a)_i$ indicates the environmental state (s), the goals being pursued by the actor (G), and the action being performed (a). The state and action portion of the behavior trace can be captured by observing the actor perform the specified task. The actor's goals are necessary to disambiguate instances when different actions are performed in equivalent environmental states. Depending on whether the actor is human or computer agent, the actor may need to record how their goals change during the task.

A simple approach to comparing the actor's behavior can be performed with the following steps:

Acquire a set of behavior traces from the human expert and the agent for the specified task. These sets, H and A , represent the human expert's and agent's behavior respectively over a number of different trials.

Extract relevant features from the behavior traces. Some information gathered through observation may not be useful to detect errors. In this step, the salient features from the sets H and A are used to create two new sets of sequences H^* and A^* .

Compare each sequence $a \in A^*$, to the contents of H^* . Compute the minimal number of edit operations (insert, delete, modify) that would be required to transform a into h , where h is the sequence in H^* that is most similar to a . Each edit operation indicates a potential error.

Report all deviations (after removing any redundancies) between the human's and agent's behavior. This report summarizes all potential errors.

This simple approach performs a more detailed analysis of behavior than simply checking that the agent and the expert reach the same final (goal) state. In this way, the agent's externally observable behavior as well as some aspects of its internal reasoning process can be inspected to ensure that it is consistent with the human expert's. In addition, this methodology has the ability to identify a large number of possible errors because it has access to all the salient properties of the behavior trace.

However, this simple approach also suffers from a number of potentially serious flaws. First, the representation of the actors' behavior is a set of sequences extracted from the behavior traces. These sets grow as more observations are considered. Because interactive human-level agents can typically solve problems in a number of different ways, and because the environments they operate within are complex, it is likely that a very large number of observations will be required to adequately cover the actor's behavior. This problem is exacerbated by the fact that the sequential representation makes no assumption about how the actor's behavior might be constrained. Although this makes it possible to use this simple approach with any variety of behavior, it also makes

it impossible to leverage regularities that might exist in large classes of goal directed tasks.

4 Model Based Approaches

To improve upon the simple approach for automated error detection described in Section 3, we propose a model-based approach to comparing actors' behavior. Central to any such approach are the properties of the behavioral model. Our choice is guided by the following requirements:

Low Complexity Unless the new model is significantly less complex than the agent's knowledge base, understanding the model and the behavior it represents is no easier than examining the knowledge base directly. For the model to be an asset, it must provide an adequately accurate representation of behavior while remaining easy to understand.

Low Effort We have argued that one of the main uses of the behavior comparison is to reduce the cost of validating a human-level agent. In order to accomplish this goal, the human effort required to build the behavioral models must remain low.

Compatibility Behavior comparison has a number of potential applications, but most rely on being able to examine both human and software agent behavior. Thus the representation must be limited to data that can be collected from either of these types of participants.

Efficiency Human-level agents operate in complex environments and may perform their tasks in a variety of different ways. To address this problem, a model may be built using observations of expert behavior. In this case, it must be possible to generate the model efficiently, even if many observations are required.

Efficacy Meeting the preceding requirements will come at a cost. Most likely this will be a decreased ability to distinguish between some types of behavioral deviations (potential errors). A good representation will nonetheless be able to identify a wide range of behavioral deviations that are likely to occur within the target environments and overlook meaningless differences.

Prior work in model-based diagnosis (e.g. [Lucas, 1998]) has examined how to detect errors given a model of correct behavior. In general, however, the models in these systems are relatively complicated and intended to identify problems with mechanical or solid state devices as opposed to software agents. However, one system, CLIPS-R [Murphy and Pazani, 1994] was designed expressly for validating software agents.

In CLIPS-R, the behavior model consists of environmental constraints that must be met initially, as well as during and after task execution. In addition, the model can include a finite state machine which identifies acceptable sequences of actions pursued by the agent. Superficially, the requirements for the CLIPS-R approach seem relatively simple to meet. However, specifying this additional knowledge is a manual process that can significantly increase human effort and ironically can introduce a recursive validation problem for the constraints.

5 Behavior Bounding

As an improvement to CLIPS-R and to the simple method presented in Section 3, our approach to behavior comparison, called behavior bounding, automatically and efficiently builds concise models of both the human's and agent's behavior by examining behavior traces. The model of the expert's behavior is used to identify boundaries on acceptable behavior, and potential errors are reported by comparing the model of agent behavior to these boundaries.

5.1 A Hierarchical Model

The advantages of behavior bounding all stem from its representation of behavior. Behavior bounding is inspired by the hierarchical representations used in AND/OR trees, HTN planning [Erol *et al.*, 1994] and GOMS modeling [John and Kieras, 1996] to encode the variety of ways in which particular tasks can be accomplished.

The hierarchical behavior representation (HBR) used in our approach is illustrated in Figure 1A. The hierarchy is an AND/OR tree with binary temporal constraints representing the relationships between the actor's goals and actions. In this representation, internal nodes correspond to goals and leaves correspond to primitive actions. A node's children indicates the set of sub-goals or primitive actions that are relevant to accomplishing the specified goal. For example, in Figure 1A, the sub-goals **Destroy-Lead** and **Destroy-Wingman** are relevant for completing their parent goal, **Engage-Enemy**. The manner in which sub-goals should be used to achieve their parent goal is encoded by the parent's node-type constraint (AND vs OR) and the ordering constraints between sub-goals. In Figure 1A, AND and OR nodes are represented with ovals and rectangles respectively. Binary temporal constraints are represented with arrows between siblings. Thus, the hierarchy specifies that **Engage-Enemy** may be correctly accomplished by first accomplishing **Destroy-Lead** and then accomplishing **Destroy-Wingman**.

This model of behavior is clearly less complex than the agent's underlying knowledge base. Indeed, it is likely to be less complex than the model used by CLIPS-R. Behavior bounding abstracts away internal data-structures the agent may use in problem solving that cannot be represented by the constraints in the hierarchy. This means, that the HBR alone could not be used to perform some basic tasks such as depth first search. This begs the question, if the agent's behavior can be represented using such a simple structure, why was it not programmed in this representation to begin with? The hypothesis here is not that this representation is sufficient to *completely* capture the agent's behavior. Most human-level agents do rely on intermediate data-structures that are not available through the environment or through the structure of the goal hierarchy. However, our hypothesis is that the representation provided by behavior bounding is sufficient to identify a large class of possible errors in agent behavior without sacrificing efficiency. Moreover, we believe that behavior bounding can also help identify potential problem spots in the agent's knowledge (e.g. a specific goal) even if an exact error cannot be identified.

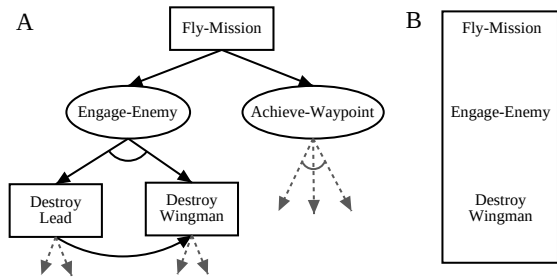


Figure 1: Hierarchical Behavior Representation & Goal Stack

In contrast to the behavior representations used for the simple comparison described in Section 3, the HBR makes two strong assumptions about the organization of the actors' knowledge and the effects this will have on their behavior. These assumptions increase the efficiency and efficacy of error detection for certain types of human-level agents.

The first assumption used by the behavior bounding approach is that the actor's goals are organized hierarchically, with more abstract goals placed toward the top of the tree. We also assume that at any point in the problem solving process the actor pursues a set of goals belonging to different levels in the hierarchy. This set, referred to as the goal stack, corresponds to a path in the hierarchy beginning at the top node and descending to the most concrete sub-goal that is currently being pursued by the actor. Figure 1B illustrates a possible goal stack maintained by the actor whose behavior is represented in Figure 1A.

The second assumption leveraged by behavior bounding relates to the independence of goals. Temporal constraints can only be formed between sibling nodes, and AND/OR classification determines which of a node's children must be performed for a particular task. This makes it is easy to constrain the way a particular goal is achieved, but difficult to represent constraints between arbitrary parts of the hierarchy. Although this may cause problems with some agent implementations, this property has significant benefits. Most importantly, it decreases the number of observations that are required. Consider a task that requires completing two goals, each of which could be fulfilled in four distinct ways. A sequential representation that makes no assumptions about goal independence (such as the one described in Section 3) would require sixteen distinct observations to cover the acceptable behavior space where as behavior bounding would only require four observations. This significant impact on efficiency is the direct result of leveraging the assumption about how goals are likely to add regular structure to an actor's behavior.

5.2 Identifying Errors

In general, we can view a behavior comparison method as an algorithm which divides the space of possible behaviors into two regions: behaviors that are likely to be consistent with the expert, and behaviors that are likely to be inconsistent with the expert. The simple comparison method described in Section 3 does this by enumerating consistent behaviors.

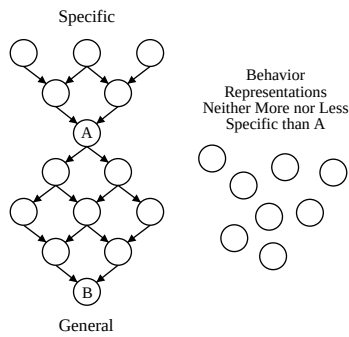


Figure 2: Imposing Order on the Behavior Space

In behavior bounding, however, the constrained hierarchical representation allows us to break the space of possible behaviors into more refined regions.

We begin by noting that the constrained hierarchical representation allows us impose order on the space of possible behaviors. In particular, we can define an ordering from specific to general over the behavior hierarchies, by starting with a maximally constrained hierarchy (at the top) and iteratively removing constraints until none remain. Constructing a representation of an expert's behavior (Section 6) performs this same generalization, but most often stops before all constraints have been removed. Figure 2, in which each node represents a behavior hierarchy, illustrates this ordering. Once we have created a representation for the expert's behavior, we can identify the node it occupies in this ordered space (call this node A in Figure 2). This node (the upper boundary node) allows us to easily determine if the agent's behavior is likely to be correct. Because correct behavior must be consistent with expert behavior, an agent whose behavior representation is a specialization of the expert's (i.e. lies above A in the generalization lattice) exhibits behavior that is likely to be correct.

The node that represents the completely unconstrained goal hierarchy is at the bottom of Figure 2 (labeled B) and provides a lower boundary. It contains the most basic specification for what may constitute acceptable agent behavior and as a result could be used to identify behavior representations that are known to be incorrect. Such representations would have a goal decomposition structure that was inconsistent with (i.e. contained different parent/child relationships than) this lower boundary (nodes in the right side of Figure 2).

Using the upper and lower boundaries described above we can classify any representation of agent behavior as: likely-correct (a specialization of the expert's behavior representation); likely-incorrect (a specialization of the expert's goal decomposition structure); and known-incorrect (inconsistent with the expert's goal decomposition structure).

Clearly, the hierarchical representation used by our approach describes behavior at a much higher level of abstraction than a typical knowledge base and in so doing, it presents a much more concise illustration of potential behavior than, for example, a set of individual rules. As a result, it meets our first requirement (low complexity). In addition, this representation can be generated automatically by examining behavior

traces of an actor's performance on a task thus meeting the second requirement (low human effort). And because the behavior traces can be captured from either human or agent behavior with only minor support from the human participant, this model meets the third requirement. In the following sections, we will examine the remaining requirements in detail.

6 Learnability

In this section, we examine two aspects of behavior bounding's hierarchical representation: the effort required to create and maintain it, and its ability to represent behavior efficiently. Both of these requirements are addressed by the overall learnability of the representation. That is, if the representation can be learned from observations (as we have suggested), then it requires human effort only to initiate the learning process. If the learning procedure is efficient, and the data structure's growth is limited, we can further say that the hierarchy represents behavior efficiently.

The learning procedure for constructing the HBR extracts goal stacks and actions from a behavior trace, forming a hierarchical structure such as the one illustrated in the previous section. After processing the first behavior trace, the hierarchy contains the maximum number of constraints (i.e. AND/OR constraints on the goals and binary temporal constraints between siblings) that are consistent with the behavior in the trace. So, if each goal in the hierarchy is pursued only once while performing the task, all internal node-types are AND (maximally constrained) and all sibling internal nodes are totally ordered (again, maximally constrained).¹ Upon examining subsequent behavior traces, the hierarchy is generalized in such a way that it remains maximally constrained with respect to all of the behavior traces it has processed.

Due to page limitations, we cannot present the learning algorithm in detail, however it should be clear that the hierarchy can be built as described above with complexity $O(L|N|^2)$, where $|N|$ is the size of the goal hierarchy and L is the length of the behavior trace. In most cases, it is reasonable to assume that one property of expert quality behavior is completion of the task within a number of steps proportional to $|N|$. When this assumption holds, we can say that this algorithm is bounded by $O(|N|^3)$ in time and space, with respect to the size of the input (i.e. the length of the behavior trace). Because this complexity is a low order polynomial of $|N|$, the hierarchy is efficient when encoding an instance of behavior.

We can also classify the sample complexity of our hierarchical representation. We can think of our representation as an ordered tuple $P = (p_1, p_2, \dots, p_{|N|})$ where each p_i is itself a tuple containing the type of the node i (either AND or OR), as well as a list $L = (l_1, l_2, \dots, l_{|N|})$ such that $l_a = 1$ iff g_i is ordered before l_a . Note that since ordering constraints only occur between siblings, the length of the list L would only need to be length $|N|$ in the degenerate case. The size of this hypothesis space is bounded by $2^{|N|+|N|^2}$. Using Haussler's equation [Haussler, 1988], the number of

¹The leaves, representing primitive actions, will only be totally ordered if each action was used only a single time to achieve its parent goal.

training examples m required to learn the appropriate behavior representation is bounded by:

$$m \geq \frac{1}{\epsilon} \left((|N|^2 + |N|) \ln(2) + \ln \left(\frac{1}{\delta} \right) \right) \quad (1)$$

This indicates that the required sample size is polynomial with respect to the number of goals in the hierarchy ($|N|$). This, together with the fact that the time required to incorporate a new behavior trace into the learned HBR is also polynomial in $|N|$, shows that our representation is PAC-Learnable. This means that the HBR efficiently represents aggregate behavior as well an individual instance of behavior, thus meeting our fourth requirement.

7 Efficacy

The efficacy of behavior bounding is addressed by two components. First, how good is the unconstrained hierarchical representation (the lower boundary) at identifying behavior that is known to be incorrect. Second, how well does the expert's representation (the upper boundary) serve to distinguish between potentially correct and incorrect behavior.

At first glance, it is not obvious how much behavior can be filtered by the lower boundary. However, its effectiveness as a filter is quite surprising. Consider an unconstrained behavior representation with branching factor b and depth d . Without loss of generality, assume that the nodes are uniquely labeled. For simplicity, also assume that at any level in this hierarchy, the actor completes its current goal before starting the next goal. Then, we could define an actor's behavior as a sequence of symbols chosen from the lowest level of the unconstrained hierarchy. For behavior sequences of length b^d , in which no symbol is repeated, there are $b!^s |s| = \sum_{j=0}^{d-1} b^j$ possible sequences that are consistent with the goal decomposition of the unconstrained hierarchy. In contrast, there are $b^d!$ sequences in which the symbols may be placed without necessarily conforming to the unconstrained hierarchy. For a hierarchical structure of depth 4 and branching factor 2, only 1 in approximately $6.4 \cdot 10^8$ of the possible sequences of length 16 are consistent with the goal decomposition specified by the unconstrained hierarchy. This illustrates the potential power of the lower behavior boundary to discriminate between behavior that is potentially correct and the large collection of behavior that is inconsistent with the expert's goal decomposition structure, and thus known to be incorrect.

To examine how well the expert's representation distinguishes between correct and incorrect behavior we would ideally examine a large set of hand-programmed agents before they have been validated. Unfortunately, this is not feasible. Instead, we make random modifications to an agent's knowledge base. These modifications introduce unbiased behavioral flaws in the agent program, and experiments are performed to determine how well each type of error is identified. Because the modifications are made randomly, the errors that are examined will not be biased by our expectations about how easily they will be to identified.

Our experiments are performed on a series of agents within a simulated object-retrieval environment. The object-retrieval task requires both planning and reactive reasoning. Initially,

when given the task, the agent must plan a route through known territory to a building thought to contain the desired artifact. Because the agent has no prior knowledge of the building's layout, it must explore the facility until the object is found, and then find its way back out. The task is complete once the agent leaves the building with the object. A behavior comparison metric's performance is judged based its ability to correctly identify errors in agent behavior, to identify *all* errors that have occurred, and to produce minimal amounts of spurious information in the report.

7.1 Methodology

We implemented the algorithm described in Section 6, along with two version of the simple approach described in Section 3 to serve as benchmarks. The first benchmark, the action sequence, extracts the sequence $A = (a_0, a_1, \dots, a_n)$ from the behavior trace $B = ((s, G, a)_0, (s, G, a)_1, \dots, (s, G, a)_n)$ while the second benchmark extracts the sequence of goals $G = (G_0, G_1, \dots, g_n)$ from B . Remember that the benchmarks are not particularly efficient representations; they can grow exponentially and have an exponential sample complexity. However, they do make interesting benchmarks of efficacy.

We initially constructed an agent that solves the problem in a very rigid manner. That is, across different attempts, the agent will complete the task using identical behavior so long as it is provided identical initial states and so long as the environment responds identically to its behavior. Given this agent, we performed modifications on its knowledge-base by randomly removing rules that determine preferences between competing goals and actions. The results of these modifications are agents that complete the task successfully in the traditional sense (i.e. they reach the same end state), but have increased flexibility in terms of the sequence of goals and actions they use to achieve that final state. In addition, the behavior exhibited by these modified agent's *cannot* be classified as incorrect by the lower behavior boundary node. As a result, these tests directly examine the abilities of the upper boundary node to distinguish between correct and incorrect behavior.

Each family of experiments begins by selecting two agents, e and n , such that n is a modified (more flexible) version of e . We designate e as the expert, and n as the novice. Because n is more flexible than e , it will behave in certain ways that are not consistent with expert behavior—these are errors.

After the expert and novice have been selected, they are individually incorporated into a simulation so their behavior can be observed. We then gather between 10 and 15 behavior traces of the actors performing their task, ensuring that no two behavior traces are identical. These traces form the sets BT_E and BT_N for the expert and novice respectively. Finally, each behavior trace in BT_N is examined manually to determine what errors it contains.

The captured behavior traces are then split into a number of subsets: $n_i \subset BT_N$ and $e_i \subset BT_E$. A single experiment consists of examining each comparison method's performance on a pair of these subsets (n_i and e_j). A *family* of experiments contains the experiments that compare all n_i to all e_j for a particular novice/expert pair. Thus comparing

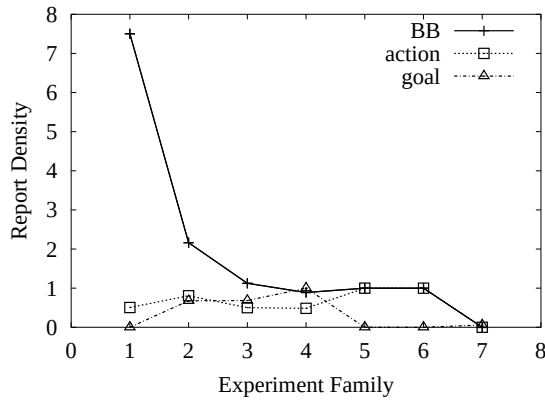


Figure 3: Report Density

four expert/novice pairs results in four experiment families although the total number of individual experiments may be much larger. By constructing experiment families in the way, we are able to examine the impact of different observational data more easily.

7.2 Results

For each experiment, we begin with a record (summarized by our manual examination of the agent's behavior traces) of what behavior errors were committed by the agent; these are true errors. This record identifies both low-level true errors such as omissions, commissions, and intrusions, as well as higher-level true errors such as misplacement, or repetition which are the result of multiple low-level errors. We then collect a summary from each comparison method about how the agent's behavior differed from the expert's behavior. Based on this, we record the number of true errors that occurred but were not identified by the summary (false negatives). Then, for each behavioral deviation in the summary we manually determine whether it indicates a new true error in the agent's behavior. If so, this deviation is meaningful and thus a useful part of the report, otherwise it is spurious.

To judge the performance of a comparison method, we first determine how many errors it fails to identify (false negatives). From this standpoint, all comparison methods performed relatively similarly. Across all experiment families the range on false negatives is between zero and 1.75 for all comparison metrics. However, in each experiment family behavior bounding does perform slightly better than the rest, maintaining the lowest average score, and achieving zero false negatives in five experiment families.

The other performance criteria we are interested in is how much of the comparison method's summary is useful. Because the summary is likely to be analyzed by a human, spurious errors are undesirable because they require human effort to examine and follow-up before they can be dismissed. If a summary is largely spurious errors, it is unlikely that anyone would actually care to read it. On the other hand, if the summary provides a large amount of good information, an occasional spurious error is likely to be acceptable.

This can be expressed as a ratio of how many true errors

can be identified by the deviations listed in the summary to the total number of deviations listed in the summary. We call this value the report density of the summary. Thus, a comparison method that reports no meaningful deviations, and only spurious information will have a report density of zero. Note that it is possible for the report density to be higher than one. When a deviation in the summary correctly identifies a high-level error (such as the agent performing action *a* before *b* in contrast to the expert who performs *b* before *a*) we consider the summary as also identifying all the low-level errors that form this high-level error (e.g. two commissions in which *a* replaces *b* and vice-versa). This means that from the point of view of report density, it is better to identify high-level errors than low level errors. Because we should favor *concise* summaries, this is exactly what we want. Figure 3 illustrates the average report density of each comparison method's summary over all experiment families. Overall, behavior bounding outperforms the other metrics by achieving an report density of one or higher in six of the seven experiment families.

7.3 Conclusions

Our behavior bounding method uses an abstract model of behavior to identify potentially problematic differences between two actors. This in turn leads to a semi-automated method that can be used to validate complex, human-level behavior. Moreover, behavior bounding offers significant advantages over prior validation approaches and traditional manual techniques: it can be easily be used to validate an agent when human performance is the specification for acceptability; it requires only minimal human effort to initiate and perform the behavior comparisons, and behavior bounding is effective at identifying errors, even when compared to methods that are not constrained by efficiency requirements.

References

- [Erol *et al.*, 1994] K. Erol, J. Hendler, and D. S. Nau. HTN planning: Complexity and expressivity. In *Proc. of the 12th National Conf. on Artificial Intelligence*, pages 1123–1128. AAAI Press/MIT Press, 1994.
- [Haussler, 1988] D. Haussler. Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence*, 36:177–221, 1988.
- [John and Kieras, 1996] B. E. John and D. E. Kieras. The GOMS family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer-Human Interaction*, 3(4):320–351, 1996.
- [Jones *et al.*, 1999] R. M. Jones, J. E. Laird, P. E. Nielsen, K. J. Coulter, P. Kenny, and F. V. Koss. Automated intelligent pilots for combat flight simulation. *AI Magazine*, 20(1):27–42, 1999.
- [Lucas, 1998] P. Lucas. Analysis of notions of diagnosis. *Artificial Intelligence*, 105:295–343, 1998.
- [Murphy and Pazzani, 1994] P. M. Murphy and M. J. Pazzani. Revision of production system rule-bases. In *Proc. of the 11th Int. Conf. on Machine Learning*, pages 199–207. Morgan Kaufmann, 1994.

Characterization of Strategy/False-name Proof Combinatorial Auction Protocols: Price-oriented, Rationing-free Protocol

Makoto Yokoo

NTT Communication Science Laboratories

2-4 Hikaridai, Seika-cho

Soraku-gun, Kyoto 619-0237 Japan

email: yokoo@cslab.kecl.ntt.co.jp

url: <http://www.kecl.ntt.co.jp/csl/ccrg/members/yokoo>

Abstract

This paper introduces a new distinctive class of combinatorial auction protocols called price-oriented, rationing-free (PORF) protocols. The outline of a PORF protocol is as follows: (i) for each bidder, the price of each bundle of goods is determined independently of his/her own declaration (while it can depend on the declarations of other bidders), (ii) we allocate each bidder a bundle that maximizes his/her utility independently of the allocations of other bidders (i.e., rationing-free).

Although a PORF protocol appears quite different from traditional protocol descriptions, surprisingly, it is a sufficient and necessary condition for a protocol to be strategy-proof. Furthermore, we show that a PORF protocol satisfying additional conditions is false-name-proof; at the same time, any false-name-proof protocol can be described as a PORF protocol that satisfies the additional conditions. A PORF protocol is an innovative characterization of strategy-proof protocols and the first attempt to characterize false-name-proof protocols. Such a characterization is not only theoretically significant but also useful in practice, since it can serve as a guideline for developing new strategy/false-name proof protocols. We present a new false-name-proof protocol based on the concept of a PORF protocol.

1 Introduction

Internet auctions have become an integral part of Electronic Commerce and a promising field for applying AI technologies. Among various studies related to Internet auctions, those on combinatorial auctions have lately attracted considerable attention (an extensive survey is presented in [de Vries and Vohra, 2003]). Although conventional auctions sell a single item at a time, combinatorial auctions sell multiple items with interdependent values simultaneously and allow the bidders to bid on any combination of items. In a combinatorial auction, a bidder can express complementary/substitutable preferences over multiple bids. By taking into account complementary/substitutable preferences, we can increase the participants' utilities and the revenue of the seller.

One important characteristic of an auction protocol is that it is *strategy-proof*. A protocol is strategy-proof if, for each bidder, declaring his/her true evaluation values is a *dominant strategy*, i.e., an optimal strategy regardless of the actions of other bidders. In theory, the revelation principle states that in the design of an auction protocol, we can restrict our attention to strategy-proof protocols without loss of generality [Myerson, 1981]. In other words, if a certain property (e.g., Pareto efficiency) can be achieved using some auction protocol in a dominant-strategy equilibrium, i.e., a combination of dominant strategies of bidders, the property can also be achieved using a strategy-proof auction protocol.

Furthermore, a strategy-proof protocol is also practically useful for applying to Internet auctions. For example, if we use the first-price sealed-bid auction (which is not strategy-proof), the bidding prices must be securely concealed until the auction is closed. On the other hand, if we use a strategy-proof protocol, knowing the bidding prices of other bidders is useless; thus, such security issues become less critical.

Also, the author pointed out the possibility of a new type of fraud called false-name bids, which utilizes the anonymity available in the Internet [Yokoo *et al.*, forthcoming; 2001a; 2000; Sakurai *et al.*, 1999]. False-name bids are bids submitted under fictitious names, e.g., multiple e-mail addresses. Such a dishonest action is very difficult to detect, since identifying each participant on the Internet is virtually impossible.

We say a protocol is *false-name-proof* if, for each bidder, declaring his/her true evaluation values using a single identifier (although the bidder can use multiple identifiers) is a dominant strategy. As for strategy-proof protocols, the revelation principle holds for false-name-proof protocols [Yokoo *et al.*, forthcoming; 2000]. Thus, we can restrict our attention to false-name-proof protocols without loss of generality.

Given that strategy/false-name proof protocols are important both in theory and practice, obvious questions we need to answer are, how can we design such protocols and what features do these protocols have in common, i.e., *characterization* of the protocols. Although there have been several works on characterizing strategy-proof protocols (e.g., [Roberts, 1979; Holmstrom, 1979]), as far as the author is aware, there is no work on characterizing false-name-proof protocols.

In this paper, we introduce an innovative characterization of strategy/false-name proof protocols by introducing a new

distinctive class of combinatorial auction protocols called price-oriented, rationing-free (PORF) protocols. The outline of a PORF protocol is as follows.

- For each bidder, the price of each bundle of goods is determined independently of his/her own declaration, while it can depend on the declarations of other bidders.
- We allocate each bidder a bundle that maximizes his/her utility independently of the allocations of other bidders (i.e., rationing-free).

A PORF protocol looks quite different from traditional protocol descriptions. In a traditional protocol, the allocations of goods are usually determined first, and then the payments of the winners are determined. On the other hand, in a PROF protocol, the prices of bundles for each bidder are determined first, and then the allocation is determined independently based on these prices.

However, surprisingly, a PORF protocol captures all the essential features of a strategy-proof protocol, i.e., if a protocol can be described as a PORF protocol, it is strategy-proof, and vice versa. Also, if a protocol can be described as a PORF protocol that satisfies additional conditions, it is false-name-proof, and vice versa.

As far as the author is aware, a PORF protocol is an innovative characterization of strategy-proof protocols and the first attempt to characterize false-name-proof protocols. Such a characterization is not only theoretically significant but also useful in practice, since it can serve as a guideline for developing new strategy/false-name proof protocols. We present a new false-name-proof protocol based on the idea of a PORF protocol.

2 Problem Settings

Assume there are a set of bidders $N = \{1, 2, \dots, n\}$ and a set of goods $M = \{1, 2, \dots, m\}$. Each bidder i has his/her preferences over $B \subseteq M$. Formally, we model this by supposing that bidder i privately observes a parameter, or signal, θ_i , which determines his/her preferences. We refer to θ_i as the type of bidder i . We assume θ_i is drawn from a set Θ . We assume a quasi-linear, private value model with no allocative externality, defined as follows.

Definition 1 (utility of a bidder)

The utility of bidder i , when i obtains a bundle, i.e., a subset of goods $B \subseteq M$ and pays $p_{B,i}$, is represented as $v(B, \theta_i) - p_{B,i}$.

We assume evaluation value v is normalized by $v(\emptyset, \theta_i) = 0$. Also, we assume for all i , B , $v(B, \theta_i) \leq \infty$ holds. Furthermore, we assume free disposal, i.e., $v(B', \theta_i) \geq v(B, \theta_i)$ for all $B' \supseteq B$.

In a traditional definition [Mas-Colell *et al.*, 1995], an auction protocol is (dominant-strategy) *incentive compatible* (or *strategy-proof*) if declaring the true type/evaluation values is a dominant strategy for each bidder, i.e., an optimal strategy regardless of the actions of other bidders.

In this paper, we extend the traditional definition of incentive compatibility so that it can address false-name bid manipulations, i.e., we define that an auction protocol is (dominant-strategy) incentive compatible if declaring the true type by

using a single identifier is a dominant strategy for each bidder. To distinguish between the traditional and extended definitions of incentive compatibility, we refer to the traditional definition as strategy-proof and to the extended definition as false-name-proof.

An auction protocol is *individually rational* if no participant suffers any loss in a dominant-strategy equilibrium, i.e., the payment never exceeds the evaluation value of the obtained goods. In a private value auction, individual rationality is indispensable; no bidder wants to participate in an auction where he/she might be charged more money than he/she is willing to pay. Therefore, in this paper, we restrict our attention to individually rational protocols. Also, we restrict our attention to deterministic protocols, which always obtain the same outcome for the same input.

We say an auction protocol is *Pareto efficient* when the sum of all participants' utilities (including that of the auctioneer), i.e., the social surplus, is maximized in a dominant-strategy equilibrium. The author has proved that there exists no false-name-proof protocol that satisfies Pareto efficiency and individual rationality at the same time [Yokoo *et al.*, forthcoming; Sakurai *et al.*, 1999]. Therefore, we need to sacrifice efficiency to some extent when false-name bids are possible.

3 Price-oriented, Rationing-Free (PORF) Protocol

A PORF Protocol is defined as follows.

Definition 2 (PORF Protocol)

- Each bidder i declares his/her type $\tilde{\theta}_i$, which is not necessarily the true type θ_i .
- For each bidder i , for each bundle $B \subseteq M$, the price $p_{B,i}$ is defined. This price must be determined independently of i 's declared type $\tilde{\theta}_i$, while it can be dependent on declared types of other bidders.
- We assume $p_{\emptyset,i} = 0$ holds. Also, if $B \subseteq B'$, $p_{B,i} \leq p_{B',i}$ holds.
- For bidder i , a bundle B^* is allocated, where $B^* = \arg \max_{B \subseteq M} v(B, \tilde{\theta}_i) - p_{B,i}$. Bidder i pays $p_{B^*,i}$. If there exist multiple bundles that maximize i 's utility, one of these bundles is allocated.
- The result of the allocation satisfies allocation-feasibility, i.e., for two bidders i, j and bundles allocated to these bidders B_i^* and B_j^* , $B_i^* \cap B_j^* = \emptyset$ holds.

It is straightforward to show that a PORF protocol is strategy-proof. The price of bidder i is determined independently of i 's declared type, and he/she can obtain the bundle that maximizes his/her utility independently of the allocations of other bidders, i.e., the protocol is rationing-free.

On the other hand, in a PORF protocol, the prices must be determined appropriately to satisfy allocation-feasibility¹.

¹Since the price for each bidder can be different and the price of a bundle is not necessarily the sum of the prices of all goods in the bundle, there is no direct relation between the prices that achieve allocation-feasibility and equilibrium prices.

The definition of a PORF protocol requires that if there exist multiple bundles that maximize i 's utility, then one of these bundles must be allocated, but it does not specify exactly which bundle should be allocated. Therefore, if there exist multiple choices, the auctioneer can adjust the allocation of multiple bidders in order to satisfy allocation-feasibility.

Next, we provide some examples of PORF protocols. Since a PORF protocol is strategy-proof, in the rest of this paper, we assume each bidder i declares his/her true type θ_i .

Example 1 Let us consider the auction of a single unit of a single item.

- The price of the only bundle $B = M$ is defined as $p_{B,i} = \max_{j \neq i} v(B, \theta_j)$.

This protocol is identical to the Vickrey auction protocol [Vickrey, 1961].

More specifically, for the bidder with the highest evaluation value, the price of the good is equal to the second highest evaluation value. On the other hand, for other bidders, the price is equal to the highest evaluation value, so nobody except the bidder with the highest evaluation value is willing to buy the good.

Example 2 Let us consider a combinatorial auction. To simplify the protocol description, we introduce the following notation. For a set of goods B and a set of bidders X , where Θ_X is a set of types of bidders in X , we define $V^*(B, \Theta_X)$ as the sum of the evaluation values of X when B is allocated optimally among X .

To be precise, for an feasible allocation $g = (B_1, B_2, \dots)$, where $\bigcup_{j \in X} B_j \subseteq B$ and for all $j \neq j', B_j \cap B_{j'} = \emptyset$, $V^*(X, B)$ is defined as $\max_g \sum_{j \in X} v(B_j, \theta_j)$, where θ_j is the type of bidder j .

The price of bundle B for bidder i is defined as follows:

$$p_{B,i} = V^*(M, \Theta_{N \setminus \{i\}}) - V^*(M \setminus B, \Theta_{N \setminus \{i\}}).$$

This protocol is identical to the Vickrey-Clarke-Groves (VCG) mechanism [Vickrey, 1961; Clarke, 1971; Groves, 1973], i.e., if B is allocated to i in a Pareto efficient allocation, then $p_{B,i}$ is equal to the payment in the VCG; otherwise, $p_{B,i}$ is larger than $v(B, \theta_i)$.

Let us describe how this protocol works. Assume there are two goods 1 and 2, and three bidders, bidder 1, 2, and 3, whose types are θ_1, θ_2 , and θ_3 , respectively. The evaluation value for a bundle $v(B, \theta_i)$ is determined as follows.

	{1}	{2}	{1, 2}
θ_1	6	0	6
θ_2	0	0	8
θ_3	0	5	5

Accordingly, the prices of these bundles for each bidder is given as follows.

	{1}	{2}	{1, 2}
bidder 1	3	8	8
bidder 2	6	5	11
bidder 3	8	2	8

As a result, bidder 1 obtains good 1 with price 3, and bidder 3 obtains good 2 with price 2.

4 Strategy-proof $\rightarrow$ PORF

A PORF protocol looks quite different from traditional protocol descriptions, in which the allocation of the goods/winners are determined, and then the payments of these winners are determined. In a PROF protocol, the prices of bundles for each bidder is determined first, and then the allocation is determined based on these prices. In a traditional protocol description, the payment of bidder i must be determined independently of i 's type to make the protocol strategy-proof. This is similar to the fact that the price of bidder i in a PORF protocol must be determined independently of i 's declared type. The most distinctive characteristic of a PORF protocol is that it is rationing-free, i.e., each bidder can obtain the optimal bundle based on the prices, and the allocation for each bidder is done independently (except when a bidder is totally indifferent between multiple bundles).

Surprisingly, a PORF protocol is not only a sufficient condition that a protocol is strategy-proof, but it is also a necessary condition, i.e., the following theorem holds.

Theorem 1 If a protocol is strategy-proof, then the protocol can be described as a PORF protocol.

The argument presented in this section is very general and requires only the fact that a protocol is deterministic and individually rational.

The outline of the proof in the remainder of this section can be summarized as follows. First, we show that if a bidder is *single-minded* (Definition 3), i.e., he/she is interested only in a particular bundle, any strategy-proof protocol can be described as a PORF protocol (Lemma 3), i.e., the price of the bundle is determined and the bidder will obtain the bundle if his/her evaluation value is larger than the price. An intuitive explanation for this result is that if such a price does not exist, a single-minded bidder can have an incentive to under/over-declare his/her evaluation value for the bundle. The only way to make the protocol strategy-proof is to set a fixed threshold based on other bidders' evaluation values, which determines whether the single-minded bidder will obtain the bundle or not.

Next, we show that this result can be extended to the case where a bidder is *k-minded* (Definition 5), i.e., a bidder is interested in multiple bundles at the same time (Lemma 5). An intuitive explanation for this result is as follows. A single-minded bidder can pretend to be a *k-minded* bidder, and vice versa. Therefore, to a *k-minded* bidder, the protocol must give the results that are basically equivalent to a single-minded bidder.

Any bidder can be represented as a $(2^m - 1)$ -minded bidder, where m is the number of goods and $2^m - 1$ is the number of all possible bundles (except an empty set). Thus, we can show that any strategy-proof protocol can be represented as a PROF protocol.

In the following, we show the detailed proof of Theorem 1. To derive the theorem, we introduce notions such as *single-minded bidder*, *monotone allocation rule*, and *critical-value*, which are used in [Lehmann et al., 2002; Mu'alem and Nisan,

2002]. The proofs of Lemma 1 and 2 are basically due to [Lehmann et al., 2002; Mu'alem and Nisan, 2002].

Definition 3 (single-minded bidder)

We say bidder i is single-minded if i requires only one bundle B_i , i.e., for any bundle B , if $B_i \subseteq B$, then $v(B, \theta_i) = v_i$, otherwise, $v(B, \theta_i) = 0$.

If bidder i is single-minded, i.e., the declared type of i can be considered to be single-minded, then a PORF protocol can be described as follows, assuming the set of other bidders and their types are fixed. We denote this protocol as a PORF protocol for a single-minded bidder.

- For bidder i , p_i , which is the price of B_i , is defined. If $v(B_i, \theta_i) > p_i$, then B_i (or a superset of B_i) is allocated to bidder i and i pays p_i . If $v(B_i, \theta_i) < p_i$, then no good is allocated. If $v(B_i, \theta_i) = p_i$, then either B_i (or a superset of B_i) is allocated and i pays p_i , or no good is allocated.

Definition 4 (monotone allocation rule)

We say a protocol is monotone for a single-minded bidder i if the following condition is satisfied, assuming the set of other bidders and their types are fixed.

If bundle B_i (or a superset of B_i) is allocated to bidder i when i 's evaluation value for B_i is v_i , then B_i (or a superset of B_i) is also allocated when i 's evaluation value for B_i is $v'_i > v_i$.

The following lemma holds.

Lemma 1 If a protocol is strategy-proof, then the protocol is monotone for a single-minded bidder.

Proof: If a protocol is not monotone, there exists a case where B_i (or a superset) is allocated to bidder i when i 's evaluation value for B_i is v_i , while B_i (or a superset) is not allocated to bidder i when i 's evaluation value for B_i is $v'_i > v_i$. Since the protocol is individually rational, the payment when i 's evaluation value is v_i must be less than or equal to v_i . Therefore, when i 's true evaluation value is v'_i , if i truthfully declares his/her type, i 's utility is 0, since neither B_i nor a superset is allocated. However, if i declares a false type as a single-minded bidder where the evaluation value for B_i is v_i , i can obtain a positive utility. This contradicts the assumption that the protocol is strategy-proof. $\square$

Furthermore, the following lemma holds.

Lemma 2 If a protocol is monotone for a single-minded bidder, then there exists critical-value c that satisfies the following condition, assuming the set of other bidders and their types are fixed: if $v_i > c$, B_i (or a superset) is allocated to i , while if $v_i < c$, no good is allocated.

Proof: Let us assume that no critical value exists. Then, for an arbitrary value c' , either one of the following two cases holds.

- case i: there exists $v' > c'$, where bidder i cannot obtain B_i when i 's evaluation value is v' .
- case ii: there exists $v'' < c'$, where bidder i can obtain B_i when i 's evaluation value is v'' .

If we set $c' = 0$, case ii cannot be true so case i must hold. Let us re-assign c' as v' and repeat this procedure until case ii occurs (if case ii never occurs, then ∞ becomes a critical value). In this case, bidder i cannot obtain B_i when the evaluation value is c' , while i can obtain B_i when the evaluation value is $v'' < c'$. This contradicts the assumption that the protocol is monotone. $\square$

Next, we show that the following lemma holds.

Lemma 3 If a protocol is strategy-proof, then, for a single-minded bidder, the protocol can be described as a PORF protocol for a single-minded bidder, i.e., if i 's evaluation value for B_i is larger than a given value p_i , then B_i (or a superset) is allocated and i 's payment is p_i . If i 's evaluation value for B_i is smaller than p_i , no good is allocated.

Proof: From Lemma 1 and 2, the protocol is monotone and there exists critical-value c , i.e., B_i (or a superset) is allocated when the evaluation value is larger than c , while no good is allocated when the evaluation value is smaller than c . The only thing we need to show is that the payment is equal to c when B_i (or a superset) is allocated. Let us assume that the payment is $c' \neq c$ and derive a contradiction.

First, let us consider the case $c' < c$, i.e., when the evaluation value of i is $v_i > c$, i obtains B_i (or a superset) but the payment is $c' < c$. When the evaluation value of i is $c' + \epsilon$, if i declares the true type, no good is allocated and the obtained utility is 0, while if i declares a false type where the evaluation value is v_i , i can obtain B_i (or a superset) and the payment is c' , thus the obtained utility becomes positive. This contradicts the assumption that the protocol is strategy-proof.

Next, let us consider the case $c' > c$, i.e., when the evaluation value of i is $v_i > c$, i obtains B_i (or a superset) but the payment is $c' > c$. If i declares a false type where the evaluation value is $c + \epsilon$, i can obtain B_i (or a superset) because c is a critical value. Since the protocol is individually rational, the payment must be less than or equal to $c + \epsilon$, which is less than c' , i.e., the payment when i declares the true type. This contradicts the assumption that the protocol is strategy-proof.

From the above, the protocol can be described as a PORF protocol for a single-minded bidder where $p_i = c$. $\square$

Definition 5 (k-minded bidder)

We say bidder i is a k -minded bidder if i requires exactly one bundle from k bundles $B_{i_1}, \dots, B_{i_j}, \dots, B_{i_k}$. Let us represent i 's evaluation value for B_{i_j} as v_{i_j} . For notation simplicity, let us assume $B_{i_0} = \emptyset$ and $v_{i_0} = 0$. The evaluation value of i for bundle B is defined as follows.

$$v(B, \theta_i) = \max_{0 \leq j \leq k} v_{i_j}, \text{ where } B_{i_j} \subseteq B.$$

From Lemma 3, if bidder i is a single-minded bidder who requires only B_{i_j} , then a strategy-proof protocol must be a PORF protocol for a single-minded bidder. Let us represent the price for bidder i of B_{i_j} in this protocol as p_{i_j} .

First, we show that the following lemma holds.

Lemma 4 If a protocol is strategy-proof, then, for k -minded bidder i , the payment when i obtains B is given by $p_{B,i} = \max_{0 \leq j \leq k} p_{i_j}$, where $B_{i_j} \subseteq B$.

Proof: Let us assume that $j' = \arg \max_{0 \leq j \leq k} p_{i_j}$ where $B_{i_j} \subseteq B$. We derive a contradiction assuming $p_{B,i} \neq p_{i_{j'}}$. First, let us consider the case $p_{B,i} < p_{i_{j'}}$. Assume that i is a

single-minded bidder who requires B_{i_j} only, and i 's evaluation value for B_{i_j} is p_{i_j} . In this case, if i declares the true type, the obtained utility is 0. On the other hand, if i declares his/her type as a k -minded bidder, i can obtain $B \supseteq B_{i_j}$, and the payment is $p_{B,i} < p_{i_j}$, thus i can obtain a positive utility. This contradicts the assumption that the protocol is strategy-proof.

Next, let us consider the case $p_{B,i} > p_{i_j}$. If i declares its type as a single-minded bidder who requires only B_{i_j} , i can obtain B_{i_j} (or a superset) and the payment is p_{i_j} , which is less than $p_{B,i}$, i.e., the payment when he/she declares the true type. This contradicts the assumption that the protocol is strategy-proof. $\square$

Finally, we show that the following lemma holds.

Lemma 5 *If a protocol is strategy-proof, then, for a k -minded bidder i , the protocol can be described as a PORF protocol for a k -minded bidder, i.e., for each B_{i_j} , p_{i_j} , which is the price for bidder i , is defined, and the protocol assigns $B_{i_{j^*}}$ (or a superset), where $j^* = \arg \max_{0 \leq j \leq k} (v_{i_j} - p_{i_j})$. The payment is $p_{i_{j^*}}$.*

Proof: We derive a contradiction assuming that a strategy-proof protocol assigns $B \not\supseteq B_{i_{j^*}}$ for k -minded bidder i . Let us choose $j'' = \arg \max_{0 \leq j \leq k} v_{i_j}$, where $B_{i_j} \subseteq B$. From the definition of a k -minded bidder, $v(B, \theta_i) = v_{i_{j''}}$ holds. Furthermore, from the fact that $B \not\supseteq B_{i_{j^*}}$, $v_{i_{j^*}} - p_{i_{j^*}} > v_{i_{j''}} - p_{i_{j''}}$ holds.

From Lemma 4, the payment when i obtains B is given by $p_{B,i} = \max_{0 \leq j \leq k} p_{i_j}$, where $B_{i_j} \subseteq B$. Obviously, $p_{B,i} \geq p_{i_{j''}}$ holds. Thus, $v_{i_{j^*}} - p_{i_{j^*}} > v_{i_{j''}} - p_{i_{j''}} \geq v(B, \theta_i) - p_{B,i}$ holds. This formula represents the fact that the utility when i declares the true type (i.e., $v(B, \theta_i) - p_{B,i}$) is less than the utility when i declares a false type, where i is a single-minded bidder that requires only $B_{i_{j^*}}$ (i.e., $v_{i_{j^*}} - p_{i_{j^*}}$). This contradicts the assumption that the protocol is strategy-proof. $\square$

Any bidder can be represented as a $(2^m - 1)$ -minded bidder, where m is the number of goods and $2^m - 1$ is the number of all possible bundles (except an empty set). Since Lemma 5 holds for all k , from Lemma 5, we can derive Theorem 1.

5 PORF with additional conditions $\leftrightarrow$

False-name-proof

From the definition, if a protocol is false-name-proof, it is also strategy-proof. Therefore, it is obvious that false-name-proof $\rightarrow$ PORF holds. On the other hand, PORF $\rightarrow$ false-name-proof does not hold in general. For example, the VCG mechanism is strategy-proof, so it can be described as a PORF protocol, but it is not false-name-proof, as shown in [Yokoo *et al.*, forthcoming; Sakurai *et al.*, 1999].

In this section, we limit our attention to protocols that satisfy the following condition.

Definition 6 (weakly-anonymous pricing rule (WAP))

For bidder i , the price of bundle B is given as a function of types of other bidders, i.e., the price can be described as $p(B, \Theta_X)$, where X is the set of bidders except i , and Θ_X is the set of types of bidders in X .

The above condition requires that if two bidders are facing the same types of opponents, their prices must be identical for all bundles. The WAP condition is intuitively natural and virtually all well-known protocols, including the VCG, satisfy this condition.

For a PORF protocol that satisfies the WAP condition, we define the following additional condition.

Definition 7 (no super-additive price increase (NSA))

For all subset of bidders $S \subseteq N$ and $X = N \setminus S$, and for $i \in S$, let us denote B_i as a bundle that maximizes i 's utility, then $\sum_{i \in S} p(B_i, \bigcup_{j \in S \setminus \{i\}} \{\theta_j\} \cup \Theta_X) \geq p(\bigcup_{i \in S} B_i, \Theta_X)$.

An intuitive meaning of this condition is that the price of buying a combination of bundles (the right side of the inequality) must be smaller than or equal to the sum of the prices for buying these bundles separately (the left side).

The next theorem states that for a PORF protocol with the WAP, the NSA is a sufficient condition for a protocol to be false-name-proof.

Theorem 2 *If a PORF protocol with the WAP satisfies the NSA condition, then the protocol is false-name-proof.*

Proof: The proof is rather clear. If a bidder uses a set of identifiers S , then from the NSA condition, the bidder can obtain the same set of goods by using a single identifier, while the payment becomes smaller (or remains the same). $\square$

We can show that for a PORF protocol with WAP, the NSA is not only a sufficient condition but also a necessary condition, i.e., the following theorem holds.

Theorem 3 *If a protocol is a PORF protocol with the WAP and is false-name-proof, then it satisfies the NSA condition.*

Proof: Let us assume that there exists a false-name-proof protocol that can be described as a PORF protocol with the WAP but does not satisfy the NSA condition. More precisely, for a set of identifiers S , $X = N \setminus S$, and for $i \in S$, B_i is the bundle that maximizes i 's utility, but $\sum_{i \in S} p(B_i, \bigcup_{j \in S \setminus \{i\}} \{\theta_j\} \cup \Theta_X) < p(\bigcup_{i \in S} B_i, \Theta_X)$.

Let us assume the case where bidder i' is facing opponents whose types are Θ_X . Also, let us assume bidder i' is single-minded for bundle $\bigcup_{i \in S} B_i$, and the evaluation value is $p(\bigcup_{i \in S} B_i, \Theta_X)$. If bidder i' declares his/her true type, the obtained utility is 0 (since if i' can obtain the bundle, the payment is equal to his/her evaluation value). On the other hand, if i' uses a set of identifiers S , and for each $i \in S$, he/she declares the type as θ_i , then for each identifier i , B_i is obtained.

The sum of the payment is $\sum_{i \in S} p(B_i, \bigcup_{j \in S \setminus \{i\}} \{\theta_j\} \cup \Theta_X)$, which is less than the evaluation value of i' , i.e., $p(\bigcup_{i \in S} B_i, \Theta_X)$. Thus, bidder i' can obtain positive utility by utilizing false-name bids. This contradicts the assumption that the protocol is false-name-proof. $\square$

For the protocols that are strategy-proof (SP), false-name-proof (FP), PORF, WAP, and NSA, the subset/superset relations can be illustrated as Figure 1. The VCG mechanism can be described as a PORF protocol with WAP, but it does not satisfy the NSA condition. Therefore, it is not false-name-proof. One example of a false-name-proof protocol that does not use the WAP rule is a dictatorial protocol where all goods

are allocated to one special bidder (the dictator) regardless of other bidders' evaluation values.

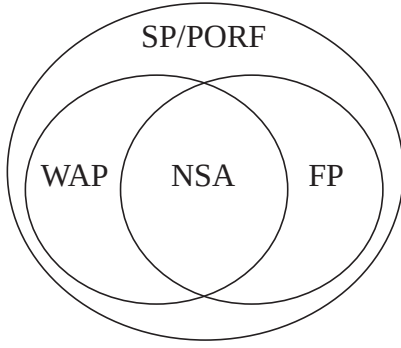


Figure 1: Relations of Protocols

Due to space limitations, we omit detailed descriptions, but all existing false-name-proof protocols developed so far, e.g., the LDS [Yokoo *et al.*, 2001a], the IR [Yokoo *et al.*, 2001b], and the GAL protocol [Terada and Yokoo, 2003], can be described as a PORF protocol that satisfies the NSA condition.

6 New False-name-proof Protocol

In this section, we develop a new false-name-proof protocol based on the concept of the PORF protocol. To simplify the protocol description, we introduce a concept called a *minimal bundle*.

Definition 8 (minimal bundle) Bundle B is called *minimal* for bidder i if for all $B' \subset B$ and $B' \neq B$, $v(B', \theta_i) < v(B, \theta_i)$ holds.

In this new protocol, the price of bundle B for bidder i is defined as follows:

- $p_{B,i} = \max_{B_j \subseteq M, j \neq i} v(B_j, \theta_j)$, where $B \cap B_j \neq \emptyset$ and B_j is minimal for bidder j .

In short, the price of bundle B is equal to the highest evaluation value of a bundle, which is minimal and conflicting with bundle B .

Compared with the LDS protocol [Yokoo *et al.*, 2001a], this protocol is much simpler and does not require any parameters to be set by the auctioneer, while in the LDS protocol, the auctioneer must carefully determine the reservation price and the way of dividing goods into multiple bundles. If all bidders are single-minded, this protocol is one example of greedy protocols described in [Lehmann *et al.*, 2002].

Let us describe how this protocol works. Let us assume there are three goods 1, 2, and 3, and two bidders, bidder 1 and bidder 2, whose types are θ_1, θ_2 , respectively. The evaluation value for a bundle $v(B, \theta_i)$ is determined as follows.

	{1}	{2}	{3}	{1, 2}	{2, 3}	{1, 3}	{1, 2, 3}
θ_1	0	0	60	210	60	60	210
θ_2	0	110	110	110	110	110	110

These evaluation values mean that bidder 1 is 2-minded for bundles {1, 2} and {3}, while bidder 2 is 2-minded for bundles {2} and {3}. These bundles are minimal bundles. The prices of these bundles are given as follows.

	{1}	{2}	{3}	{1, 2}	{2, 3}	{1, 3}	{1, 2, 3}
bidder 1	0	110	110	110	110	110	110
bidder 2	210	210	60	210	210	210	210

As a result, bundle {1, 2} is allocated to bidder 1 and bundle {3} is allocated to bidder 2.

It is clear that this protocol satisfies the allocation-feasibility. For each good l , let us choose bidder j^* and bundle B_j^* that maximize $v(B_j, \theta_j)$, where $l \in B_j$ and B_j is minimal for bidder j . Then, only bidder j^* is willing to obtain a bundle that contains good l . For all other bidders, the price of a bundle that contains l is higher than (or equal to) his/her evaluation value.

Furthermore, it is clear that this protocol satisfies the NSA condition. In this pricing scheme, $p(B \cup B', \Theta_X) = \max(p(B, \Theta_X), p(B', \Theta_X))$ holds for all B, B' , and Θ_X . Therefore, the following formula holds.

$$p\left(\bigcup_{i \in S} B_i, \Theta_X\right) = \max_{i \in S} p(B_i, \Theta_X) \leq \sum_{i \in S} p(B_i, \Theta_X)$$

Furthermore, in this pricing scheme, prices increase monotonically by adding opponents, i.e., for all $X' \supseteq X$, $p(B, \Theta_{X'}) \geq p(B, \Theta_X)$ holds. Therefore, for each i , $p(B_i, \bigcup_{j \in S \setminus \{i\}} \{\theta_j\} \cup \Theta_X) \geq p(B_i, \Theta_X)$ holds. Therefore, the NSA condition, i.e., $\sum_{i \in S} p(B_i, \bigcup_{j \in S \setminus \{i\}} \{\theta_j\} \cup \Theta_X) \geq p(\bigcup_{i \in S} B_i, \Theta_X)$ holds.

7 Discussions

As far as the author is aware, a PORF protocol is an innovative characterization of strategy-proof protocols and the first attempt to characterize false-name-proof protocols. Here, we discuss several previous works on characterizing strategy-proof protocols.

In [Lehmann *et al.*, 2002; Mu'alem and Nisan, 2002], it is shown that if there exist only single-minded bidders, a strategy-proof protocol is monotonic and has a critical value. Since their motivation is to develop computationally efficient strategy-proof protocols that can achieve semi-optimal allocations, they do not extend their results to more general cases such as k -minded bidders.

In [Roberts, 1979], a characterization of strategy-proof mechanisms is shown for general social choice problems. It is shown that any strategy-proof protocol can be described as a variation of the Groves mechanisms [Groves, 1973]. On the other hand, in the model used in this paper, we assume that the evaluation values of each bidder satisfy no allocative-externality condition. Therefore, the results described in [Roberts, 1979] cannot be applied, i.e., a PORF protocol is not necessarily to be a variation of the Groves mechanisms.

In [Holmstrom, 1979], a characterization of strategy-proof mechanisms is described. It is shown that with the assumption that the preferences of each participant satisfy a condition called *smoothly-connected*, any strategy-proof protocol that satisfies Pareto efficiency must be an instance of the Groves mechanisms. This result can be applied to the model used in this paper since the smoothly-connected condition still holds. Therefore, if we require that a protocol be Pareto efficient, it

is likely that the pricing scheme described in Example 2 is the only way to make a PORF protocol Pareto efficient².

As well as a PORF protocol is theoretically significant, since it is an equivalent class of strategy-proof protocols, it has practical importance since it can serve as a guideline for developing new strategy/false-name proof protocols. Designing a strategy/false-name proof protocol has been a difficult task. As shown in Section 6, we successfully developed a new false-name-proof protocol based on the idea of a PORF protocol. The simplicity of this newly developed protocol compared with the LDS protocol illustrates the expressive power of a PORF protocol. Of course, we need to prove that a PORF protocol satisfies allocation-feasibility. However, this tends to be much easier than directly proving a protocol is strategy/false-name proof, since we can assume each bidder declares his/her true type by using a single identifier.

As for the computational cost of executing a protocol, a naive implementation of a PORF protocol requires calculating prices for all bundles of all bidders. However, as in the case of the VCG, we can describe a protocol either as a PORF protocol or in a traditional manner in which an allocation of goods is determined, and then the payments are calculated based on the allocation. We can assume that the description of a PORF protocol is not for actual implementation but for serving as a normative guideline in proving characteristics of a protocol.

8 Conclusions

In this paper, we introduced a new distinctive class of combinatorial auction protocols called PORF protocols. Although a PORF protocol looks quite different from traditional protocol descriptions, surprisingly, it is a sufficient and necessary condition for a protocol to be strategy-proof. Furthermore, we showed that a PORF protocol satisfying additional conditions is false-name-proof; at the same time, any false-name-proof protocol can be described as a PORF protocol that satisfies the additional conditions.

A PORF protocol is not only theoretically significant but also useful in practice, since it can serve as a guideline for developing new strategy/false-name proof protocols. We successfully developed a new false-name-proof protocol based on the idea of a PORF protocol. We are currently extending the obtained results to combinatorial exchange.

Acknowledgements

The author wishes to thank Yoshikatsu Tatamitani, Kenji Terada, Yuko Sakurai, Atsushi Iwasaki, and Shigeo Matsubara for their helpful discussions.

References

- [Clarke, 1971] E. H. Clarke. Multipart pricing of public goods. *Public Choice*, 2:19–33, 1971.
- [de Vries and Vohra, 2003] S. de Vries and R. V. Vohra. Combinatorial auctions: A survey. *INFORMS Journal on Computing*, 15, 2003.
- [Groves, 1973] T. Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.
- [Holmstrom, 1979] B. Holmstrom. Groves' scheme on restricted domains. *Econometrica*, 47(5):1137–1144, 1979.
- [Lehmann et al., 2002] D. Lehmann, L. I. O'Callaghan, and Y. Shoham. Truth revelation in approximately efficient combinatorial auctions. *Journal of the ACM*, 49(5):577–602, 2002.
- [Mas-Colell et al., 1995] A. Mas-Colell, M. D. Whinston, and J. R. Green. *Microeconomic Theory*. Oxford University Press, 1995.
- [Mu'alem and Nisan, 2002] A. Mu'alem and N. Nisan. Truthful approximation mechanisms for restricted combinatorial auctions. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-2002)*, pages 379–384, 2002.
- [Myerson, 1981] R. B. Myerson. Optimal auction design. *Mathematics of Operation Research*, 6:58–73, 1981.
- [Roberts, 1979] K. Roberts. The characterization of implementable choice rules. In J.-J. Laffont, editor, *Aggregation and Revelation of Preferences*, pages 321–350. North-Holland Publishing Company, 1979.
- [Sakurai et al., 1999] Y. Sakurai, M. Yokoo, and S. Matsubara. A limitation of the Generalized Vickrey Auction in Electronic Commerce: Robustness against false-name bids. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, pages 86–92, 1999.
- [Terada and Yokoo, 2003] K. Terada and M. Yokoo. False-name-proof multi-unit auction protocol utilizing greedy allocation based on approximate evaluation values. In *Proceedings of the Second International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2003)*, 2003.
- [Vickrey, 1961] W. Vickrey. Counter speculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.
- [Yokoo et al., 2000] M. Yokoo, Y. Sakurai, and S. Matsubara. The effect of false-name declarations in mechanism design: Towards collective decision making on the Internet. In *Proceedings of the Twentieth International Conference on Distributed Computing Systems (ICDCS-2000)*, pages 146–153, 2000.
- [Yokoo et al., 2001a] M. Yokoo, Y. Sakurai, and S. Matsubara. Robust combinatorial auction protocol against false-name bids. *Artificial Intelligence*, 130(2):167–181, 2001.
- [Yokoo et al., 2001b] M. Yokoo, Y. Sakurai, and S. Matsubara. Robust multi-unit auction protocol against false-name bids. In *Proceedings of 17th International Joint Conference on Artificial Intelligence (IJCAI-2001)*, pages 1089–1094, 2001.
- [Yokoo et al., forthcoming] M. Yokoo, Y. Sakurai, and S. Matsubara. The effect of false-name bids in combinatorial auctions: New fraud in internet auctions. *Games and Economic Behavior*, forthcoming.

²Since we require that the protocol also be individually rational, the Groves mechanisms would be reduced to the VCG.

MULTIAGENT SYSTEMS

MULTIAGENT TRACKING

On Identifying and Managing Relationships in Multi-Agent Systems

Ronald Ashri and Michael Luck

University of Southampton, Southampton, SO17 1BJ, United Kingdom
{ra00r,mml}@ecs.soton.ac.uk

Mark d’Inverno

University of Westminster, London, E1 4NS, United Kingdom
dinverm@westminster.ac.uk

Abstract

Multi-agent systems result from interactions between individual agents. Through these interactions different kinds of relationships are formed, which can impact substantially on the overall system performance. However, the behaviour of agents cannot always be anticipated, especially when dealing with open and complex systems. Open agent systems must incorporate *relationship management* mechanisms to constrain agent actions and allow only desirable interactions. In consequence, in this paper we tackle two important issues. Firstly, in addressing *management*, we identify the range of different control mechanisms that are required and when they should be applied. Secondly, in addressing *relationships*, we present a model for identifying and characterising relationships in a manner that is application-neutral and amenable to automation.

1 Introduction

Agent-based computing is often presented as a viable paradigm for tackling problems where the main challenge is the need to enable dynamic interactions between heterogeneous components, in which each component may have its own thread of control [Jennings, 2000]. Agents are understood as independent entities, capable of individual action and working towards their design goals and, as such, represent a natural, intuitive approach to these problems. Furthermore, agents are also *social* entities that interact by performing actions that affect their environment and other agents in the environment. Such actions can enable coordination (in which agents arrange their individual activities in a coherent manner), collaboration (in which agents work together to achieve a common objective) or competition (in which agents contend for access to common resources), and so on. Through these interactions, different kinds of relationships are formed between agents, and they can impact substantially on the overall system functioning.

The challenge for the system designer is to ensure that only the *right* kinds of relationships develop so that the overall system functions within acceptable parameters. It is not enough for a designer to assume that the only interactions

between agents that will take place are those explicitly specified at design-time, especially when dealing with *autonomous* entities operating in open environments where agents may join or leave the system at any time and no assumptions are made about agent behaviour. As a result, open agent systems must incorporate mechanisms, beyond the design of individual agents, to constrain agent actions and allow only desirable interactions. We refer to these as *relationship management* mechanisms.

To address the problem of relationship management, we must be aware of the different types of management that may be required in a multi-agent system and also be able to identify and characterise relationships in order to understand what type of management is best suited for the situation at hand. It is exactly these concerns that we attempt to tackle in this paper. This paper advances the current understanding of how to deal with the issue of relationship management in two important ways. Firstly, in addressing *management*, we identify the range of different control mechanisms that are required and when they should be applied. Secondly, in addressing *relationships*, we present a model for identifying and characterising relationships in a manner that is application-neutral and amenable to automation.

We begin by presenting a motivating scenario that outlines some realistic situations that require relationship management mechanisms. Subsequently, we identify situations in which mechanisms for managing relationships can be applied and we define a space of possible management mechanisms types. We then focus on the issue of relationship identification through an analysis of individual agent actions and perceptions. Finally, we conclude with a brief look at related work and outline the way in which the contributions of this paper add to such work.

2 Motivating Scenario

Let us consider a typical computer science research lab, where communication and cooperation between researchers is facilitated by an application developed as a multi-agent system. Each researcher is represented by an agent that makes public to the lab’s network, through the researcher’s personal computer, their personal profile (interests, publications, availability) as well as research material (downloaded papers, presentations, software, links to online material) that they have stored locally. The application also supports researchers

while working away from their desks through wireless connections on laptops or more limited personal digital assistants. Finally, visitors to the lab are provided with similar functionality, through mobile devices, so that they can more easily locate people within the lab.

For such an application to be successful we must, at the very least, ensure that agents make available the right kinds of information about the researcher they represent, so as to add to the overall system functioning, and that agents do not abuse the system, causing degradation of the abilities of others, e.g. by making too many queries and using up bandwidth. Such overarching system goals can be achieved by the appropriate management of relationships between agents. Below we present some situations that call for relationship management.

1. When retrieving files from a researcher's local hard disk limits on how much can be retrieved, and how many agents can simultaneously retrieve at any given moment should be set. This would ensure that no single machine suffers from too much demand.
2. A researcher may want to tailor access to locally stored research material based on who is accessing it. For example, those working on common projects might access work in progress, while PhD students might not have access to all material of their supervisor.
3. Each agent should follow certain conventions. A researcher should not restrict access to their own material while expecting to have full access to others.
4. Groups of researchers in the lab could benefit from closer relationships due to a common interest in a specific subject or project. This tighter relationship could be reflected by the creation of *special interest groups* where the sharing of materials between them takes place at a more frequent rate and without explicit requests to receive them.
5. Different devices offer different capabilities. A document may not be retrieved by a visitor's PDA if it is not capable of displaying it, but the PDA may request a commitment to send the document to the visitor via e-mail.

The examples illustrate the different types of problems and the different needs relating to the management of relationships between agents. Even though some may seem to be amenable to traditional file management techniques (e.g. the first two) the *range* of situations described is more complex and the envisaged application of regulation is more dynamic and fine-grained, and is handled by the agent system and not a human administrator.

The examples can be roughly divided into those that may demand the establishment of compulsory, rigid control of relationships and those where control could be more flexible or even optional. The first two examples could fall in the former category and the last three in the latter. Furthermore, some define the need for control based on the current *situation* (Example 1), while others view it based on *who* is interacting with whom (Example 2) or past history of interactions (Examples 3 and 4). Agents can benefit from the establishment

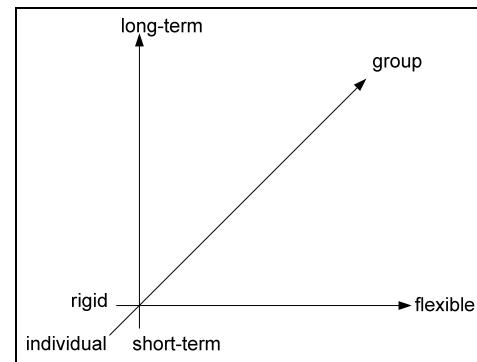


Figure 1: Space of management types

of long term commitments between a group of agents, such as forming interest groups (Example 4) or short term commitments for the fulfilment of a specific goal, such as dealing with access through limited capability devices (Example 5). Finally, the needs of the application may call for agents to adopt a specific stance due to their individual characteristics, such as in Example 2 and Example 5 where we have supervisors and supervisees, and local and visiting researchers respectively.

3 Relationship Management Types

These different cases provide a guide for a more comprehensive categorisation of situations that may *activate* the need for management and the *kind* of management that can be applied.

We identify three different criteria that may activate the need for management. Management of relationships can be required when a specific event occurs (e.g. an agent accessing a researcher's profile), when the environment corresponds to a predefined situation (e.g. the number of agents simultaneously accessing a profile exceeds four) or when a specific agent is performing an action (e.g. a student accessing a supervisor's profile). Thus we can divide the cases into *event-based*, *situation-based* and *agent-attribute-based*. These cases are not mutually exclusive but can occur at the same time. For example, management may be needed for an agent with specific attributes in a particular situation when a specific event occurs.

The kind of management can be defined along three different dimensions: firstly, the *number of agents* it refers to; secondly, the *rigidity* of the applied regulation, which indicates to which extent agents can choose not to follow the regulation; and finally, the *duration* of management, which indicates for how long agents should follow the applied regulation. Using these dimensions we can define a *management space*, illustrated in Figure 1. By using the notion of a *space*, rather than rigid categories, we want to emphasise that the understanding of relationship management needs to better reflect the *range* of situations that may occur. So, for example, management may not need to be simply rigid or optional but also the whole range of flexibility in between.

With knowledge of when management may be applied and what form it can take we can more clearly characterise and compare different management methods. For example, re-

turning to our scenario, we could control the level of access to material (Example 1) via a regulation that agents could choose to obey, which is activated when the agent performs the action of accessing the information (i.e. it is event based) and applies only to that agent accessing information (i.e. it refers to only one agent). Alternatively, the same problem could be tackled through a long-term regulation that applies to all agents and cannot be questioned. Each method has its own pros and cons and can be characterised through the concepts we provide.

4 Identifying Relationships

However, before one can manage relationships one must be able to identify and characterise them. We begin by briefly outlining some foundational concepts that we will use to formalise the model later on. To do so, we adopt the Z specification language [Spivey, 1992]. Based on the SMART framework [d’Inverno and Luck, 2001], we start by defining two primitives, *attribute* and *action*. Formally, these primitives are specified as given sets which means that we say nothing about how they might be represented for any particular system. In addition, two secondary concepts, *goal* and *environment*, are specified in terms of attributes. Attributes are simply features of the world, and are the only characteristics that are manifest. Actions can change environments by adding or removing attributes.

[*Attribute*, *Action*]

Goals, in this context, are desirable environmental states described by non-empty sets of attributes. An environment is a set of attributes that describes all the features within that environment.

$Environment == \mathbb{P}_1 \text{ Attribute}$

$Goal == \mathbb{P}_1 \text{ Attribute}$

Using these primitive concepts we can describe an agent as shown in the following schema. An agent is described by a set of attributes, can perform certain actions and has a number of goals to achieve.

<i>Agent</i> $attributes : \mathbb{P} \text{ Attribute}$ $actions : \mathbb{P} \text{ Action}$ $goals : \mathbb{P} \text{ Goal}$

The model for relationship identification builds on just these concepts and can deal with a wide variety of situations.

4.1 Influence Types

As discussed above, relationships can take a number of forms ranging from cooperation towards a common goal to competition for possession of, or access to, a common resource. When such relationships occur they may affect the actions agents perform or were intending to perform. In such situations, the ability of one agent to achieve its goals becomes *dependent* on another agent’s actions. It is these dependencies between agents that we are categorising through an analysis

of the agent’s respective actions and perceptions of the environment.

We begin by defining the state of an agent as those aspects of the environment it can perceive, along with its actual situation in the environment.

<i>AgentState</i> <i>Agent</i> $possiblepercepts : \mathbb{P} \text{ Attribute}$ $situation : \mathbb{P} \text{ Attribute}$

The state of the entire multi-agent system would then be given by the environment and the states of each individual agent in that environment.

<i>MAState</i> $environment : \mathbb{P} \text{ Attribute}$ $agents : \mathbb{P} \text{ AgentState}$ $\forall a : agents \bullet a.attributes \subseteq environment \wedge$ $a.situation \subseteq environment$ $\forall a : agents \bullet a.possiblepercepts \subseteq$ $environment$
--

Now, if we consider that an agent usually takes into account some measure of the current state of the environment and performs actions based on those measures which, in turn, may change the environment, we have an indication of which aspects of an agent we should investigate to understand under which conditions two agents may be related. We must examine the overlaps between the aspects of the environment that one agent can view, through its *sensing* capabilities, and those aspects it can affect, through its *actuator* capabilities, in relation to other agents. We call the former the *viewable environment* and the latter the *region of influence*. The *Viewable Environment* schema formalises the former notion.

<i>ViewableEnvironment</i> <i>MAState</i> $viewable : \text{AgentState} \rightarrow \mathbb{P} \text{ Attribute}$ $dom \text{ viewable} = agents$ $\forall a : agents \bullet viewable \ a \subseteq environment$

An analogous schema describes the region of influence.

<i>RegionOfInfluence</i> <i>MAState</i> $regioninfluence : \text{AgentState} \rightarrow \mathbb{P} \text{ Attribute}$ $dom \text{ regioninfluence} = agents$ $\forall a : agents \bullet regioninfluence \ a \subseteq environment$
--

In Figure 2, we illustrate these concepts. The regions of the environment that agents view and take into account are represented as ellipses while the regions they are able to affect, i.e. the regions of influence, are represented as pentagons. Given this information we can infer that Agent A and Agent B could be related, with A able to directly affect the viewable

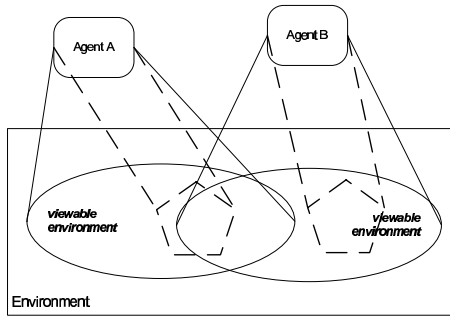


Figure 2: Region of influence affects viewable environment

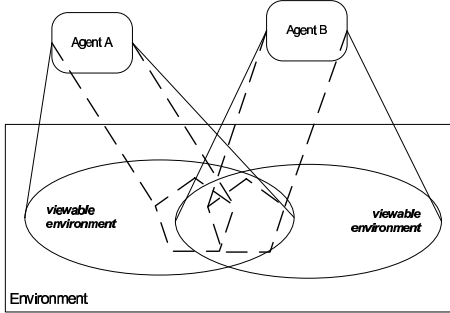


Figure 3: Regions of influence overlap

environment of B since it partly falls under A's region of influence. In other words, B can be *influenced* by the actions of A. Agent A, on the other hand, can in no way be influenced by B. Crucially, A cannot *directly* affect the results of an action of B because it has no influence in the region of influence of B. Returning to our scenario, such a situation could occur if the overlap between the viewable environments represented the documents stored in A's desktop computer. Agent B, with the task of reporting to other agents on all documents of a specific type (e.g. papers on multi-agent systems), could periodically view the documents stored by A (i.e. sample the environment) waiting for a relevant document to appear before informing other agents about its existence. So, whenever A performs an action that adds a relevant document to its public document store, it will eventually *influence* B's actions, since B must now inform interested parties about this addition.

In Figure 3, the situation is one where the regions of influence overlap. This means that both agents can have a direct impact on the actions of each other. Thus, an action from either agent could affect the environment in such a way that a goal of the other agent is upset. Returning to our scenario, this could happen if the two agents were both attempting to retrieve a document from a public document store that sets a limit to the number of documents retrieved.

At this point, we should clarify that although in our diagrams the regions of influence have always fallen under the viewable environment of the respective agents this is only for illustrative reasons. In fact, there is no requirement for the viewable environment and the region of influence of an agent to overlap at all. If the region of influence of an agent does not

fall under that agent's viewable environment that agent would not be able to view the results of its actions, a situation that is not improbable. Even humans often do not fully realise the ramifications of their actions. The more usual case is when only part the region of influence of an agent falls under the viewable environment. In other words, the agent may not be fully aware of all the implications of its actions.

Now, through the concepts discussed so far we can outline four different ways in which one agent can influence another.

Weak influence A *weak influence* relationship occurs when an agent is able to affect aspects of the environment that another agent uses to decide what actions to perform (i.e. aspects of the environment the agent can perceive). Although an influence relationship can lead to a different outcome for the influenced agent's goal it cannot directly affect actions of that agent.

Agent B is *weakly influenced* by Agent A if and only if (i) the intersection of A's region of influence and B's viewable environment is non-empty, and (ii) the intersection of A's region of influence and B's region of influence is empty.

WeakInfluence

RegionOfInfluence

ViewableEnvironment

weakinfluenced : AgentState $\leftrightarrow$ AgentState

$\forall a, b : \text{AgentState} \mid a \neq b \bullet$

$(b, a) \in \text{weakinfluenced} \Leftrightarrow$

$\text{regioninfluence } b \cap \text{regioninfluence } a = \{\} \wedge$
 $\text{viewable } b \cap \text{regioninfluence } a \neq \{\}$

Strong influence A *strong influence* relationship occurs when an agent is able to affect both the viewable environment of another agent as well as its region of influence. In this case an agent can directly affect the goals of another agent because it can act on exactly those aspects of the environment that may represent desirable environmental states for the other agent.

Agent B is *strongly influenced* by Agent A if and only if (i) the intersection of A's region of influence and B's viewable environment is non-empty, and (ii) the intersection of A's region of influence and B's region of influence is non-empty.

StrongInfluence

RegionOfInfluence

ViewableEnvironment

stronginfluenced : AgentState $\leftrightarrow$ AgentState

$\forall a, b : \text{AgentState} \mid a \neq b \bullet$

$(b, a) \in \text{stronginfluenced} \Leftrightarrow$

$\text{regioninfluence } b \cap \text{regioninfluence } a \neq \{\} \wedge$
 $\text{viewable } b \cap \text{regioninfluence } a \neq \{\}$

Sneaky influence A *sneaky influence* relationship occurs when an agent is able to affect the region of influence of another agent but not the viewable environment. This, of course, implies that the influenced agent cannot view the results of its actions, so cannot be aware that some other agent is affecting those results.

Agent B is *sneakily influenced* by Agent A if and only if (i) the intersection of A's region of influence and B's viewable environment is empty, and (ii) the intersection of A's region of influence and B's region of influence is non-empty.

<i>SneakyInfluence</i>
<i>RegionOfInfluence</i>
<i>ViewableEnvironment</i>
<i>sneakyinfluenced</i> : $AgentState \leftrightarrow AgentState$
$\forall a, b : AgentState \mid a \neq b \bullet$ $(b, a) \in sneakyinfluenced \Leftrightarrow$ $regioninfluence\ b \cap regioninfluence\ a \neq \{\}$ $\wedge$ $viewable\ b \cap regioninfluence\ a = \{\}$

No influence Finally, when an agent cannot affect the viewable environment or the region of influence of another agent, no direct relationship can develop between them.

Agent B is *not influenced* by Agent A if and only if (i) the intersection of A's region of influence and B's viewable environment is empty, and (ii) the intersection of A's region of influence and B's region of influence is empty.

<i>NoInfluence</i>
<i>RegionOfInfluence</i>
<i>ViewableEnvironment</i>
<i>notinfluenced</i> : $AgentState \leftrightarrow AgentState$
$\forall a, b : AgentState \mid a \neq b \bullet$ $(b, a) \in notinfluenced \Leftrightarrow$ $regioninfluence\ b \cap regioninfluence\ a = \{\}$ $\wedge$ $viewable\ b \cap regioninfluence\ a = \{\}$

These four types of influence can now act as a guide to characterise a range of specific kinds of relationships. For example, a competitive relationship for access to common resources can only take place if both agents can strongly influence each other, i.e. their regions of influence and viewable environments overlap. A supervisor-student relationship is one where the supervisor can strongly influence the student (e.g. by providing direct guidance on what research the student should do) and the student can weakly influence the supervisor (e.g. by coming up with new results that may convince the supervisor to change research direction).

4.2 Effect of influence on actions and goals

In order to have a clearer understanding on exactly how one agent could affect the goals or actions of another we provide an analysis of the different cases. This is based on two assumptions: firstly, that agents perform actions because they want to achieve goals; and secondly, that goals agents are trying to achieve fall under their region of influence. As such, it makes sense to define the relationships that evolve through the interactions between agents in terms of the contribution that such interactions have towards the achievement of their goals. However, if an agent's goal cannot be achieved within that agent's region of influence the agent must seek assistance from another agent that has access to the region of the environment within which the goal can be achieved. In this paper we do not look at those situations.

Weak influence relationships

When only weak influence relationships occur, the influencing agent cannot directly impact goals. Nevertheless, it can still have a significant effect on the way the influenced agent achieves a goal, or whether the goal can be achieved at all. In essence, an agent could either be influenced so as to change its actions in order to achieve a goal or to change the goal completely. Below, we outline the different scenarios.

Goal does not change In the first type of case, the goal of the agent does not change as a result of the influencing agent. However, the actions performed to achieve the goal might change, as might the exact results of the actions, because of the goal.

No effect The influencing agent has no impact on the outcome of the goal because the attributes of the environment that are affected by the influencing agent are not taken into account for the execution of an action by the influenced agent.

Outcome of action changes Here the influencing agent affects the environment in such a way that the outcome of the action performed by the influenced agent changes. However, the *goal* of the influenced agent does not change. For example, consider an agent with the goal of compiling a list of all researchers with an interest in the subject of argumentation. The goal is satisfied as long as such a list exists. The agent compiles the list by asking other agents to declare their interest or not in the subject. The queried agents influence the *outcome* of the action by providing an answer. In any case, the goal is eventually achieved. However, the exact values described in the list have been influenced by others.

Action changes Agents may influence another agent to such an extent that the agent needs to change its planned actions in order to achieve the goal. For example, if some agents refuse to declare whether they are interested in argumentation, the agent may need to follow an alternative route, such as looking at their list of publications for evidence of an interest in the subject.

Goal changes The second type of scenario is when the influencing agent may change the environment in such a way that the influenced agent has to change its goal entirely. For example, let us assume that Agent A has two goals. The first goal, of primary importance, is to discover any paper on negotiation, and the second goal, of secondary importance, is to discover papers relating to middleware. If A was pursuing the secondary goal and discovers that new papers relating to the primary goal have been posted by B, A must then change goals to reflect the change in the environment. As such, B has sufficiently influenced A, through actions that impacted on just the agent's viewable environment, so that A changed its goal.

Strong and sneaky influence

Strong and sneaky influence relationships can impact on a goal in a more immediate way since agents could change exactly those attributes that represent a goal state for another agent. We identify three main cases below.

No change In the first case, the actions of Agent A do not affect the goal of Agent B. This means that although A is able to act in the region of influence of B, it does not perform actions that upset goal states for B.

Goal upset An agent can perform an action that changes the environment in such a way that a goal state of another agent is upset. For example, one agent may wish to access a document but cannot because another agent is already accessing it or has placed restrictions on its access.

Goal aided Alternatively, an agent can perform an action that helps towards creating the goal state of another agent. Such actions may have been intentional or may occur unintentionally. For example, if an agent has the goal of discovering a paper on auctions and another posts that paper, it inadvertently aids the second one in achieving its goal.

5 Conclusions

The issue of agent relationship management has been tackled in a variety of settings through existing research. In the distributed systems area there has been work on policy specifications. Policies are understood as “one aspect of information which influences the behaviour of objects within the system” [Sloman, 1994] and deal with the definition of rules that identify what actions can or cannot be performed with respect to a target. Although there is a wide body of work on policy specification (e.g. [Barker, 2000; Damianou *et al.*, 2001]), only recently has such work been applied to multi-agent systems [Dulay *et al.*, 2001]. Closer to the field of multi-agent systems is research into the issues of interaction between agents with a view of such interdependencies as social structures [Castelfranchi, 1990], which has led to the adoption of the notion of *norms*. A norm is, similar to a policy, considered as a means of regulating behaviour between agents. Various ways of representing norms have developed, such as obligations, authorisations and conventions [Dignum, 1999], commitments [Jennings, 1993] and mental attitudes [Conte and Castelfranchi, 1995], as well as methods for reasoning about norms (e.g. [Conte *et al.*, 1999; y Lopez *et al.*, 2002]).

However, existing research has not addressed the need for identifying the whole range of relationship management structures that can be developed (usually looking at just the case of optional or compulsory rules) and has not provided appropriate models for identifying exactly what kinds of relationships develop in a manner that can be widely applied and incorporated into practical application settings.

In this paper we provide the groundwork for addressing these issues, which are central to the effective design of multi-agent systems. Firstly, through our description of the space of management mechanisms we have a single basis by which to access and compare the various frameworks developed so far. Secondly, through the conceptual model for the identification of relationships we have a means to rationalise the process of when and how relationships should be managed and include it within a design methodology for multi-agent systems. Further work will move in two directions. On the one hand, we will move towards a relationship management framework, based on regulations, that covers the range of management

types presented here and place it within a wider solution for multi-agent systems infrastructure [Ashri *et al.*, 2002]. On the other hand, we will proceed with the characterisation of agent relationships, looking at issues such as how regions of influence are affected when one agent makes a commitment to perform an action for another agent and where *groups* of agents are involved.

References

- [Ashri *et al.*, 2002] R. Ashri, M. Luck, and M. d’Inverno. Infrastructure Support for Agent-based Development. In M. d’Inverno, M. Luck, M. Fisher, and C. Preist, editors, *Foundations and Applications of Multi-Agent Systems*, volume 2403 of *LNAI*, pages 73–88. Springer, 2002.
- [Barker, 2000] S. Barker. Data protection by logic programming. In J. Loyd, V. Dahl, U. Furbach, M. Kerber, K.-K. Lau, C. Palamedessi, L.M. Pereira, Y.Sagiv, and P.J. Stuckey, editors, *Computational Logic-CL2000*, volume 1861 of *LNCS*, pages 1300–1315. Springer, 2000.
- [Castelfranchi, 1990] C. Castelfranchi. Social power. In Y. Demazeau and J.-P. Muller, editors, *Decentralized AI*. Elsevier, 1990.
- [Conte and Castelfranchi, 1995] R. Conte and C. Castelfranchi. Norms as mental objects: From normative beliefs to normative goals. In C. Castelfranchi and J. P. Muller, editors, *From Reaction To Cognition*, volume 957 of *LNAI*. Springer, 1995.
- [Conte *et al.*, 1999] R. Conte, C. Castelfranchi, and F. Dignum. Autonomous norm-acceptance. In J. Muller, M. Singh, and A. Rao, editors, *Intelligent Agents V (ATAL98)*, volume 1555 of *LNAI*, pages 319–333. Springer, 1999.
- [Damianou *et al.*, 2001] N. Damianou, N. Dulay, E. Lupu, and M. Sloman. The Ponder Specification Language. In *Workshop on Policies for Distributed Systems and Networks (Policy2001)*, volume 1995 of *LNCS*, pages 18–39. Springer, 2001.
- [Dignum, 1999] F. Dignum. Autonomous agents with norms. *Artificial Intelligence and Law*, (7):69–79, 1999.
- [d’Inverno and Luck, 2001] M. d’Inverno and M. Luck. *Understanding Agent Systems*. Springer, 2001.
- [Dulay *et al.*, 2001] N. Dulay, N. Damianou, E. Lupu, and M. Sloman. A policy language for the management of distributed agents. In M. J. Wooldridge, G. Weiss, and P. Cinarcarini, editors, *Agent-Oriented Software Engineering II*, volume 2222 of *LNAI*, pages 84–100. Springer, 2001.
- [Jennings, 1993] N. Jennings. Commitments and conventions: The foundation of co-ordination in multi-agent systems. *The Knowledge Engineering Review*, 8(3):223–250, 1993.
- [Jennings, 2000] N. Jennings. On agent-based software engineering. *Artificial Intelligence*, 117(2):277–296, 2000.
- [Sloman, 1994] M. Sloman. Policy driven management for distributed systems. *Network and Systems Management*, 2(4):333–360, 1994.
- [Spivey, 1992] J.M. Spivey. *The Z Notation*. Prentice Hall, 2nd edition, 1992.
- [y Lopez *et al.*, 2002] F. Lopez y Lopez, M. Luck, and M. d’Inverno. Constraining Autonomy through Norms. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 674–681. ACM Press, 2002.

ODISET: On-line Distributed Session Tracing using Agents *

Salvador Mandujano (smv@itesm.mx) Arturo Galván (agalvan@itesm.mx)

Center for Intelligent Systems
Instituto Tecnológico y de Estudios Superiores de Monterrey
Monterrey, Mexico

Abstract

When a security incident occurs it is sometimes necessary to identify its causes for legal and cautionary purposes. In an attempt to hide the origin of her connection, a malicious user may have jumped from a source host h_s into a series of hosts $h_i \in H = \{h_1, h_2, \dots, h_n\}$ before breaking into final target h_t . This connection sequence describes a path that makes it difficult to find h_s given h_t due, in part, to the prohibitive amount of cooperation and synchronization that is required in practice by administrators. This paper describes a distributed rule-based model that automates this tracing process on-line with a $O(|H|)$ worst case scenario. Autonomous agents collaborate on the tracing and detection of the origin of an interactive connection using a loop unwinding technique and incorporating public key cryptography to create ciphered channels that allow them for secure communication. To meet the challenges of minimum system workload and improved robustness, the prototype features lightweight design and implementation as well as a dynamic port-allocation scheme to prevent sniffing and denial of service attempts. We describe the proposed model and present the experimental results obtained with the prototype system ODISET.

1 Introduction

Unauthorized access to computer systems is increasing in parallel with the growth of the Internet ([Rapalus, 2002]). For purposes of protection, accountability and liability, the need for connection back-tracing is a reality in the areas of cybernetic-law enforcement and computer security. The authentication structure of our networks as well as their distributed nature complicates the task of spotting the host from

which a connection sequence stems. If some sort of anomalous activity is observed from one of our users, it may be necessary to trace the origin of the session in order to identify the person behind it. In most cases, this is not easy to accomplish [Yoda and Etoh, 2001].

Within the domain of a LAN, it is rather straightforward to perform this tracing given that system administrators usually have access to all the hosts within their LAN. Unfortunately, not all sessions come from the inside. If a connection is received from the outside, we will have to contact the administrator of a remote host h_r in order to discover the identity of the user or the next link of the connection path or *chain*. Our fellow administrator will have to look into her system to realize that, perhaps, it has been compromised as well. It is possible that the user we are looking for is using h_r as a step-stone to reach our server. In that case, we will have to contact some more system administrators within the country and/or abroad [Cheswick and Bellovin, 1994] [Stoll, 1987]. This has obvious implications time and cost-wise.

Once a system has been compromised, the perpetrator may have left back-doors installed to facilitate future access to the system. Using interactive sessions, she may visit the system again even when the software has been patched and her back-door remains untouched. It is not necessary for the user to be logged on in order to be detected. Audit trail files can be used to do off-line tracing but this can be a tough task since audit logs may have been tampered with or deleted.

In the case of connections to web servers through HTTP and HTTPS, for instance, the hosts that compose the connection chain between client and server cannot be easily modified. Some components of the routing system would have to be modified in order to force the packets to follow a particular path. In the case of interactive sessions, however, applications such as telnet, login, rlogin and ssh allow a user with suitable access permissions to connect from host to host back and forth creating loops at will to scramble her connection path a little. This significantly complicates the structure of the chain and the amount of effort required to detect the root host h_s from where she is actually starting everything up. In fact, the complexity of back-tracing a connection across a set H of n hosts maybe small for a small n and a small number of connection hops, but it rapidly increases as n becomes larger due to the number of hosts that a chain may contain.

One of the challenges of intrusion detection is to accu-

*This project was supported by the INSYS Computer Security Research Grant (*Cátedra INSYS de Investigación en Seguridad Informática*) of Instituto Tecnológico y de Estudios Superiores de Monterrey

rately identify the main causes of an incident. The ultimate goal is to catch intruders in real time and our objective is to help intrusion detection systems do the part of session tracing on-line. There are some intrusion detection and intrusion response tools [Yoda and Etoh, 2001] [Asaka *et al.*, 1999] [Wang *et al.*, 2001] that provide some sort of off-line connection tracing that also performs data correlation with previously recorded user activity information.

This paper proposes an on-line approach that makes use of intelligent agents to identify the origin of a live session by unwinding connection loops. Instead of using historical audit trails, this model relies on evidence captured on operating system tables that are continuously refreshed by the system. For this reason, it is not likely that these tables experience changes that go unnoticed. The communication structure of the agents uses RSA public-key cryptography [Rivest *et al.*, 1979] [Schneier, 2001] to implement an encryption model that protects all communications between agents. One of the main issues of the multi-agent approach to automatic security monitoring is the overhead imposed by this sort of tools [Weiss, 1999] [Spafford and Zamboni, 2000]. Our prototype system *ODISET* was built from a lightweight design that does not represent a performance threat, making it ideal for wide deployment through an open source operating system. We present the design ideas behind the model, the tracing and communication techniques, the loop-unwinding method, as well as some experimental results of the first release of the prototype.

The rest of this paper is organized as follows: Section 2 explains the problem of connection back-tracing. Section 3 discusses relevant related work in the area. Section 4 is a description of our solution approach and the implementation issues of the prototype. Section 5 includes experimental results and findings. Conclusions and future work are in section 6 right before the listing of bibliography and references.

2 The Problem of Connection Back-tracing

Given a set of hosts $H = \{h_1, h_2, \dots, h_n\}$, with $n \gg 2$, a user could create a connection chain of m hops as follows: from host h_i into h_{i+1} ; from h_{i+1} into h_{i+2} ; from h_{i+2} into h_{i+3} ; and so on, until finally connecting to target host h_{i+m} . The problem of connection back-tracing is: given the set H , the target host h_{i+m} , and username u connected to h_{i+m} , find the hosts that compose the connection chain and identify its root host h_i .

For a small $n = |H|$ this problem is straightforward. Just a few hosts need to be analyzed in order to identify the source of a connection sequence. However, as n becomes larger (e.g. $n \gg 2$), the complexity of the problem increases since a larger number of hosts in H multiplies the possibilities of creating a more confusing connection path.

System administrators have limited access to information on who is connecting to one of their servers. On a LAN, administrators typically have login access to all servers, which makes the tracing process easier; however, for connections coming from the outside of the LAN, they will be unable to freely access all other servers for scrutiny. The visibility of this type of connections reduces to one link of the connec-

tion chain. That is, an administrator can only see the terminal number or the IP address/host-name of the immediate host from where the users are logging on. She is unable to tell, however, whether that host is in fact h_i (the machine from where the user is actually typing in commands). In order to find out whether that is the root host or not, she will have to contact the administrator of the remote system who, in turn, will have to perform a similar checking.

This job is cumbersome and requires time, communication, and trust from the parties involved. For the case of a connection that has been closed already, audit information will have to be examined in order to find the nodes of the chain that lead to the origin of the connection. This off-line sort of tracing has an important flaw: it counts on the existence and integrity of audit log files. For clean systems this would not be a problem but, for systems that have been compromised, and, in the case of a connection chain created to perpetrate an attack this could probably be the case, these files could have been tampered with during a previous break-in [Yoda and Etoh, 2001]. On the other hand, if a connection is still alive, i.e. there is a flow of packets between source and destination, information contained on system tables and network packets can be used to trace the user while she is on-line. That is, precisely, our approach.

3 Related Work

There are some projects that touch on the problem of connection route tracing. One of them is the method proposed in [Yoda and Etoh, 2001]. This model implements network traffic monitors that perform session tracing once a server has been compromised. A system that computes deviations from the traffic observed at two different hosts helps determine if these hosts were used in the same connection chain. Since checking network traffic at a node involves huge amounts of data, multiple packet monitors permanently filter and record specific system activity to build their own logs. These files are analyzed by a data correlator to identify the hosts of a connection chain. It is important to notice that this system deals with the problem of deleted logs by creating its own records. These records, however, may be the target of the same type of attack.

IDA (Intrusion Detection Agent system) [Asaka *et al.*, 1999] is another system that detects the origin of an information exchange related to an incident. It is primarily an intrusion detection system that employs mobile agents to detect local attacks. From a main host, investigation agents are sent over to the requester that needs an integrity check. These agents use the MLSI (*Marks Left by a Suspected Intruder*) strategy to reduce the amount of data needed to flag system activity as anomalous. In order to collect further information on a break-in, IDA does passive route tracing using audit log data. The structure of the tracing method is not specified by the authors but its agents broadcast connection evidence to their peers as a way of identifying possible hosts involved in the attack.

The *TBAIR system (Tracing-based Active Intrusion Response)* [Wang *et al.*, 2001] is a network-based intrusion detection tool that, unlike most traditionally-passive intrusion

detection systems, adds active response to security events using a technique called *Based on Sleepy Watermark Tracing*. TBAIR tries to attack the root of the vulnerability exploiting problem by locating the originators of the incident to hold them accountable for their intrusions and their method is based on the analysis of evidence collected from the network.

Thumbprinting [Staniford-Chen and Heberlein, 1995] is a technique that places several processes across the network in order to capture activity signatures or “thumbprints”. This method does not require to have processes on all hosts and, like in the case of [Yoda and Etoh, 2001], it replicates packet information for future review. This system is based on the fact that the packet content of a session passing through a set of hosts is invariant and can be compared in order to identify a host in the chain. This system is capable of detecting the root of a connection but it is not designed to identify the exact hosts that are part of the chain.

The approach that we present here through the implementation of the ODISET system differs from these previous projects in the following manner. It performs on-line rather than off-line tracing (i.e., it does not use audit logs). The system does not incorporate agent mobility. Instead, it uses a distributed model using agents that communicate over secure channels for sharing information. Unlike some other models, ODISET is able to not only identify the origin of the connection but also all the hosts that are part of the connection chain.

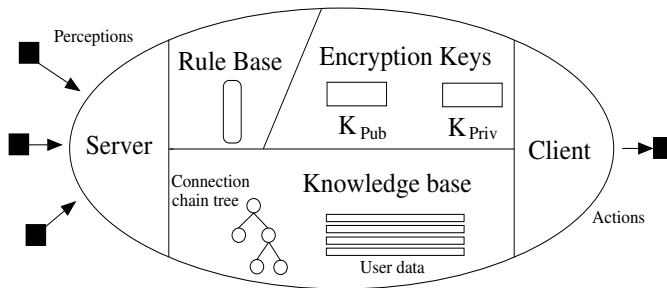


Figure 1: Basic composition of an ODISET tracing agent: public and private encryption keys, client functionality, server functionality, rule base, and knowledge base (connection chain tree + user data).

4 Solution Approach using Agents

“We require systems that decide for themselves what they need to do in order to satisfy their design objectives. Such computer systems are known as agents” [Weiss, 1999]. The use of agents for performing system assurance activities [Spafford and Zamboni, 2000] that typically require the skills of a human (e.g. inference, learning and decision making) makes possible the development of automatic or semi-automatic tools that aid system administrators in their task of securing a system.

The distributed nature of agents allows for more efficient, parallel data processing. We take advantage of this feature

that perfectly fits into a network environment [Huhns and Singh, 1998] in order to speed up connection back-tracing using autonomous agents. This approach eliminates the need for centralized data collection and analysis, a method that is still in use and that represents a design deficiency by putting the availability of a system in risk (monolithic systems typically suffer from having a single-point-of-attack).

In order to provide the agents with survivability features that make them more resistant to attack, each agent owns a digital certificate to create private communication channels when interacting with its peers. With the intention of reducing the probability of *Denial of Service (DoS)* attacks targeted toward the agents, a port-allocation scheme is used. This scheme dynamically changes the socket port numbers used by the agents in order to avoid being monitored or receiving unfinished-protocol requests.

This system features rule-based agent behavior through which individual agents are capable of detecting connection loops in a chain. The approach deals with the problem of deleted or damaged audit information by not using historical audit data. Agents look up information on dynamic session tables in order to extract data relevant to the tracing process. The process table, for instance, is not easily modifiable as audit logs are. The system might go down due to inconsistency if changes are made to this type of file, which would make the change evident.

4.1 Proposed Model

Tracing method and security

The system is composed of a number of tracing agents (Figure 1) that communicate with each other sharing information on user connections. There is an agent running at each host h_i and the ideal situation is having an agent running on every host of the network. Every agent has client and server capabilities that enable it to request information from other agents and to supply information to them. An agent is not useful by itself, it needs to collect information from the others in order to identify the links of a connection chain. The agent at the target host will eventually collect information from a number of its peers and will inform to the local administrator about the origin of the connection.

The perceptions of a tracing agent are received from two main sources: 1) the state of the system regarding user sessions and connections, and 2) messages from other agents requesting or providing information.

The environment an agent inhabits is highly dynamic and provides the communication channel necessary for an agent to stay in touch with its peers over the network. This environment can also represent a threat to the tracing system if malicious users gain access to it.

As a result of the changes perceived by an agent, it can perform a number of different actions: a) send messages to an agent, b) request messages from an agent, c) broadcast messages to all agents, d) generate encryption keys, e) encrypt messages, f) decrypt messages, g) identify connection loops, h) retrieve system status information, and h) save its knowledge to a safe location.

The knowledge of an agent is stored on a repository known

as *knowledge base (KB)*. This KB includes tracing host information and user data. A tree structure stores the information on the hosts that are part of a connection chain. Information regarding user names, login times and dates is stored on a separate structure. These two elements represent all the knowledge an agent has about the state of its environment. Should this knowledge be lost, the agent would lose track of previous events and would have to be updated by its peers.

The actions an agent takes depend on the information of the state of the system and its perceptions [Russell and Norvig, 1995]. Embodied into the agent are a set of rules that enable the agent to work autonomously deciding what to do. As new knowledge is fed into the KB more rules from the *rule base* need to be analyzed before making any conclusion and proceeding (the rule base is composed of 23 rules). The type of perceptions and actions do not change but the state of the environment does evolve hereby modifying the behavior of agents. The type of rules that make up the rule base are of the form:

```
R1:
IF ((HOW-CONNECTED(User)=Console) AND
(WHAT-DOING(User)=Ssh-to(Target)))
⇒ TRACING-DONE();

R2:
IF ((MESSAGE-TYPE(Message)=Tracing-
request) AND (VALID-PUBLIC-KEY(P-
key)=False))
⇒ DENY-CONNECTION() AND BROADCAST-
TREE() AND ALERT();
```

In order to protect their information exchanges, all agents have an RSA key-pair that is used to negotiate symmetric session keys before encrypting their messages. Encryption at this level guarantees that the information will not be visible at any point before reaching the top of the IP stack of the recipient party. This will deter sniffing attacks that are otherwise possible on plain text agent communications [Jansen *et al.*, 2000].

The tracing process starts when a connection needs to be monitored. Agent A at target host h_t receives local information from the system and checks for remote connections. If there is a connection from a remote machine h_r , A will contact agent B at host h_r in order to research the connection. For this purpose, a private channel will be opened using the asymmetric encryption keys of the agents. A symmetric session key will be agreed upon and all messages will be encrypted with it for that exchange. The same procedure will be followed if there are relevant connections from outside of the host of h_r coming into the server. A set of secure channels will be setup by the agents in order to share the information they have regarding connections and user activity. One hop at a time, the links composing a connection chain will be found.

The tracing information obtained by an agent is shared with the rest of the agents through *broadcast* communication (i.e., an agent sends a message to all existing agents). This guarantees that the KB's of the agents are consistent with the state of the environment and that, in the event of a security incident, the knowledge acquired by an agent will not be lost as it has been replicated and enriched by other agents.

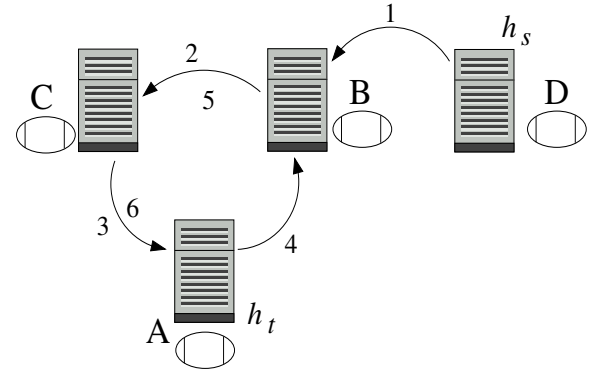


Figure 2: Loop unwinding. If a user connects from host h_s into host h_t doing a loop through hops 1, 2, 3, 4, 5 and 6 as shown above, agent A at h_t will start the tracing process and will be able to identify the loop to reduce tracing time in the future.

If we have a set H of n hosts, and a user creates a connection chain C of m hops, the system will behave well with a $O(n)$ worst case scenario. Suppose $m \gg n$. If the number q ($0 < q < m$) of connections that go from host h_x into host h_y – or vice versa – form a loop, the arrangement of the agents will find at most two relevant connections between the hosts hereby unwinding the loop. This is due to the fact that an agent is capable of identifying repeated connections toward itself by storing information on what users are connected. For the case of loops passing through k hosts, the chain will be reduced to at most $2k$ hops. This makes a worst case scenario of $O(n)$ with $n = |H|$.

Several methods exist (see [Wolf and Lam, 2000] for a reference) for loop unwinding, mostly in the compilers and discrete mathematics areas. In our model, we utilize a simple technique based on observed activity. An agent first stores user name and host-name information on its KB. When a tracing process is started, an agent will engage in communication with several neighbor agents in order to collect data that allow them to draw conclusions. Each time agent A receives new information, it checks its KB for loops. This is done as follows (see Figure 2). All tracing requests have an ID number w . Agent A initiates a tracing round. If while waiting for news on a particular w the agent is being asked by a cooperating agent B for information to complete tracing round w as well, the agent knows it is part of a loop since it is receiving information on a tracing round that it started (all agents, including initiators have this property). The trace is stored on its KB tree T and, if the user being traced repeats a loop that passes through the same host where agent A is located, this agent will reduce the tracing time by omitting a further requests to follow a loop.

Dynamic port-allocation protocol

Network service daemons usually work on a particular port listening for requests. Port-scan attacks try to reveal what are the services listening on the ports of a host. Once this is done, they look for vulnerabilities present on one of those services and, if they find one, they are in a position to exploit such a

flaw on the port they have previously identified. DoS attacks are started with port-scans to send a large number of request for service to a vulnerable port so that, at the end, there are so many requests pending to be answered – usually incomplete requests at protocol level – that the machine slows down and needs to be restarted. Additionally, if the service on that port does not support encryption, the traffic going through it may be observed by a network sniffer.

All network services using sockets on a port may suffer from this sort of attacks. We propose a model to minimize the likelihood of being a target. As we mentioned, all communications between agents is encrypted, so sniffing of plain text is impossible in practice. In order to deter attacks at the service port, we propose a technique that can be used in real life with radio equipment: channel switching (this is currently being implemented on the prototype). Initially, all agents listen on a well-know port p . The requests for service on that port do not allocate significant system resources according to the implementation. This basic port is used just to agree on the actual communication port q for the exchange ($q \neq p$). For this, the receiver proposes a valid port number that is sent to the other party over a secure channel. They immediately switch to that port for exchanging information. This port switching is repeated during the session within a fixed short period of time. In order to defeat this protection mechanism, a port scan will have to be run again and, by the time it succeeds, the communication may have moved to a different port. Frequency of switching and encryption this mechanism possible.

4.2 Implementation of the Model

As a proof of concept, the model was implemented on a prototype system named *ODISET* (On-line DIstributed SEssion Tracing). Every agent is a stand-alone process implemented with sockets for providing client and server functionality. It can request information from others and it can share information as well. A 1024-bit RSA key-pair $\{K_{Pub}, K_{Priv}\}$ is generated before the agent is launched. Once the agent is started up, it is ready to collaborate in the tracing of a session. When contacting a peer agent, it exchanges public keys and verifies the signature on the received key. The certification must come from the *ODISET* master key whose public key is accessible to all agents. In order to speed up message encryption, and given that encryption with public-key methods is much slower than the one using private-key algorithms [Schneier, 2001], the agent that is to send a message does the following: 1) it generates a session key K_{DES} for symmetric encryption, 2) it then encrypts the message M using K_{DES} , 3) it encrypts K_{DES} using the public key of the recipient, and, finally, 4) it sends the encrypted session key and the encrypted message to their destination. The implementation of the RSA and DES algorithms is based on version 2.0 of the RSAREFTM cryptographic library by RSA Laboratories.

The first release of the *ODISET* prototype was developed on a RedHat Linux 8.0 box. Although agent templates can be easily obtained from multiple agent-generator systems and for different compilers and interpreters, they typically put unnecessary functionality into the agent that produces heavy agents. For this reason, and with the intention of providing

the best performance possible, the implementation was made in C by minimizing the number of libraries to include. Considering that programs and data files may be damaged after a security incident and that a security tool can not rely on them, our agents do not use the output of programs such as `w`, `who` and `ps` to read system tables like `utmp` and `wtmp`. They incorporate this functionality into its own body, which gives them extended independence. The size of each agent is around 20 Kbytes to which we add two encryption keys that need to be uploaded at certain point (around 4 additional Kbytes) as well as the knowledge and rule bases that are never uploaded entirely into memory.

The knowledge base of each agent includes a tree structure T where all connection chains are stored. Whenever a host has received all the information regarding a particular session, it stores the connection chain $c = [h_1 \rightarrow h_2 \rightarrow \dots \rightarrow h_n]$ into T and sends it to all available agents. Every agent reads c and finds the hosts in c that match its own tree. It then creates new branches to keep its KB up-to-date. The rule base is encoded along with the body of the agent. Being an static structure, it is not necessary to keep it as a separate entity.

5 Experimental Results

In order to evaluate the efficiency of the method, multiple experiments were prepared. After some implementation corrections, all of them were successful after the unwinding technique was incorporated. The testing facility is a set of seven Linux machines running kernels 2.1 and above on Redhat 7.2, RedHat 7.3, RedHat 8.0 and Mandrake 8.0 operating system installations. The general structure of the experiments consists of creating connection chains of length $m \in \{2, 4, 8, 16, 32\}$ using n hosts where $n \in \{3, 5, 7\}$. This maximum number of machines was selected since our experience tells it is highly improbable that, for session performance purposes, an attacker uses much more step-stone hosts. All cases where there is no loop in the chain, that is, where $m = n$ are easily resolved so we prepared connections that include a series of loops along the chain. For instance, in the three-server setup, the connection goes in circles from host h_1 into h_2 , from host h_2 into host h_3 , from host h_3 back into host h_1 and so on for a total number of m hops. The experiments show that the algorithm effectively unwinds the loops. When an agent finds the same loop repeated several times, it will not try to solve it over and over again. Its KB contains user and connection information that allows it to conclude that a user is creating a loop.

Part of the contribution of this method is the fact that by using loop unwinding, the performance of the algorithm has an upper threshold that is, at most, linear on the number of hosts n through which the user connects. A set of n agents will be in charge of tracing the connection and even for a very long chain, an agent located on each hosts guarantees that loops will not delay the tracing process in any way.

Table 1 includes tracing numbers with and without encryption. Both columns indicate that the growth of the tracing time is below linear time with respect to the number of hops m (this makes us think this model should escalate well to large networks). The time difference from tracing with and without

Table 1: Tracing time with and without encryption (RSA key exchange + DES chaining block ciphering) with seven hosts ($n=7$); m corresponds to the number of connection hops.

m	Time without encryption (sec)	Time with encryption (sec)
2	0.03	0.84
4	0.10	1.13
8	0.62	3.09
16	0.69	4.20
32	0.73	3.99

encryption is significant. This only reminds us of the performance cost of using cryptographic algorithms to protect our data. It is not likely that a connection chain extends over more than thirty hosts, for instance. It follows that, even with 1024-bit keys, the use of encrypted communication is the way to go regarding agent communication for this problem.

The strengths of this model are its unwinding algorithm, the possibility of performing on-line tracing and the structure of an autonomous lightweight agent. The proposed dynamic port-allocation method can effectively deter communication sniffing and promises to be effective for the case of agent-based applications as well. A disadvantage of the system is that it works exclusively with alive connections. If, for one reason, the user disconnects the session from our host, the agents will go blind and will not be able to trace the intruder. The model, however, can be easily extended toward off-line tracing, but there are already other systems, like the ones highlighted in Section 3, that cover that case. Another improvement area is the fact that an agent will see only users that are connected to a host through an interactive session using commands that update system tables. If an intruder exploits a vulnerability and gets to spawn a shell session at the host without having to run a remote connection command, she will not be seen. That is because these connection commands like `rlogin` and `ssh` write system activity information to system tables. If none of this programs is used by an intruder, the host will not record her presence and she will be able to go unnoticed.

6 Conclusions and Future Work

This model proves the feasibility of performing on-line connection back-tracing using lightweight autonomous agents in a distributed fashion. It also proposes two security mechanisms that can be implemented on other software agents to make their communication structure more robust (these mechanisms are random port-switching and encrypted agent communication using certificates). We conclude that the adoption of a tracing system like this by multiple operating systems through a sort of *agent sand box* per host would indeed contribute to solve security incidents more rapidly.

Future work includes a) the integration of the ODISET tracing tool into an agent-based intrusion detection system, b) the development of survivability methods for agents using replication, mobility and zero-loss mechanisms, and c) the extension of the tracing space to include dial-up connections.

References

- [Asaka *et al.*, 1999] M. Asaka, M. Tsuchiya, T. Onabuta, S. Okasawa, and S. Goto. Local attack detection and intrusion route tracing. *IEICE Trans. Commun. New Paradigms in Network Management*, E82-B(11), November 1999.
- [Cheswick and Bellovin, 1994] W.R. Cheswick and S.M. Bellovin. *Firewalls and Internet Security: Repelling the Wily Hacker*. Addison Wesley, 1994.
- [Huhns and Singh, 1998] M.N. Huhns and M.P. Singh. Agents and multiagent systems: Themes, approaches, and challenges. *Readings in Agents*, pages 1–23, Morgan Kaufmann, San Francisco, CA 1998.
- [Jansen *et al.*, 2000] W. Jansen, P. Mell, T. Karygiannis, and D. Marks. Mobile agents in intrusion detection and response. June 2000.
- [Rapalus, 2002] P. Rapalus. Computer security survey 2002. Technical Report 1, Computer Security Institute, CSI, and the Federal Business of Investigations, FBI, April 2002.
- [Rivest *et al.*, 1979] R.L. Rivest, A. Shjamir, and L.M. Adleman. On digital signatures and public key cryptosystems. *MIT Laboratory for Computer Science*, (MIT/LCS/TR-212, technical report), January 1979.
- [Russell and Norvig, 1995] P. Russell and E. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, New Jersey, 1995.
- [Schneier, 2001] B. Schneier. Managed security monitoring: Network security for the 21st century. *Computer Security Journal*, 17.2, 2001.
- [Spafford and Zamboni, 2000] E. H. Spafford and D. Zamboni. Intrusion detection using autonomous agents. *Computer Networks*, 34:547–570, 2000.
- [Staniford-Chen and Heberlein, 1995] S. Staniford-Chen and L.T. Heberlein. Holding intruders accountable on the internet. IEEE Symposium on Security and Privacy, 1995.
- [Stoll, 1987] C. Stoll. *The Cuckoo's Egg*. Double Day, 1987.
- [Wang *et al.*, 2001] X. Wang, D. Reeves, and S. Wu. Tracing-based intrusion response. *Journal of Information Warfare*, 1(1), September 2001.
- [Weiss, 1999] G. Weiss. *Multiagent Systems: A modern Introduction to Distributed Artificial Intelligence*. MIT Press, 1999.
- [Wolf and Lam, 2000] M.E. Wolf and M.S. Lam. A loop transformation theory and an algorithm to maximize parallelism. volume 2, pages 452–471, October 2000.
- [Yoda and Etoh, 2001] K. Yoda and H. Etoh. Finding a connection chain for tracing intruders. 6th European Symposium on Research in Computer Security (ESORICS 2000), pp. 191–205, October 2001.

MULTIAGENT SYSTEMS

NASH EQUILIBRIA

A Continuation Method for Nash Equilibria in Structured Games

Ben Blum
Stanford University
bblum@stanford.edu

Christian R. Shelton
Stanford University
cshelton@cs.stanford.edu

Daphne Koller
Stanford University
koller@cs.stanford.edu

Abstract

We describe algorithms for computing Nash equilibria in structured game representations, including both graphical games and multi-agent influence diagrams (MAIDs). The algorithms are derived from a continuation method for normal-form and extensive-form games due to Govindan and Wilson; they follow a trajectory through the space of perturbed games and their equilibria. Our algorithms exploit game structure through fast computation of the Jacobian of the game's payoff function. They are guaranteed to find at least one equilibrium of the game and may find more. Our approach provides the first exact algorithm for computing an exact equilibrium in graphical games with arbitrary topology, and the first algorithm to exploit fine-grain structural properties of MAIDs. We present experimental results for our algorithms. The running time for our graphical game algorithm is similar to, and often better than, the running time of previous approximate algorithms. Our algorithm for MAIDs can effectively solve games that are much larger than those that could be solved using previous methods.

1 Introduction

Game theory is a mathematical framework that describes interactions between multiple rational agents and allows for reasoning about their outcomes. However, the complexity of standard game descriptions grows exponentially with the number of agents involved. For many multi-agent situations, this blowup presents a serious problem. Recent work in artificial intelligence [La Mura, 2000; Kearns *et al.*, 2001; Koller and Milch, 2001] proposes the use of structured game representations that utilize a notion of locality of interaction; these representations allow a wide range of complex games to be represented compactly.

In this paper we consider the task of computing *Nash equilibria* for structured games. A Nash equilibrium is a strategy profile such that it is no agent's interest to deviate unilaterally. A naive approach to finding Nash equilibria is to convert the structured game into a standard game representation, and apply a standard game-theoretic solution algorithm [McKelvey and McLennan, 1996]. This approach is, in general, infeasible for all but the simplest games. We would like an algorithm that exploits the structure in these game representations for efficient computation.

In this paper, we describe a set of algorithms that use *continuation methods* for solving structured games. These algorithms follow a trajectory of equilibria of perturbed games until an equilibrium of the original game is found. Our algorithms are based on the recent work of Govindan and Wilson [2002; 2003a; 2003b] (GW hereafter), which apply to standard game representations (normal-form and extensive-form). We show how the structure of the games can be exploited to perform the key computational step of the algorithms of GW.

Our methods address both *graphical games* [Kearns *et al.*, 2001] and *multi-agent influence diagrams (MAIDs)* [Koller and Milch, 2001]. We present the first algorithm for finding exact equilibria in graphical games of arbitrary structure. We also provide the first algorithm that can take advantage of the fine-grained structure of MAIDs, and therefore can handle MAIDs which are significantly outside the scope of previous methods. We provide experimental results demonstrating the efficiency of our approach relative to previous methods.

2 Game Representations and Equilibria

2.1 Game Theory

We begin by briefly reviewing concepts from game theory used in this paper, referring to Owen [1995] for a good introduction. A game defines an interaction between a set N of agents. Each agent $n \in N$ has a set of available *strategies* Σ_n , where a strategy determines the agent's behavior in the game. The precise definition of the set Σ_n depends on the game representation, as we discuss below. A *strategy profile* σ defines a strategy $\sigma_n \in \Sigma_n$ for each $n \in N$. Given a strategy profile σ , the game defines a payoff $G_n(\sigma)$ for each agent $n \in N$. We use σ_{-n} to denote the strategy profiles of the agents $n' \in N - \{n\}$. Similarly, we use Σ_{-n} to refer to the set of all strategy profiles of agents in $N - \{n\}$.

A solution to a game is a prescription of a strategy profile for the agents. The agent's goal is to maximize its payoff. Thus, a basic desideratum for a solution profile is stability — it should not be in any agent's interests to deviate from it. More precisely, the fundamental notion of a *Nash equilibrium* [Nash, 1951] is defined to be a strategy profile σ such that, for all $n \in N$, $G_n(\sigma_n, \sigma_{-n}) \geq G_n(\sigma'_n, \sigma_{-n})$, for all other strategies $\sigma'_n \in \Sigma_n$. Thus, if an agent knew that the others were playing according to an equilibrium profile, it would have no incentive to deviate.

An ϵ -equilibrium is a strategy profile such that no agent can improve its expected payoff by more than ϵ by unilaterally changing its strategy. Unfortunately, finding an ϵ -equilibrium is not necessarily a step toward finding an exact equilibrium: the fact that σ is an ϵ -equilibrium does not guarantee the existence of an exact equilibrium in the neighborhood of σ .

Normal-Form Games

A general-sum normal-form game defines a simultaneous-move multiagent scenario in which each agent independently selects an action and then receives a payoff that depends on the actions selected by all of the agents. More precisely, let G be a normal-form game with a set N of agents. Each agent $n \in N$ has a discrete action set S_n and a payoff array G_n with entries for every action profile in $\prod_{n \in N} S_n$.

Equilibrium strategies often require that agents randomize their choice of action. A *mixed strategy* σ_n is a probability distribution over S_n . The set Σ_n is the set of all mixed strategies. The *support* of a mixed strategy is the set of actions in S_n that have non-zero probability. A strategy σ_n for agent n is said to be a *pure strategy* if it has only a single action in its support. The set Σ of mixed strategy profiles is $\prod_{n \in N} \Sigma_n$. A mixed strategy profile $\sigma \in \Sigma$ is thus an m -vector, where $m = \sum_{n \in N} |S_n|$. Every game is guaranteed to have at least one mixed-strategy equilibrium, and the number of equilibria may be exponential in the number of agents.

Extensive-Form Games

An extensive-form game is represented by a tree in which each node represents a choice either of an agent or of nature. Each of nature's choice nodes is associated with a probability distribution over its outgoing branches. Each leaf $z \in Z$ of the tree is associated with a vector of payoffs $G(z)$, where $G_n(z)$ denotes the payoff to agent n at leaf z . The choices of the agents and nature dictate which path of the tree is followed and therefore the payoffs to the agents.

The decision nodes belonging to each agent are partitioned into information sets, where each information set is a set of states among which the agent cannot distinguish. Thus, an agent's strategy must take the same action at all nodes in the same information set. We define an *agent history* $H_n(y)$ for a node y in the tree and an agent n to be a sequence containing the information sets traversed in the path from the root to y , and the action selected at each one. Thus, two nodes have the same agent- n history if the paths used to reach them are indistinguishable to n . (The paths may differ in other ways, such as nature's decisions or the decisions of other agents.) We make the common assumption of *perfect recall*: an agent does not forget information known nor the choices made at previous decisions. More precisely, if two nodes y, y' are in the same information set for agent n , then $H_n(y) = H_n(y')$.

We need a representation of a strategy for an extensive-form game; unlike the case of normal-form games, there are several quite different choices. For our purposes, the most appropriate representation is the *sequence form* [Koller and Megiddo, 1992; von Stengel, 1996]. Here, the strategy σ_n for an agent n is represented as a vector of real values of size h_n , one for each distinct history $H_n(z)$ for a leaf z in the tree. The number $\sigma_n(H_n(z))$, abbreviated $\sigma_n(z)$, is the product of the probabilities controlled by agent n along the history H_n .

From this representation, we can easily derive the probability of taking an action at a particular information set.

The set of sequence form strategies for agent n is therefore a subset of $\mathbb{R}^{h_n}$, where h_n is at most the number of leaves in the tree. The set of legal sequence form strategies Σ_n for agent n is defined by a set of linear constraints on vectors in $\mathbb{R}^{h_n}$. The set of sequence form strategy profiles is then defined as $\Sigma = \prod_{n \in N} \Sigma_n$. The payoff to agent n in an extensive-form game can be shown to be

$$G_n(\sigma) = \sum_{z \in Z} G_n(z) \prod_{i \in N} \sigma_i(z). \quad (1)$$

Thus, the payoffs are a sum, over the leaves in the tree, of the payoff at a leaf times the product of the sequence form parameters for that leaf.¹ Importantly, this expression has a similar multi-linear form to the payoff in a normal-form game, but using sequence form strategies rather than mixed strategies. In an extensive form game satisfying perfect recall, any mixed strategy equilibrium can be represented using an essentially equivalent sequence form strategy profile.

2.2 Structured Representations

Graphical Games

The size of the payoff arrays required to describe a normal-form game grows exponentially with the number of agents. Kearns *et al.* [2001] introduced the framework of *graphical games*, which provide a more structured representation based on probabilistic graphical models. Graphical games capture local structure in multi-agent interactions, allowing a compact representation for scenarios where each agent's payoff is only affected by a small subset of other agents. Examples of interactions where this structure occurs include agents that interact along organization hierarchies and agents that interact according to geographic proximity.

A graphical game is described like a normal-form game. The basic representation (slightly generalized) is a directed graph with one node for each agent. An edge from agent n' to agent n in the graph indicates that agent n 's payoffs depend on the action of agent n' . More precisely, we define Fam_n to be the set of agents consisting of n itself and its parents in the graph. The agent's payoff G_n is an array indexed only by the actions of the agents in Fam_n . Thus, the description of the game is exponential in the in-degree of the graph and not in the total number of agents. In this case, we use Σ_{-n} and S_{-n} to refer to strategy profiles of the agent in $Fam_n - \{n\}$.

Multi-agent Influence Diagrams

The description length of extensive-form games often also grows exponentially with the number of agents. In many situations this large tree can be represented more compactly. *Multi-agent influence diagrams* (MAIDs) [Koller and Milch, 2001] allow a structured representation of games involving time and information by extending *influence diagrams* [Howard and Matheson, 1984] to the multi-agent case.

A MAID is represented as a directed acyclic graph over nodes of three types: chance, decision, and utility. (Utility

¹For notational simplicity, $G_n(z)$ includes nature's probabilities.

nodes are assumed to have no children.) Chance nodes represent nature's actions, and a *conditional probability distribution (CPD)* is associated with each such node, describing the distribution over outcomes of that variable conditioned on the values of its parents. Each decision node is associated with a single agent. The parents of a decision node represent the variables whose values are known to the agent when making that decision. Thus, an agent's decision rule for a node can specify a different strategy for each assignment of values to the node's parents. In effect, each such assignment corresponds to an information set. A randomizing decision rule for a decision node is simply a CPD: a distribution over its values for each instantiation of its parents.

Each utility node is associated with an agent, and represents a component in that agent's payoff. The utility node takes real values as a deterministic function of its parents. Thus, that component of the agent's payoff depends only on a subset of the variables in the MAID. The agent's overall utility is the sum of the utilities obtained at its different utility nodes. It is easy to show that a MAID defines an extensive-form game. If we use CPDs and decision rules that are tree-structured, then the MAID representation is no larger than the corresponding extensive-form representation, and is exponentially smaller in many cases. Note that a graphical game is simply a MAID where each agent has a single decision node and a single utility node, and where the parents of an agent n 's utility node are the decision nodes for the agents in Fam_n .

3 Continuation Methods

We begin with a high-level overview of continuation methods, referring the reader to [Watson, 2000] for a more detailed discussion. Continuation methods work by solving a simpler perturbed problem and then tracing the solution as the magnitude of the perturbation decreases, converging to a solution to the original problem.

More precisely, let λ be a scalar parameterizing a continuum of perturbed problems. When $\lambda = 0$, the perturbed problem is the original one; when $\lambda = 1$, the perturbed problem is one for which the solution is known. Let w represent the vector of real values of the solution. For any perturbed problem defined by λ , we characterize solutions by the equation $F(w, \lambda) = 0$, where F is a real-valued vector function (so that 0 is a vector of zeros). The function F is such that if $F(w, \lambda) = 0$ holds then w is a solution to the problem perturbed by λ .

The continuation method traces solutions along the manifold of solution pairs (w, λ) satisfying $F(w, \lambda) = 0$. Specifically, if we have a solution pair (w, λ) , we would like to trace that solution to adjacent solutions. Differential changes to w and λ must cancel out so that F remains equal to 0 . Thus, locally, changes dw and $d\lambda$ along the path must obey $\nabla_w F(w, \lambda) dw = -d\lambda \nabla_\lambda F(w, \lambda)$, which is equivalent to the matrix equation

$$\begin{bmatrix} \nabla_w F & \nabla_\lambda F \end{bmatrix} \begin{bmatrix} dw \\ d\lambda \end{bmatrix} = 0. \quad (2)$$

If the matrix $\begin{bmatrix} \nabla_w F & \nabla_\lambda F \end{bmatrix}$ has a null-space of rank 1 everywhere, the curve is uniquely defined. If properly constructed, the curve starting at $\lambda = 1$ is guaranteed to cross $\lambda = 0$, at

which point the corresponding value of w is a solution to the original problem. A continuation method begins at the known solution for $\lambda = 1$. The null-space of the Jacobian ∇F at a current solution (w, λ) defines a direction, along which the solution is moved by a small amount. The process then repeats, tracing the curve until $\lambda = 0$. The cost of each step in this computation, given the Jacobian, is cubic in the size of the Jacobian, due to the required matrix operations.

Continuation methods involve the tracing of a dynamical system through the continuous variation of the parameter λ . For computational purposes, discrete steps must be taken. As a result, error inevitably accumulates as the path is traced. One can use several techniques to reduce the error, which we do not describe for lack of space. Unfortunately, these techniques can potentially send the algorithm into a cycle, and in practice they occasionally do. If the algorithm cycles, random restarts and a decrease in step size can improve convergence.

4 Continuation Methods for Games

We now review the work of Kohlberg and Mertens [1986] and GW on applying the continuation method to the task of finding equilibria in games. These algorithms form the basis for our extension to structured games, described in the next section. The continuation method perturbs the game by adding λ times a fixed bonus to each agent's payoffs, such that an agent's bonus depends only on its own actions. If the bonuses are large enough (and unique), the bonuses dominate the original game structure, and the agents need not consider their opponents' plays. Thus, for $\lambda = 1$, the perturbed game has only one equilibrium: each agent plays the action with the largest bonus. We then use the continuation method to follow a path in the space of λ and equilibrium profiles for the resulting perturbed game, decreasing λ until it is zero; at this point, the corresponding strategy profile is an equilibrium of the original game. We now make this intuition more precise.

4.1 Normal Form Games

In order to apply Eq. (2), we need to characterize the equilibria of perturbed games as the zeros of a function F . We first define an auxiliary function measuring the benefit of deviating from a given strategy profile. Specifically, $V(\sigma)$ is a vector payoff function of the payoff to agent n for deviating from the mixed strategy profile σ by playing each action s :

$$V_s(\sigma) = \sum_{t \in S_{-n}} G_n(s, t) \prod_{i \in N - \{n\}} \sigma_{t_i}.$$

We now define a *retraction operator* $R : \mathbb{R}^m \rightarrow \Sigma$ to be an operator that maps arbitrary m -vectors w to the point in the space Σ of mixed strategies which is nearest to w in Euclidean distance. As outlined in the structure theorem of Kohlberg and Mertens [1986], an equilibrium σ is recoverable from $\sigma + V(\sigma)$ by the retraction operator R : $R(\sigma + V(\sigma)) = \sigma$. In fact, this condition is a full characterization of equilibria. Thus, we can define an equilibrium as a solution to the equation $\sigma = R(\sigma + V(\sigma))$. Conversely, if $\sigma = R(w)$ and $w = \sigma + V(\sigma)$ we have the equivalent condition that $w = R(w) + V(R(w))$. Thus, we can search for a point $w \in \mathbb{R}^m$ which satisfies this equality, in which case $R(w)$ is guaranteed to be an equilibrium.

We now define the game perturbation and associated continuation method. For a target game G with payoff function V , we create an easily soluble perturbed game by adding an m -vector $\mathbf{b}$ of unique bonuses that agents receive for playing certain actions, independently of what all other agents do. In perturbing the game G by $\mathbf{b}$, we arrive at a new game $G + \mathbf{b}$ in which for each $s \in S_n$, and for any $t \in S_{-n}$, $(G + \mathbf{b})_n(s, t) = G_n(s, t) + \mathbf{b}_s$. If we make $\mathbf{b}$ sufficiently large, then $G + \mathbf{b}$ has a unique equilibrium, in which each agent plays the pure strategy s for which $\mathbf{b}_s$ is maximal.

Now, let V be the payoff function for the target game G . The induced payoff function $V + \lambda \mathbf{b}$ is also perturbed from V by $\lambda \mathbf{b}$. Thus, the form of our continuation equation is:

$$F(\mathbf{w}, \lambda) = \mathbf{w} - R(\mathbf{w}) - (V(R(\mathbf{w})) + \lambda \mathbf{b}) . \quad (3)$$

We have that $V + \lambda \mathbf{b}$ is the payoff function for the perturbed game $G + \lambda \mathbf{b}$, so $F(\mathbf{w}, \lambda)$ is zero if and only if $R(\mathbf{w})$ is an equilibrium of $G + \lambda \mathbf{b}$. At $\lambda = 0$ the game is unperturbed, so $F(\mathbf{w}, 0) = \mathbf{0}$ iff $R(\mathbf{w})$ is an equilibrium of G .

The expensive step in the continuation method is the calculation of the Jacobian $\nabla_{\mathbf{w}} F$, required for the computation that maintains the constraint of Eq. (2). Here, we have that $\nabla_{\mathbf{w}} F = I - (I + \nabla V) \nabla R$, where I is the $m \times m$ identity matrix. The hard part is the calculation of ∇V . For pure strategies $s_1 \in S_{n_1}$ and $s_2 \in S_{n_2}$, the value at location (s_1, s_2) in $\nabla V(\sigma)$ is equal to the expected payoff to agent n_1 when it plays the pure strategy s_1 , agent n_2 plays the pure strategy s_2 , and all other agents act according to the strategy profile σ :

$$\nabla V_{s_1, s_2}(\sigma) = \sum_{t \in S_{-n_1, -n_2}} G_{n_1}(s_1, s_2, t) \prod_{i \in N - \{n_1, n_2\}} \sigma_{t_i} . \quad (4)$$

Computing Eq. (4) requires a number of multiplications which is exponential in the game size; the sum is over the exponentially large space $S_{-n_1, -n_2} = \prod_{i \in N - \{n_1, n_2\}} S_i$.

4.2 Extensive Form Games

The same method applies to extensive-form games, using the sequence form parameterization of strategies. We first need to define the bonus vector, and the retraction operator which allows us to characterize equilibria.

The bonus vector in the extensive-form adds a bonus for each sequence of each agent. GW show that a sufficiently large bonus guarantees a unique equilibrium for the perturbed game. The retraction operator R takes a general vector and projects it onto the valid region of sequence forms. As all of the constraints are linear, the projection amounts to a decomposable quadratic program (QP) that can be solved quickly. We employ standard QP methods. The Jacobian of the retraction is easily computable from the set of active constraints.

The solution for the sequence form is now surprisingly similar to that of the normal-form. The key property of the sequence form strategy representation is that the payoff function is a multi-linear function of the extensive-form parameters, as shown in Eq. (1). The elements of the Jacobian ∇V also have the same general structure. In particular the element corresponding to sequence s_1 for agent n_1 and sequence s_2 for agent n_2 is

$$\nabla V_{s_1, s_2}(\sigma) = \sum_{z \in Z_{s_1, s_2}} G_{n_1}(z) \prod_{i \in N - \{n_1, n_2\}} \sigma_i(z) \quad (5)$$

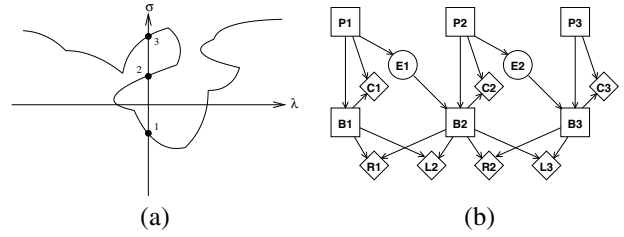


Figure 1: (a) An abstract diagram of the path. The horizontal axis represents λ and the vertical axis represents the space of strategy profiles (actually multidimensional). The algorithm starts on the right at $\lambda = 1$ and follows the dynamical system until $\lambda = 0$ at point 1, where it has found an equilibrium of the original game. It can continue to trace the path and find the equilibria labeled 2 and 3. (b) Two-stage road building MAID for three agents.

where Z_{s_1, s_2} is the set of leaves that are consistent with the sequences s_1 (for agent n_1) and s_2 (for agent n_2). We take Z_{s_1, s_2} to be the empty set (and hence $\nabla V = 0$) if s_1 and s_2 are incompatible. Eq. (5) is precisely analogous to Eq. (4) for normal-form games. We have a sum, over outcomes, of the utility of the outcome multiplied by the strategy probabilities for all other agents. Note that this sum is over the leaves of the tree, which may be exponential in the number of agents.

Zero-probability actions in extensive-form games give rise to an additional subtlety. Such actions induce a probability of zero for entire trajectories in the tree, possibly leading to equilibria based on unrealizable threats and other undesirable phenomena. For us, they can also lead to bifurcations in the continuation path, preventing convergence. Thus, we constrain all sequence form parameters to be greater than or equal to ϵ for some small ϵ . This constraint ensures that the continuation path is a 1-manifold. The algorithm thus finds an equilibrium to a perturbed game, where agents have a small probability of choosing an unintended action. As ϵ tends to zero, these equilibria converge to *perfect equilibria* of the original game [Owen, 1995], a (nonempty) subset of all equilibria. For ϵ small enough, continuity implies that there is always an exact perfect equilibrium in the vicinity of the perturbed equilibrium, which can easily be found using local search.

4.3 Path Properties

In the case of normal-form games, the structure theorem of Kohlberg and Mertens [1986] implies that, with probability one over all choices for $\mathbf{b}$, the path of the algorithm is a one-manifold without boundary. GW provide an analogous structure theorem that guarantees the same property for extensive-form games. Figure 1(a) shows an abstract representation of the path followed by the continuation method. The equilibrium for large positive λ is unique, so the one-manifold cannot double back to the side of $\lambda = \infty$. Furthermore, the perturbed games along the path can have only a finite number of discrete equilibria, so the path cannot travel back and forth indefinitely. Therefore, it must cross the $\lambda = 0$ hyperplane at least once, yielding an equilibrium. In fact, the path may cross multiple times, yielding many equilibria in a single run. As the path must eventually continue to the $\lambda = -\infty$ side, it will find an odd number of equilibria when run to completion.

In both normal-form and extensive-form games, the path is piece-wise polynomial, with each piece corresponding to a

different support set of the strategy profile. These pieces are called *support cells*. The path is not smooth at cell boundaries, due to discontinuities in the Jacobian of the retraction operator, and hence in $\nabla_w F$, when the support changes. Thus, in following the path, care must be taken to step up to these boundaries exactly.

In the case of two agents, the path is piece-wise linear and, rather than taking steps, the algorithm can jump from “elbow” to “elbow” along this path. When this algorithm is applied to a two-agent game and a particular bonus vector is used, the steps from support cell to support cell that the algorithm takes are exactly equal to the pivots of the Lemke-Howson solution algorithm [Lemke and Howson, 1964] for two-agent games, and the two algorithms find precisely the same set of solutions. Thus, the continuation method is a strict generalization of the Lemke-Howson algorithm that allows different perturbation rays and games of more than two agents.

4.4 Iterated Polymatrix Approximation

Because perturbed games may themselves have an exponential number of equilibria, and the path may wind back and forth through any number of them, the continuation algorithm can take a while to trace its way back to a solution to the original game. We can speed up the algorithm using an initialization procedure based on the *iterated polymatrix approximation* (IPA) algorithm of GW. A *polymatrix game* is a normal-form game where the payoffs to a agent n are equal to the sum of the payoffs from a set of two-agent games, each involving n and another agent. Polymatrix games can be solved quickly using the Lemke-Howson algorithm [1964].

Given a normal-form game G and a strategy profile σ , we can construct a polymatrix game P_σ whose Jacobian at σ is the same as the Jacobian of G ’s payoff function V at σ . The game P_σ is a linearized approximation to G around σ , and can be computed efficiently from the Jacobian of G . GW provide an iterative algorithm that, in each step, takes a profile σ and improves it using the solution of the polymatrix approximation P_σ . This algorithm is not guaranteed to converge, but in practice, it quickly moves “near” a good solution. We then construct a perturbed game close to the original game for which this approximate equilibrium is an exact equilibrium. The continuation method is then run from this starting point to find an exact equilibrium of the original game.

5 Exploiting Structure

As mentioned above, the calculation of ∇V at each step of the algorithm consumes most of the time. Both in normal-form and (in the worst case) in extensive-form games, it requires time that is exponential in the number of agents. However, as we show in this section, when using a structured representation such as a graphical game or a MAID, we can effectively exploit the structure of the game to drastically reduce the computational time required.

5.1 Graphical Games

Consider the computation of the normal-form Jacobian in Eq. (4). The key insight is that the choice of strategy for an agent outside the family of n_1 does not affect G_{n_1} . This observation allows us to compute the n_1 entries in the Jacobian

locally, considering only n_1 ’s family. More precisely, we can consider two cases. If $n_2 \notin \text{Fam}_{n_1}$, then

$$\nabla V_{s_1, s_2}(\sigma) = \sum_{t \in S_{-n_1}} G_{n_1}(s_1, t) \prod_{i \in \text{Fam}_{n_1} - \{n_1\}} \sigma_{t_i}. \quad (6)$$

Recalling that S_{-n_1} is a vector of actions only of $\text{Fam}_{n_1} - \{n_1\}$, we see that this computation is exponential only in the family size of n_1 . Furthermore, this value does not depend on s_2 or n_2 , and can therefore be calculated once and copied for all agents $n_2 \notin \text{Fam}_{n_1}$ and for all choices of their strategies. If $n_2 \in \text{Fam}_{n_1}$, then we simply have:

$$\nabla V_{s_1, s_2}(\sigma) = \sum_{t \in S_{-n_1, -n_2}} G_{n_1}(s_1, s_2, t) \prod_{i \in \text{Fam}_{n_1} - \{n_1, n_2\}} \sigma_{t_i}. \quad (7)$$

Letting f be the maximal family size, and d the maximal number of actions per agent, we have that the computation of the Jacobian requires time $O(|N|fd^f + |N|^2)$.

5.2 MAIDs

The Jacobian for MAIDs

To find equilibria in MAIDs, we extend the sequence form continuation method of Section 4.2. As above, our key task is computing the Jacobian of Eq. (5). The Jacobian has an entry for each pair of sequences in the game (one for each agent). We therefore begin by noting that the sequence form representation for MAIDs with perfect recall is no larger than the agent’s decision rules for that MAID. Due to perfect recall, the last decision node (in topological order) must have incoming edges from all of its previous actions and all parents of previous actions. Moreover, it must have an information set for any distinct agent history. Thus, the agent’s decision rule for that final decision has the same size as the sequence form. Hence, the dimension m of the $m \times m$ Jacobian matrix is linear in the size of the MAID (where size, as usual, includes the size of the parameterization).

We next turn to the computation of the Jacobian entries. Eq. (5) can be rewritten as

$$\nabla V_{s_1, s_2}(\sigma) = \sum_{z \in Z_{s_1, s_2}} \frac{G_{n_1}(z) \prod_{i \in N} \sigma_i(z)}{\sigma_{n_1}(z) \sigma_{n_2}(z)}. \quad (8)$$

A leaf node z in the extensive-form game is simply an assignment x to all of the variables in the MAID, and $G_{n_1}(z)$ is n_1 ’s utility given x . The sequence probability $\sigma_n(z)$ is the product of the probabilities for the decisions of agent n in the assignment x . Thus, Eq. (8) is an expectation of $G_{n_1}(z) / [\sigma_{n_1}(z) \sigma_{n_2}(z)]$. The expectation is over the distribution defined by the Bayesian network $\mathcal{B}_\sigma$ whose structure is the same as the MAID, and where the agents’ decision nodes have CPDs determined by σ .

The agent’s utility $G_{n_1}(z)$ is the sum of its utility nodes. Due to linearity of expectation, we can perform the computation separately for each of the agent’s utility nodes, and then simply add up the separate contributions. Thus, we assume from here on, without loss of generality, that n_1 has only a single utility node U .

The value of $\sigma_{n_i}(z)$ ($i = 1, 2$) depends only on the values of the set of nodes $\mathcal{D}_{n_i}$ consisting of n_i ’s decision nodes and

their parents. Thus, instead of computing the probabilities for all assignments to all variables, we need only to compute the marginal joint distribution over U , D_{n_1} , and D_{n_2} . From this distribution, we can compute the expectation in Eq. (5).

Using Bayesian Network Inference

Our analysis above reduces the required computations significantly. We need only compute one joint distribution for every pair of agents n_1, n_2 . This joint distribution is the one defined by the Bayesian network B_σ . Naively, this computation requires that we execute Bayesian network inference $|N|^2$ times: once for each ordered pair of agents n_1, n_2 . Fortunately, we can exploit the structure of the MAID to perform this computation much more efficiently.

The basis for our method is the *clique tree algorithm* of Lauritzen and Spiegelhalter [1998]. A clique tree for a Bayesian network B is a data structure defined over an undirected tree over a set of nodes $\mathcal{C}$. Each node $C_i \in \mathcal{C}$ is a subset of the nodes in B called a *clique*. The clique tree satisfies certain important properties. It must be *family preserving*: for each node X in B , there exists a clique $C_i \in \mathcal{C}$ such that X and its parents are a subset of C_i . It also satisfies a *separation* requirement: if C_2 blocks the path from C_1 to C_3 , then, in the distribution defined by B , we have that the variables in C_1 are conditionally independent of those in C_3 given those in C_2 .

Each clique maintains a data structure, called a *potential*, which is an unnormalized distribution over the variables in C_i . The size of the potential for C_i is therefore exponential in $|C_i|$. The clique tree inference algorithm proceeds by passing messages from one clique to another in the tree. The messages are used to update the potential in the receiving clique. After a process in which messages have been sent in both directions over each edge in the tree, the tree is said to be *calibrated*; at this point, the potential of every clique C_i contains precisely the joint distribution over the variables in C_i according to B .

We can use the clique tree algorithm to perform inference over B_σ . Now, consider the final decision node for agent n_i . Due to the perfect recall assumption, all of n_i 's previous decisions and all of their parents are also parents of this decision node. The family preservation property therefore implies that D_{n_i} is fully contained in some clique. Thus, the expectation of Eq. (8) requires the computation of the joint distribution over three cliques in the tree: the one containing U , the one containing D_{n_1} , and the one containing D_{n_2} . We need to compute this joint distribution for every pair of agents n_1, n_2 .

The first key insight is that we can reduce this problem to one of computing the joint marginal distribution for all pairs of cliques in the tree. Assume we have computed $P_B(C_i, C_j)$ for every pair of cliques C_i, C_j . Now, consider any triple of cliques C_1, C_2, C_3 . There are two cases: either one of these cliques is on the path between the other two, or not. In the first case, assume without loss of generality that C_2 is on the path from C_1 to C_3 . In this case, by the separation requirement, we have that $P_B(C_1, C_2, C_3) = P_B(C_1, C_2)P_B(C_2, C_3)/P_B(C_2)$. In the second case, there exists a unique clique C^* which blocks the paths between any pair of these cliques. Again, by the separation property, C^* renders these cliques conditionally independent, so we can use a similar method to compute $P_B(C_1, C_2, C_3)$.

Thus, we have reduced the problem to one of computing the marginals over all pairs of cliques in a calibrated clique-tree. We can use dynamic programming to execute this process efficiently. We construct a table that contains $P_B(C_i, C_j)$ for each pair of cliques C_i, C_j . We construct the table in order of length of the path from C_i to C_j . The base case is when C_i and C_j are adjacent in the tree. In this case, we have that $P_B(C_i, C_j) = P_B(C_i)P_B(C_j)/P_B(C_i \cap C_j)$. The probability expressions in the numerator are simply the clique potentials in the calibrated tree. The denominator can be obtained by marginalizing either of the two cliques. For cliques C_i and C_j that are not adjacent, we let C_k be the node adjacent to C_j on the path from C_i to C_j . The clique C_k is one step closer to C_i , so, by construction, we have already computed $P(C_i, C_k)$. We can now apply the separation property again:

$$P_B(C_i, C_j) = \sum_{C_i - C_k} \frac{P_B(C_i, C_k)P_B(C_k, C_j)}{P_B(C_k)}. \quad (9)$$

Let ℓ be the number of cliques in the tree, and d be size of the largest clique (the number of entries in its potential). The cost of calibrating the clique tree for B_σ is $O(\ell d)$. The cost of computing Eq. (9) for all pairs of cliques is $O(\ell^2 d^3)$. Finally, the cost of computing the s_i, s_j entry of the Jacobian is $O(d^4)$. In games where interactions between the agents are highly structured, the size d of the largest clique can be a constant even as the number of agents grows. In this case, the complexity grows only quadratically in the number of cliques, and hence also in the number of agents.

6 Results

6.1 Graphical Games

We compared two versions of our algorithm: **cont**, the simple continuation method, and **IPA+cont**, the continuation method with the IPA initialization. We compared our results to the published results of the the algorithm of Vickrey and Koller [2002] (VK hereafter). The VK method only returns ϵ -equilibria, but their approach is the only one that applies to graphical games whose interaction structure is not a (bi-connected) tree and for which timing results are available. The VK paper contains several different algorithms. We compared against the algorithm which had the smallest approximation error for a given problem.

Following VK, our algorithms were run on two classes of games, of varying size. The Road game, denoting a situation where agents must build in land plots along a road, is played on a 2-by- L grid; each agent has three actions, and its payoffs depend only on the actions of its (grid) neighbors. Following VK, we constructed a game where the payoff for an agent is simply the sum of payoffs of games played separately with its neighbors, and where each such subgame has the payoff structure of rock-paper-scissors. This game is, in fact, a polymatrix game, and hence is very easy to solve using our methods. We also experimented with a ring graph with three actions per agent and random payoffs.

For each class of games, we chose a set of game sizes to run on. For each, we selected, randomly in cases where the payoffs were random, a set of at least ten (and up to one hundred for MAIDs) test games to solve. We then solved each game

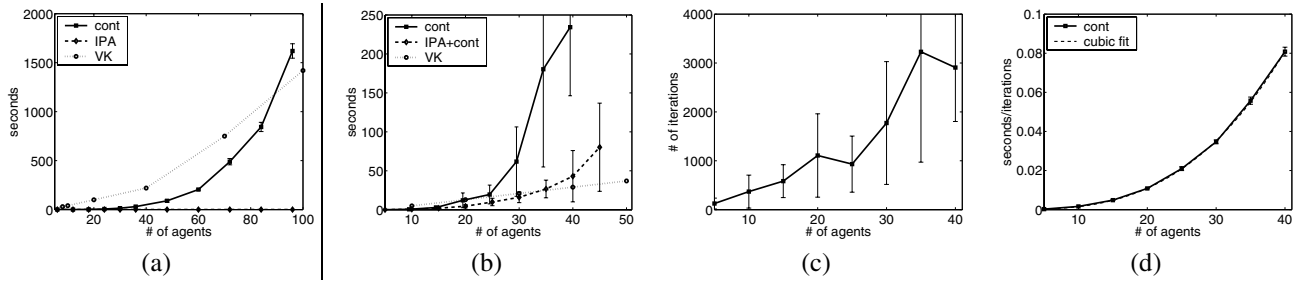


Figure 2: Results for graphical games: (a) Running time for road game with rock-paper-scissors payoffs. Results for ring game with random payoffs: (b) running time; (c) number of iterations of cont; (d) average time per iteration of cont.

with a different random perturbation vector, $\mathbf{b}$, and recorded the time and number of iterations necessary to reach the first equilibrium. We then averaged over test cases. The error bars show the variance due to the choice of perturbation vector and, for random games, the choice of game. For smaller games, the algorithms always converged to an equilibrium. In about 40% of the larger games (more than 20 agents), the algorithms did not converge on the first trial; in these cases, we restarted the same game with a different random perturbation vector. On average, about 2 restarts were sufficient for these difficult games. In a few large graphical games (e.g. 9% of games with 45 agents), IPA did not converge after 10 restarts; in these cases we did not record results for IPA+cont. In the other restarted cases, we recorded the time for the converging run. Our results are shown in Figures 2(a,b).

In all cases, our algorithm found an equilibrium with error at most 10^{-12} , essentially machine precision. In the Road games, we compared against the times for VK using their hill climbing method which found ϵ -equilibria with error of 10^{-4} . In these games, the **cont** method is more efficient for smaller games, but then becomes more costly. Due to the polymatrix nature of this game, the IPA+cont solves it immediately with the Lemke-Howson algorithm, and is therefore significantly less expensive than VK.

In the random-payoff ring games, VK had an equilibrium error of about 0.01 using their cost minimization method with a grid discretization of $1/3$. Here, our algorithms are more efficient than VK for smaller games (up to 20–30 agents), with IPA+cont performing considerably better than **cont**. However, the running time of our algorithms grows more rapidly than that of VK, so that for larger games, they become impractical. Nevertheless, our algorithms performed well in games with up to 50 agents and 3 actions per agent, games which were previously intractable for exact algorithms.

Here, we also plotted the number of iterations and time per iteration for **cont** in Figures 2(c,d). The number of iterations varies based both on the game and perturbation ray chosen. However, the time per iteration is almost exactly cubic as predicted. We note that, when IPA is used, the continuation method converges almost immediately (within a second).

6.2 MAIDs

Koller and Milch [2001] define a *relevance graph* over the decision nodes in a MAID, where there is an edge from D_1 to D_2 if the decision rule at D_1 impacts the choice of decision rule at D_2 . They show that the task of finding equilibria for a MAID can be decomposed, in that only decision nodes in the

same strongly connected component in the relevance graph must be considered together. However, their approach is unable to deal with structure within a strongly connected component, and they resorted to converting the game to extensive form, and using a standard equilibrium solver. Our approach addresses the complementary problem, dealing specifically with this finer-grained structure. Thus, we focused our experiments on MAIDs with cyclic relevance graphs.

We ran our algorithms on two classes of games, with varying sizes. The first, a simple chain, alternates between decision and chance nodes with each decision node belonging to a different agent. Each agent has two utility nodes, each connected to its decision node and to a neighbor's (except for the end agents who have one utility node for their single neighbor). All probability tables and payoff matrices are random. Our second example is shown in Figure 1(b). It is an extension of the graphical road game from above. Each agent must submit plans simultaneously (P_1 – P_n) for the type of building (home or store) that they will build along a road. However, building (B_1 – B_n) proceeds from left to right and so before committing to a build, an agent can see a noisy estimate of the plans of the agent to its left (E_1 – E_n). Agents would prefer to be the first one to start a new type of building (i.e., be different than their left neighbors, but the same as their right neighbors). They also take a penalty if their building and plan decisions differ. Carefully chosen payoffs ensure non-trivial mixed strategies.

Figures 3(a,b) show the running times for computing an equilibrium as the number of agents is increased for both types of games. We compared our results to those achieved by converting the game to extensive-form and running Gambit, a standard equilibrium computation package. Our timing results for Gambit do not include the time for the conversion to extensive-form.

Figures 3(c,d) show the number of iterations and running time per iteration for the case of the two-stage road game. The running time per iteration is once again well fit by a cubic. The variance is mainly due to the execution of the retraction operator whose running time depends on the number of strategies in the support.

7 Discussion and Conclusions

In the last few years, several papers have addressed the issue of finding equilibria in structured games. For graphical games, the exact algorithms proposed so far apply only to the very restricted class of games where the interaction structure

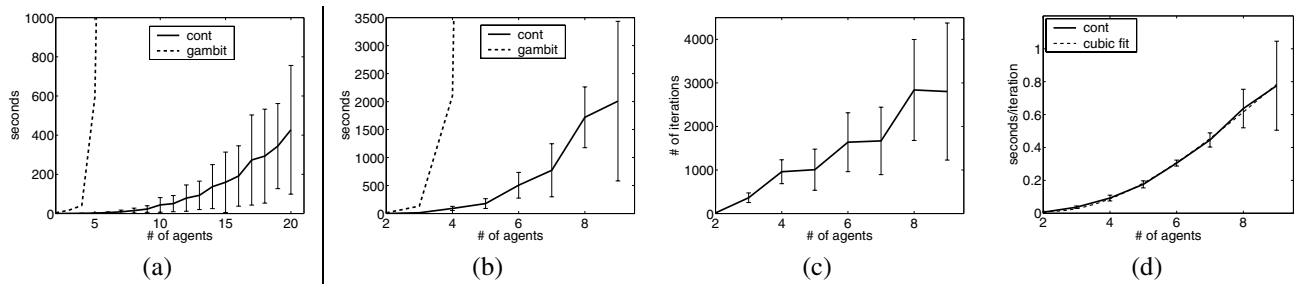


Figure 3: Results for MAIDs: (a) Running times for the chain MAID. Results for two-stage Road MAID: (b) running time; (c) number of iterations; (d) time per iteration.

is an undirected tree, and where each has only two possible actions [Kearns *et al.*, 2001; Littman *et al.*, 2002].

There have been several algorithms proposed for the computation of ϵ -equilibria in general graphical games, most of which (implicitly or explicitly) define an equilibrium as a set of constraints over a discretized space of mixed strategies, and then use some constraint solving method: Kearns *et al.* [2001] use a tree-propagation algorithm; Vickrey and Koller [2002] use variable elimination methods (VK1); and Ortiz and Kearns [2003] use arc-consistency constraint propagation followed by search. Vickrey and Koller [2002] also propose a gradient ascent algorithm (VK2). The running times of KLS and VK1 both depend on the tree-width of the graph, whereas the running times of our algorithm, VK2, and OK depend on the degree of the graph. However, these latter three algorithms all require multiple iterations and no bounds are currently known on the number of iterations required.

For MAIDs, Koller and Milch [2001] (KM) define a notion of independence between agents' decision, and provide an algorithm that can decompose the problem based on fairly coarse independence structure. Our algorithm is able to exploit a much finer-grained structure, resolving an open problem left by KM. La Mura [2000] (LM) proposes a continuation method for finding one or all equilibria in a G net, a representation which is very similar to MAIDs. This proposal only exploits a very limited set of structural properties (a strict subset of KM and of our algorithm). The proposal was also never implemented, and several issues regarding non-converging paths seem unresolved.

We have presented an algorithm for computing exact equilibria in structured games. Our algorithm is based on the methods of GW, but shows how the key computational steps in their approach can be performed much more efficiently by exploiting the game structure. Our method allows us to provide the first exact algorithm for general graphical games, and the first algorithm that takes full advantage of the independence structure of a MAID. Our methods can find exact equilibria in games with large numbers of agents, games which were previously intractable for exact methods.

Acknowledgments. This work was supported by ONR MURI Grant N00014-00-1-0637, and by Air Force contract F30602-00-2-0598 under DARPA's TASK program.

References

[Govindan and Wilson, 2002] S. Govindan and R. Wilson. Structure theorems for game trees. *Proc. Natl Academy of Sciences*, 99(13):9077–9080, 2002.

- [Govindan and Wilson, 2003a] S. Govindan and R. Wilson. Computing Nash equilibria by iterated polymatrix approximation. *J. Economic Dynamics and Control*, 2003. to appear.
- [Govindan and Wilson, 2003b] S. Govindan and R. Wilson. A global Newton method to compute Nash equilibria. *J. Economic Theory*, 2003. to appear.
- [Howard and Matheson, 1984] R. A. Howard and J. E. Matheson. Influence diagrams. In *Readings on the Principles and Applications of Decision Analysis*, volume 2, pages 719–762. Strategic Decision Group, 1984. article dated 1981.
- [Kearns *et al.*, 2001] M. Kearns, M. L. Littman, and S. Singh. Graphical models for game theory. In *Proc. UAI*, 2001.
- [Kohlberg and Mertens, 1986] E. Kohlberg and J.-F. Mertens. On the strategic stability of equilibria. *Econometrica*, 54(5):1003–1038, September 1986.
- [Koller and Megiddo, 1992] D. Koller and N. Megiddo. The complexity of two-person zero-sum games in extensive form. *Games and Economic Behavior*, 4:528–552, 1992.
- [Koller and Milch, 2001] D. Koller and B. Milch. Multi-agent influence diagrams for representing and solving games. In *Proc. IJCAI*, pages 1027–1034, 2001.
- [La Mura, 2000] P. La Mura. Game networks. In *Proc. UAI*, pages 335–342, 2000.
- [Lauritzen and Spiegelhalter, 1998] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *J. Royal Statistical Society*, B 50(2):157–224, 1998.
- [Lemke and Howson, 1964] C. E. Lemke and J. T. Howson, Jr. Equilibrium points in bimatrix games. *J. Society of Applied Mathematics*, 12(2):413–423, June 1964.
- [Littman *et al.*, 2002] M. L. Littman, M. Kearns, and S. Singh. An efficient exact algorithm for singly connected graphical games. In *NIPS-14*, volume 2, pages 817–823, 2002.
- [McKelvey and McLennan, 1996] R. D. McKelvey and A. McLennan. Computation of equilibria in finite games. In *Handbook of Computational Economics*, vol. 1, pages 87–142. Elsevier, 1996.
- [Nash, 1951] J. Nash. Non-cooperative games. *The Annals of Mathematics*, 52(2):286–295, September 1951.
- [Ortiz and Kearns, 2003] L. E. Ortiz and M. Kearns. Nash propagation for loopy graphical games. In *NIPS-15*, 2003. to appear.
- [Owen, 1995] G. Owen. *Game Theory*. Academic Press, UK, 1995.
- [Vickrey and Koller, 2002] D. Vickrey and D. Koller. Multi-agent algorithms for solving graphical games. In *Proc. AAAI*, 2002.
- [von Stengel, 1996] B. von Stengel. Efficient computation of behavior strategies. *Games and Economic Behavior*, 14, 1996.
- [Watson, 2000] L. T. Watson. Theory of globally convergent probability-one homotopies for nonlinear programming. *SIAM J. on Optimization*, 11(3):761–780, 2000.

Complexity Results about Nash Equilibria*

Vincent Conitzer and Tuomas Sandholm

Carnegie Mellon University
Computer Science Department
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
{conitzer,sandholm}@cs.cmu.edu

Abstract

Noncooperative game theory provides a normative framework for analyzing strategic interactions. However, for the toolbox to be operational, the solutions it defines will have to be *computed*. In this paper, we provide a single reduction that 1) demonstrates $\mathcal{NP}$ -hardness of determining whether Nash equilibria with certain natural properties exist, and 2) demonstrates the $\#\mathcal{P}$ -hardness of counting Nash equilibria (or connected sets of Nash equilibria). We also show that 3) determining whether a pure-strategy Bayes-Nash equilibrium exists is $\mathcal{NP}$ -hard, and that 4) determining whether a pure-strategy Nash equilibrium exists in a stochastic (Markov) game is $\mathcal{PSPACE}$ -hard even if the game is invisible (this remains $\mathcal{NP}$ -hard if the game is finite). All of our hardness results hold even if there are only two players and the game is symmetric.

1 Introduction

Noncooperative game theory provides a normative framework for analyzing strategic interactions of agents. However, for the toolbox to be operational, the solutions it defines will have to be *computed* [22]. There has been growing interest in the computational complexity of natural questions in game theory. Starting at least as early as the 1970s, complexity theorists have focused on the complexity of playing particular highly structured games (usually board games, such as chess or Go [10], but also games such as Geography or QSAT [23]). These games tend to be alternating-move zero-sum games with enormous state spaces, which can nevertheless be concisely represented due to the simple rules governing the transition between states. As a result, effort on finding results for general classes of games has often focused on complex languages in which such structured games can be concisely represented.

Real-world strategic settings are generally not nearly as structured, nor do they generally possess the other properties (most notably, zero-sumness) of board games and the

like. Algorithms for analyzing this more general class of games strategically are a necessary component of sophisticated agents that are to play such games. Additionally, they are needed by *mechanism designers* who have (some) control over the rules of the game and would like the outcome of the game to have certain properties, such as maximum social welfare.

Noncooperative game theory provides languages for representing large classes of strategic settings, as well as sophisticated notions of what it means to “solve” such games. The best known solution concept is that of *Nash equilibrium* [16], where the players’ strategies are such that no individual player can derive any benefit from deviating from its strategy. The question of how complex it is to construct such an equilibrium has been dubbed “a most fundamental computational problem whose complexity is wide open” and “together with factoring, [...] the most important concrete open question on the boundary of $\mathcal{P}$ today” [19].

While this question remains open, important concrete advances have been made in determining the complexity of related questions. For example, 2-person zero-sum games can be solved using linear programming [12] in polynomial time. As another example, determining the existence of a joint strategy where each player gets expected payoff at least k is $\mathcal{NP}$ -complete in a concisely representable extensive form game where both players receive the same utility [2].¹ As yet another example, in 2-player general-sum normal form games, determining the existence of Nash equilibria *with certain properties* is $\mathcal{NP}$ -hard [4]. Finally, the complexity of best-responding, of guaranteeing payoffs, and of finding an equilibrium in repeated and sequential games has been studied in [1, 7, 11, 18, 25].

In this paper we provide new complexity results on questions related to Nash equilibria. In Section 2 we provide a single reduction which significantly improves on many of Gilboa and Zemel’s results on determining the existence of Nash equilibria with certain properties. In Section 3, we use the same reduction to show that counting the number of Nash equilibria (or connected sets of Nash equilibria) is $\#\mathcal{P}$ -hard. In Section 4 we show that determining whether a

*The material in this paper is based upon work supported by the National Science Foundation under CAREER Award IRI-9703122, Grant IIS-9800994, ITR IIS-0081246, and ITR IIS-0121678.

¹This game can be converted to a normal form game as well, but it will grow exponentially in size, and the hardness result does not go through.

pure-strategy Bayes-Nash equilibrium exists is $\mathcal{NP}$ -hard. Finally, in Section 5 we show that determining whether a pure-strategy Nash equilibrium exists in a stochastic (Markov) game is $\mathcal{PSPACE}$ -hard even if the game is invisible (this remains $\mathcal{NP}$ -hard if the game is finite). All of our hardness results hold even if there are only two players and the game is symmetric.

2 Equilibria with certain properties in normal form games

When one analyzes the strategic structure of a game, especially from the viewpoint of a mechanism designer who tries to construct good rules for a game, finding a single equilibrium is far from satisfactory. More desirable equilibria may exist: in this case the game becomes more attractive, especially if one can coax the players into playing a desirable equilibrium. Also, less desirable equilibria may exist: in this case the game becomes less attractive. Before we can make a definite judgment about the quality of the game, we would like to know the answers to questions such as: What is the game's most desirable equilibrium? Is there a unique equilibrium? If not, how many equilibria are there? Algorithms that tackle these questions would be useful both to players and to the mechanism designer.

Furthermore, algorithms that answer certain existence questions may pave the way to designing algorithms that construct a Nash equilibrium. For example, if we had an algorithm that told us whether there exists any equilibrium where a certain player plays a certain strategy, this could be useful in eliminating possibilities in the search for a Nash equilibrium.

However, all the existence questions that we have investigated turn out to be $\mathcal{NP}$ -hard. These are not the first results of this nature; most notably, Gilboa and Zemel provide some $\mathcal{NP}$ -hardness results in the same spirit [4]. We provide a single reduction which demonstrates (sometimes stronger versions of) most of their hardness results, and interesting new results. Additionally, as we show in Section 3, the reduction shows $\#P$ -hardness of counting the number of equilibria.

We first need some standard definitions from game theory.

Definition 1 In a normal form game, we are given a set of agents A , and for each agent i , a strategy set Σ_i and a utility function $u_i : \Sigma_1 \times \Sigma_2 \times \dots \times \Sigma_{|A|} \rightarrow \mathbb{R}$.

Definition 2 A mixed strategy σ_i for player i is a probability distribution over Σ_i . A special case of a mixed strategy is a pure strategy, where all of the probability mass is on one element of Σ_i .

Definition 3 (Nash [16]) Given a normal form game, a Nash equilibrium (NE) is vector of mixed strategies, one for each agent i , such that no agent has an incentive to deviate from its mixed strategy given that the others do not deviate. That is, for any i and any alternative mixed strategy σ'_i , we have $E[u_i(s_1, s_2, \dots, s_i, \dots, s_{|A|})] \geq E[u_i(s_1, s_2, \dots, s'_i, \dots, s_{|A|})]$, where each s_i is drawn from σ_i , and s'_i from σ'_i .

Now we are ready to present our reduction.

Definition 4 Let ϕ be a Boolean formula in conjunctive normal form. Let V be its set of variables (with $|V| = n$), L the set of corresponding literals (a positive and a negative one for each variable)², and C its set of clauses. The function $v : L \rightarrow V$ gives the variable corresponding to a literal, e.g. $v(x_1) = v(-x_1) = x_1$. We define $G(\phi)$ to be the following symmetric 2-player game in normal form. Let $\Sigma \equiv \Sigma_1 = \Sigma_2 = L \cup V \cup C \cup \{f\}$. Let the utility functions be

- $u_1(l^1, l^2) = u_2(l^2, l^1) = 1$ for all $l^1, l^2 \in L$ with $l^1 \neq -l^2$;
- $u_1(l, -l) = u_2(-l, l) = -2$ for all $l \in L$;
- $u_1(l, x) = u_2(x, l) = -2$ for all $l \in L, x \in \Sigma - L$;
- $u_1(v, l) = u_2(l, v) = 2$ for all $v \in V, l \in L$ with $v(l) \neq v$;
- $u_1(v, l) = u_2(l, v) = 2 - n$ for all $v \in V, l \in L$ with $v(l) = v$;
- $u_1(v, x) = u_2(x, v) = -2$ for all $v \in V, x \in \Sigma - L$;
- $u_1(c, l) = u_2(l, c) = 2$ for all $c \in C, l \in L$ with $l \notin c$;
- $u_1(c, l) = u_2(l, c) = 2 - n$ for all $c \in C, l \in L$ with $l \in c$;
- $u_1(c, x) = u_2(x, c) = -2$ for all $c \in C, x \in \Sigma - L$;
- $u_1(f, f) = u_2(f, f) = 0$;
- $u_1(f, x) = u_2(x, f) = 1$ for all $x \in \Sigma - \{f\}$.

Theorem 1 If $(l_1, l_2, \dots, l_n)$ (where $v(l_i) = x_i$) satisfies ϕ , then there is a Nash equilibrium of $G(\phi)$ where both players play l_i with probability $\frac{1}{n}$, with expected utility 1 for each player. The only other Nash equilibrium is the one where both players play f , and receive expected utility 0 each.

Proof: We first demonstrate that these combinations of mixed strategies indeed do constitute Nash equilibria. If $(l_1, l_2, \dots, l_n)$ (where $v(l_i) = x_i$) satisfies ϕ and the other player plays l_i with probability $\frac{1}{n}$, playing one of these l_i as well gives utility 1. On the other hand, playing the negation of one of these l_i gives utility $\frac{1}{n}(-2) + \frac{n-1}{n}(1) < 1$. Playing some variable v gives utility $\frac{1}{n}(2 - n) + \frac{n-1}{n}(2) = 1$ (since one of the l_i that the other player sometimes plays has $v(l_i) = v$). Playing some clause c gives utility at most $\frac{1}{n}(2 - n) + \frac{n-1}{n}(2) = 1$ (since one of the l_i that the other player sometimes plays occurs in clause c , since the l_i satisfy ϕ). Finally, playing f gives utility 1. It follows that playing any one of the l_i that the other player sometimes plays is an optimal response, and hence that both players playing each of these l_i with probability $\frac{1}{n}$ is a Nash equilibrium. Clearly, both players playing f is also a Nash equilibrium since playing anything else when the other plays f gives utility -2 .

Now we demonstrate that there are no other Nash equilibria. If the other player always plays f , the unique best response is to also play f since playing anything else will give utility -2 . Otherwise, given a mixed strategy for the other player, consider a player's expected utility given that the other player does not play f . (That is, the probability distribution over the other player's strategies is proportional to the probability distribution constituted by that player's mixed strategy,

²Thus, if x_1 is a variable, x_1 and $-x_1$ are literals. We make a distinction between the variable x_1 and the literal x_1 .

except f occurs with probability 0). If this expected utility is less than 1, the player is strictly better off playing f (which gives utility 1 when the other player does not play f , and also performs better than the original strategy when the other player does play f). So this cannot occur in equilibrium.

There are no Nash equilibria where one player always plays f but the other does not, so suppose both players play f with probability less than one. Consider the expected social welfare ($E[u_1 + u_2]$), given that neither player plays f . It is easily verified that there is no outcome with social welfare greater than 2. Also, any outcome in which one player plays an element of V or C has social welfare strictly below 2. It follows that if either player ever plays an element of V or C , the expected social welfare given that neither player plays f is strictly below 2. By linearity of expectation it follows that the expected utility of at least one player is strictly below 1 given that neither player plays f , and by the above reasoning, this player would be strictly better off playing f instead of its randomization over strategies other than f . It follows that no element of V or C is ever played in a Nash equilibrium.

So, we can assume both players only put positive probability on strategies in $L \cup \{f\}$. Then, if the other player puts positive probability on f , playing f is a strictly better response than any element of L (since both give utility 1 if the other player plays an element of L , but f does better if the other player plays f). It follows that the only equilibrium where f is ever played is the one where both players always play f .

Now we can assume that both players only put positive probability on elements of L . Suppose that for some $l \in L$, the probability that a given player plays either l or $\neg l$ is less than $\frac{1}{n}$. Then the expected utility for the other player of playing $v(l)$ is strictly greater than $\frac{1}{n}(2 - n) + \frac{n-1}{n}(2) = 1$, and hence this cannot be a Nash equilibrium. So we can assume that for any $l \in L$, the probability that a given player plays either l or $\neg l$ is precisely $\frac{1}{n}$.

If there is an element of L such that player 1 puts positive probability on it and player 2 on its negation, both players have expected utility less than 1 and would be better off switching to f . So, in a Nash equilibrium, if player 1 plays l with some probability, player 2 must play l with probability $\frac{1}{n}$, and thus player 1 must play l with probability $\frac{1}{n}$. Thus we can assume that for each variable, exactly one of its corresponding literals is played with probability $\frac{1}{n}$ by both players. It follows that in any Nash equilibrium (besides the one where both players play f), literals that are sometimes played indeed correspond to an assignment to the variables.

All that is left to show is that if this assignment does not satisfy ϕ , it does not correspond to a Nash equilibrium. Let $c \in C$ be a clause that is not satisfied by the assignment, that is, none of its literals are ever played. Then playing c would give utility 2, and both players would be better off playing this. ■

Hence, there exists a Nash equilibrium in $G(\phi)$ where each player gets utility 1 if and only if ϕ is satisfiable; otherwise, the only equilibrium is the one where both players play f and each of them gets 0. Since any sensible definition of welfare optimization would prefer the first kind of equilibrium, it follows that determining whether a “good” equilibrium exists is

hard for any such definition. Additionally, the first kind of equilibrium is, in various senses, an optimal outcome for the game, even if the players were to cooperate, so even finding out whether such an optimal equilibrium exists is hard. The following corollaries illustrate these points (each corollary is immediate from Theorem 1).

Corollary 1 *Even in symmetric 2-player games, it is $\mathcal{NP}$ -hard to determine whether there exists a NE with expected (standard) social welfare ($E[\sum_{1 \leq i \leq |A|} u_i]$) at least k , even*

when k is the maximum social welfare that could be obtained in the game.

Corollary 2 *Even in symmetric 2-player games, it is $\mathcal{NP}$ -hard to determine whether there exists a NE where all players have expected utility at least k , even when k is the largest number such that there exists a distribution over outcomes of the game such that all players have expected utility at least k .*

Corollary 3 *Even in symmetric 2-player games, it is $\mathcal{NP}$ -hard to determine whether there exists a Pareto-optimal NE. (A distribution over outcomes is Pareto-optimal if there is no other distribution over outcomes such that every player has at least equal expected utility, and at least one player has strictly greater expected utility).*

Corollary 4 *Even in symmetric 2-player games, it is $\mathcal{NP}$ -hard to determine whether there exists a NE where player 1 has expected utility at least k .*

Some additional interesting corollaries are:

Corollary 5 *Even in symmetric 2-player games, it is $\mathcal{NP}$ -hard to determine whether there is more than one Nash equilibrium.*

Corollary 6 *Even in symmetric 2-player games, it is $\mathcal{NP}$ -hard to determine whether there is an equilibrium where player 1 sometimes plays $x \in \Sigma_1$.*

Corollary 7 *Even in symmetric 2-player games, it is $\mathcal{NP}$ -hard to determine whether there is an equilibrium where player 1 never plays $x \in \Sigma_1$.*

All of these results indicate that it is hard to obtain summary information about a game’s Nash equilibria. (Corollary 5 and weaker³ versions of Corollaries 2, 6 and 7 were first proven by Gilboa and Zemel [4].)

3 Counting the number of equilibria in normal form games

Existence questions do not tell the whole story. In general, we are interested in characterizing all the equilibria of a game. One rather weak such characterization is the number of equilibria⁴. We can use Theorem 1 to show that even determining this number in a given normal form game is hard.

Corollary 8 *Even in symmetric 2-player games, counting the number of Nash equilibria is $\#\mathcal{P}$ -hard.*

³Our results prove hardness in a slightly more restricted setting.

⁴The number of equilibria in normal form games has been studied both in the worst case [15] and in the average case [14].

Proof: The number of Nash equilibria in our game $G(\phi)$ is the number of satisfying assignments to the variables of ϕ , plus one. Counting the number of satisfying assignments to a CNF formula is $\#P$ -hard [24]. ■

It is easy to construct games where there is a continuum of Nash equilibria. In such games, it is more meaningful to ask how many distinct continuums of equilibria there are. More formally, one can ask how many maximal connected sets of equilibria a game has (a maximal connected set is a connected set which is not a proper subset of a connected set).

Corollary 9 *Even in symmetric 2-player games, counting the number of maximal connected sets of Nash equilibria is $\#P$ -hard.*

Proof: Every Nash equilibrium in $G(\phi)$ constitutes a maximal connected set by itself, so the number of maximal connected sets is the number of satisfying assignments to the variables of ϕ , plus one. ■

The most interesting $\#P$ -hardness results are the ones where the corresponding existence and search questions are easy, such as counting the number of perfect bipartite matchings. In the case of Nash equilibria, the existence question is trivial: it has been analytically shown (by Kakutani's fixed point theorem) that a Nash equilibrium always exists [16]. The complexity of the search question remains open.

4 Pure-strategy Bayes-Nash equilibria

Equilibria in pure strategies are particularly desirable because they avoid the uncomfortable requirement that players randomize over strategies among which they are indifferent [3]. In normal form games with small numbers of players, it is easy to determine the existence of pure-strategy equilibria: one can simply check, for each combination of pure strategies, whether it constitutes a Nash equilibrium. However, this is not feasible in *Bayesian* games, where the players have private information about their own preferences (represented by *types*). Here, players may condition their actions on their types, so the strategy space of each player is exponential in the number of types.

In this section, we show that the question of whether a pure-strategy Bayes-Nash equilibrium exists is in fact $\mathcal{NP}$ -hard even in symmetric two-player games. First, we need the standard definition of a Bayesian game and Bayes-Nash equilibrium from game theory.

Definition 5 *In a Bayesian game, we are given a set of agents A ; for each agent i , a set of types Θ_i ; a commonly known prior distribution ϕ over $\Theta_1 \times \Theta_2 \times \dots \times \Theta_{|A|}$; for each agent i , a set of strategies Σ_i ; and for each agent i , a utility function $u_i : \Theta_i \times \Sigma_1 \times \Sigma_2 \times \dots \times \Sigma_{|A|} \rightarrow \mathbb{R}$.*

Definition 6 (Harsanyi [5]) *Given a Bayesian game, a Bayes-Nash equilibrium (BNE) is a vector of mixed strategies, one for each pair $i, \theta_i \in \Theta_i$, such that no agent has an incentive to deviate, for any of its types, given that the others do not deviate. That is, for any $i, \theta_i \in \Theta_i$, and any alternative mixed strategy σ'_{i, θ_i} , we have*

$$E_{\theta_{-i}|\theta_i}[E[u_i(\theta_i, s_{1, \theta_1}, s_{2, \theta_2}, \dots, s_{i, \theta_i}, \dots, s_{|A|, \theta_{|A|}})]]$$

$$\geq E_{\theta_{-i}|\theta_i}[E[u_i(\theta_i, s_{1, \theta_1}, s_{2, \theta_2}, \dots, s'_{i, \theta_i}, \dots, s_{|A|, \theta_{|A|}})]]$$

where each s_{i, θ_i} is drawn from σ_{i, θ_i} , and s'_{i, θ_i} from σ'_{i, θ_i} .

We can now define the computational problem.

Definition 7 (PURE-STRATEGY-BNE) *We are given a Bayesian game. We are asked whether there exists a BNE where all the strategies σ_{i, θ_i} are pure.*

To show our $\mathcal{NP}$ -hardness result, we will reduce from the SET-COVER problem.

Definition 8 (SET-COVER) *We are given a set $S = \{s_1, \dots, s_n\}$, subsets $S_1, S_2, \dots, S_m$ of S with $\bigcup_{1 \leq i \leq m} S_i = S$, and an integer k . We are asked whether there exist $S_{c_1}, S_{c_2}, \dots, S_{c_k}$ such that $\bigcup_{1 \leq i \leq k} S_{c_i} = S$.*

Theorem 2 *PURE-STRATEGY-BNE is $\mathcal{NP}$ -hard, even in symmetric 2-player games where ϕ is uniform.*

Proof: We reduce an arbitrary SET-COVER instance to the following PURE-STRATEGY-BNE instance. Let there be two players, with $\Theta \equiv \Theta_1 = \Theta_2 = \{\theta^1, \dots, \theta^k\}$. ϕ is uniform. Furthermore, $\Sigma \equiv \Sigma_1 = \Sigma_2 = \{S_1, S_2, \dots, S_m, s_1, s_2, \dots, s_n\}$. The utility functions we choose in fact do not depend on the types, so we omit the type argument in their definitions. They are as follows:

- $u_1(S_i, S_j) = u_2(S_j, S_i) = 1$ for all S_i and S_j ;
- $u_1(S_i, s_j) = u_2(s_j, S_i) = 1$ for all S_i and $s_j \notin S_i$;
- $u_1(S_i, s_j) = u_2(s_j, S_i) = 2$ for all S_i and $s_j \in S_i$;
- $u_1(s_i, s_j) = u_2(s_j, s_i) = -3k$ for all s_i and s_j ;
- $u_1(s_j, S_i) = u_2(S_i, s_j) = 3$ for all S_i and $s_j \notin S_i$;
- $u_1(s_j, S_i) = u_2(S_i, s_j) = -3k$ for all S_i and $s_j \in S_i$.

We now show the two instances are equivalent. First suppose there exist $S_{c_1}, S_{c_2}, \dots, S_{c_k}$ such that $\bigcup_{1 \leq i \leq k} S_{c_i} = S$. Suppose both players play as follows: when their type is θ_i , they play S_{c_i} . We claim that this is a BNE. For suppose the other player employs this strategy. Then, because for any s_j , there is at least one S_{c_i} such that $s_j \in S_{c_i}$, we have that the expected utility of playing s_j is at most $\frac{1}{k}(-3k) + \frac{k-1}{k}3 < 0$. It follows that playing any of the S_j (which gives utility 1) is optimal. So there is a pure-strategy BNE.

On the other hand, suppose that there is a pure-strategy BNE. We first observe that in no pure-strategy BNE, both players play some element of S for some type: for if the other player sometimes plays some s_j , the utility of playing some s_i is at most $\frac{1}{k}(-3k) + \frac{k-1}{k}3 < 0$, whereas playing some S_i instead guarantees a utility of at least 1. So there is at least one player who never plays any element of S . Now suppose the other player sometimes plays some s_j . We know there is some S_i such that $s_j \in S_i$. If the former player plays this S_i , this will give it a utility of at least $\frac{1}{k}2 + \frac{k-1}{k}1 = 1 + \frac{1}{k}$. Since it must do at least this well in the equilibrium, and it never plays elements of S , it must sometimes receive utility 2. It follows that there exist S_a and $s_b \in S_a$ such that the former player sometimes plays S_a and the latter sometimes plays s_b . But then, playing s_b gives the latter player a utility of at most $\frac{1}{k}(-3k) + \frac{k-1}{k}3 < 0$, and it would be better off playing some S_i instead. (Contradiction.) It follows that in no pure-strategy BNE, any element of S is ever played.

Now, in our given pure-strategy equilibrium, consider the set of all the S_i that are played by player 1 for some type. Clearly there can be at most k such sets. We claim they cover S . For if they do not cover some element s_j , the expected utility of playing s_j for player 2 is 3 (because player 1 never plays any element of S). But this means that player 2 (who never plays any element of S either) is not playing optimally. (Contradiction.) Hence, there exists a set cover. ■

If one allows for general mixed strategies, a Bayes-Nash equilibrium always exists [3]. However, the question of how efficiently one can be constructed remains open.

5 Pure-strategy Nash equilibria in stochastic (Markov) games

We now shift our attention from single-shot games to games with multiple stages. In each stage, the players get to act and obtain payoffs. There has already been some research into the complexity of playing repeated and sequential games. For example, determining whether a particular automaton is a best response is $\mathcal{NP}$ -complete [1]; it is $\mathcal{NP}$ -complete to compute a best-response automaton when the automata under consideration are bounded [18]; the question of whether a given player with imperfect recall can guarantee itself a given payoff using pure strategies is $\mathcal{NP}$ -complete [7]; and in general, best-responding to an arbitrary strategy can even be noncomputable [25]. In this section, we present, to our knowledge, the first $\mathcal{PSPACE}$ -hardness result on the existence of a pure-strategy equilibrium.

A multi-stage game is typically represented as a *stochastic (Markov) game*, where there is an underlying set of states, and the game shifts between these states from stage to stage [3, 20, 21]. At every stage, each player's payoff depends not only on the players' actions, but also on the state. Furthermore, the probability of transitioning to a given state is determined by the current state and the players' current actions. Hardness results for such games cannot be obtained simply by formulating a known hard game such as generalized Go [10] or QSAT [23] as a Markov game, because such a formulation would have to specify an exponential number of states. Even if the number of states is polynomial, one might suspect hardness because the strategy spaces are extremely rich. However, in this section we show $\mathcal{PSPACE}$ -hardness even in a variant where the strategy spaces are simple (in the sense that the players cannot condition their actions on events in the game).

Definition 9 A stochastic (Markov) game consists of

- A set of players A ;
- A set of states S , among which the game transits;
- For each player i , a set of actions Σ_i that can be played in any state;
- A transition probability function $p : S \times \Sigma_1 \times \dots \times \Sigma_{|A|} \times S \rightarrow [0, 1]$, where $p(s_1, a_1, \dots, a_{|A|}, s_2)$ gives the probability of the game being in state s_2 in the next stage given that the current state of the game is s_1 and the players play actions $a_1, \dots, a_{|A|}$;
- For each player i , a payoff function $u_i : S \times \Sigma_1 \times \dots \times \Sigma_{|A|} \rightarrow \mathbb{R}$, where $u_i(s, a_1, \dots, a_{|A|})$ gives the pay-

off to player i in state s where the players play actions $a_1, \dots, a_{|A|}$;

- A discount factor δ such that the total utility of agent i is $\sum_{k=0}^{\infty} \delta^k u_i(s^k, a_1^k, \dots, a_{|A|}^k)$, where s^k is the state of the game at stage k and the players play actions $a_1^k, \dots, a_{|A|}^k$ in stage k .

In general, a player need not always be aware of the current state of the game, the actions the others played in previous stages, or the payoffs that the player has accumulated. In the extreme case, players never find out any of these and are hence playing blindly. We call such a Markov game *invisible*. It is relatively easy to specify a pure strategy in an invisible Markov game, because there is nothing to condition on. Hence, such a strategy is “simply” an infinite sequence of actions (for player i , a sequence $\{a_i^k\}$, where it plays action a_i^k in stage k , regardless).⁵ In spite of this apparent simplicity of the game, we show that determining whether pure-strategy equilibria exist is extremely hard.

Definition 10 (PURE-STRATEGY-INVISIBLE-MARKOV-NE)

We are given an invisible Markov game. We are asked whether there exists a Nash equilibrium where all the strategies are pure.

We show that this problem is $\mathcal{PSPACE}$ -hard, by reducing from PERIODIC-SAT, which is $\mathcal{PSPACE}$ -complete [17].

Definition 11 (PERIODIC-SAT) We are given a CNF formula $\phi(0)$ over the variables $\{x_1^0 \dots x_n^0\} \cup \{x_1^1 \dots x_n^1\}$. Let $\phi(k)$ be the same formula, except that all the superscripts are incremented by k . We are asked whether there exists a Boolean assignment to the variables $\bigcup_{k=0,1,\dots} \{x_1^k \dots x_n^k\}$ such that $\phi(k)$ is satisfied for every $k = 0, 1, \dots$

Theorem 3 PURE-STRATEGY-INVISIBLE-MARKOV-NE is $\mathcal{PSPACE}$ -hard, even when the game is symmetric, 2-player, and the transition process is deterministic.

Proof: We reduce an arbitrary PERIODIC-SAT instance to the following symmetric 2-player PURE-STRATEGY-INVISIBLE-MARKOV-NE instance. The state space is $S = \{s_i\}_{1 \leq i \leq n} \cup \{t_{i,c}^1\}_{1 < i \leq 2n; c \in C} \cup \{t_{i,c}^2\}_{1 < i \leq 2n; c \in C} \cup \{r\}$, where C is the set of clauses in $\phi(0)$. Furthermore, $\Sigma \equiv \Sigma_1 = \Sigma_2 = \{t, f\} \cup C$. The transition probabilities are

- $p(s_i, x^1, x^2, s_{i+1(\text{mod } n)}) = 1$ for $1 < i \leq n$ and all $x^1, x^2 \in \Sigma$;
- $p(s_1, b^1, b^2, s_2) = 1$ for all $b^1, b^2 \in \{t, f\}$;
- $p(s_1, c, b, t_{2,c}^1) = 1$ for all $b \in \{t, f\}$ and $c \in C$;
- $p(s_1, b, c, t_{2,c}^2) = 1$ for all $b \in \{t, f\}$ and $c \in C$;
- $p(s_1, c^1, c^2, r) = 1$ for all $c^1, c^2 \in C$;
- $p(t_{i,c}^j, x^1, x^2, t_{i+1,c}^j) = 1$ for all $1 < i < 2n, j \in \{1, 2\}, c \in C$, and $x^1, x^2 \in \Sigma$;
- $p(t_{2n,c}^j, x^1, x^2, r) = 1$ for all $j \in \{1, 2\}, c \in C$, and $x^1, x^2 \in \Sigma$;

⁵We do not need to worry about issues of credible threats and subgame perfection in this setting, so we can simply use Nash equilibrium as our solution concept [13].

- $p(r, x^1, x^2, r) = 1$ for all $x^1, x^2 \in \Sigma$.

Some of the utilities obtained in a given stage are as follows (we do not specify utilities irrelevant to our analysis):

- $u_1(s_i, x^1, x^2) = u_2(s_i, x^2, x^1) = 0$ for $1 < i \leq n$ and all $x^1, x^2 \in \Sigma$;
- $u_1(s_1, b^1, b^2) = u_2(s_1, b^2, b^1) = 0$ for all $b^1, b^2 \in \{t, f\}$;
- $u_1(s_1, c, b) = u_2(s_1, b, c) = 1$ for all $b \in \{t, f\}$ and $c \in C$, when setting variable x_1^0 to b does not satisfy c ;
- $u_1(s_1, c, b) = u_2(s_1, b, c) = -1$ for all $b \in \{t, f\}$ and $c \in C$, when setting variable x_1^0 to b does satisfy c ;
- $u_1(s_1, c^1, c^2) = u_2(s_1, c^2, c^1) = -1$ for all $c^1, c^2 \in C$;
- $u_1(t_{kn+i,c}^1, x, b) = u_2(t_{kn+i,c}^2, b, x) = 0$ for $k \in \{0, 1\}$, $1 \leq i \leq n$, all $c \in C$ and $b \in \{t, f\}$ such that setting variable x_i^k to b does not satisfy c , and all $x \in \Sigma$;
- $u_1(t_{kn+i,c}^1, x, b) = u_2(t_{kn+i,c}^2, b, x) = -4$ for $k \in \{0, 1\}$, $1 \leq i \leq n$, all $c \in C$ and $b \in \{t, f\}$ such that setting variable x_i^k to b does satisfy c , and all $x \in \Sigma$;
- $u_1(t_{kn+i,c}^1, x, c') = u_2(t_{kn+i,c}^2, c', x) = 0$ for $k \in \{0, 1\}$, $1 \leq i \leq n$, all $c, c' \in C$, and all $x \in \Sigma$.

Additionally, the game played in state r is some symmetric zero-sum game without a pure-strategy equilibrium (for example, a generalization of rock-paper-scissors) with very small payoffs. Finally, the discount factor is $\delta = (\frac{1}{2})^{\frac{1}{2n+1}}$ (so that $\delta^{2n} > \frac{1}{2}$).

We start our analysis with a few observations. First, there can be no pure-strategy equilibrium in which state r is reached at some point, because (since r is an absorbing state) this would require that some pure-strategy equilibrium of the game in state r were played whenever state r occurred. (Otherwise a player who is not best-responding in one of these stages could simply switch to a best response in this stage, and because the game is invisible, the rest of the game would remain unaffected, so this would give higher utility.) But such an equilibrium does not exist. Second, if we ever reach one of the $t_{i,c}^j$ states, we will inevitably reach state r at some point after this. It follows that all pure-strategy Nash equilibria never leave the s_i states.

Now suppose an assignment satisfying the periodic SAT formula exists. Let both players play as follows: in stage $kn + i$ (with $1 \leq i \leq n$), $b \in \{t, f\}$ is played, where b is the value that the variable x_i^k is set to. Clearly, both players receive utility 0 with these strategies. Does either player have an incentive to deviate? The only deviation of any significance is to play some $c \in C$ when the current state is s_1 . So, without loss of generality (because of the symmetry of the game), say player 2 deviates to playing $c \in C$ in stage $kn + 1$ (when the state is s_1). We know that in the satisfying assignment, some variable x_i^l among $x_1^k, \dots, x_n^k, x_1^{k+1}, \dots, x_n^{k+1}$ is set to some b such that setting x_i^{l-k} to b satisfies c . If it is x_1^k , which is set to b , then in stage $kn + 1$ player 1 plays b , and player 2 gets payoff -1 in this stage since we are in state s_1 and setting x_1^0 to b satisfies c . Otherwise, if it is x_i^l with $l = k + 1$ or $i \neq 1$, which is set to b , then player 2 will get payoff 1 in stage $kn + 1$, but in stage $ln + i$ player 1 plays b , and player 2 gets payoff -4 in this stage since we are in state

$t_{(l-k)n+i,c}^2$ and setting x_i^{l-k} to b satisfies c . The discounting is insignificant enough that this more than cancels out the 1 earned in stage $kn + 1$. Player 2 will get (at most) 0 in the other stages up to the first stage in state r , and given that we made the payoffs in the game in state r sufficiently small relative to δ , player 2 will not earn enough in the remaining stages to cancel out its losses so far. So there is no incentive to deviate. Thus, a pure-strategy NE exists.

On the other hand, suppose that no assignment satisfying the periodic SAT formula exists. Let us investigate whether a Nash equilibrium could exist. We know that in such a Nash equilibrium we never leave the s_i , so both players receive utility 0, and no c is ever played in a stage with state s_1 . Since playing a c in one of the other stages can have no deterrent value, we may suppose that only elements of $\{t, f\}$ are played. Now consider the following assignment to the x_i^k : if player 1 plays b in stage $kn + i$, x_i^k is set to b . Since no assignment satisfying the periodic SAT formula exists, we know there is some clause c and some k such that no variable x_i^l among $x_1^k, \dots, x_n^k, x_1^{k+1}, \dots, x_n^{k+1}$ is set to some b such that setting x_i^{l-k} to b satisfies c . But then, if player 2 deviates to play this c in stage $kn + 1$, it will receive payoff 1 in this stage, and payoff 0 in all the remaining stages up to the first stage in state r . Furthermore, player 2 can guarantee itself at least payoff 0 in each stage in state r , as this state corresponds to a zero-sum symmetric game. It follows that this deviation gives player 2 positive utility and is hence beneficial. Thus, no pure-strategy NE exists. ■

A simpler version of the same argument shows a weaker form of hardness for the case where the game is restricted to have only finitely many stages (we omit the proof due to limited space):

Theorem 4 *PURE-STRATEGY-INVISIBLE-MARKOV-NE is $\mathcal{NP}$ -hard, even when the game is symmetric, 2-player, the transition process is deterministic, and the number of stages in the game is finite.*

6 Conclusions and future research

Noncooperative game theory provides a normative framework for analyzing strategic interactions. However, for the toolbox to be operational, the solutions it defines will have to be *computed*. In this paper, we provided a single reduction that 1) demonstrates $\mathcal{NP}$ -hardness of determining whether Nash equilibria with certain natural properties exist, and 2) demonstrates the $\#\mathcal{P}$ -hardness of counting Nash equilibria (or connected sets of Nash equilibria). We also showed that 3) determining whether a pure-strategy Bayes-Nash equilibrium exists is $\mathcal{NP}$ -hard, and that 4) determining whether a pure-strategy Nash equilibrium exists in a stochastic (Markov) game is $\mathcal{PSPACE}$ -hard even in invisible games (and $\mathcal{NP}$ -hard if the game is finite). All of our hardness results hold even if there are only two players and the game is symmetric.

There are numerous open research questions in computing solutions to noncooperative games. Some recent work has focused on novel knowledge representations which, in certain settings, can drastically speed up equilibrium finding (e.g. [6, 8, 9]). One avenue of future work includes identifying re-

stricted classes of games for which equilibria (or equilibria with certain properties) can be found fast. Another avenue involves studying the complexity of characterizing (some of) the equilibria of a game *partially*. Yet another avenue includes analyzing the computational complexity of other solution concepts from noncooperative game theory.

References

- [1] E. Ben-Porath. The complexity of computing a best response automaton in repeated games with mixed strategies. *Games and Economic Behavior*, 1-12, 1990.
- [2] Francis Chu and Joseph Halpern. On the NP-completeness of finding an optimal strategy in games with common payoffs. *Internat. Journal of Game Theory*, 2001.
- [3] Drew Fudenberg and Jean Tirole. *Game Theory*. MIT Press, 1991.
- [4] I. Gilboa and E. Zemel. Nash and correlated equilibria: Some complexity considerations. *Games and Economic Behavior*, 1989.
- [5] John Harsanyi. Game with incomplete information played by Bayesian players. *Management Science*, 14:159–182; 320–334; 486–502, 1967–68.
- [6] Michael Kearns, Michael Littman, and Satinder Singh. Graphical models for game theory. *UAI*, 2001.
- [7] D. Koller and N. Megiddo. The complexity of two-person zero-sum games in extensive form. *Games and Economic Behavior*, pages 4:528–552, 1992.
- [8] Daphne Koller, Nimrod Megiddo, and Bernhard von Stengel. Efficient computation of equilibria for extensive two-person games. *Games and Economic Behavior*, 14(2):247–259, 1996.
- [9] Daphne Koller and B Milch. Multi-agent influence diagrams for representing and solving games. *IJCAI*, pages 1027–1034, Seattle, WA, 2001.
- [10] D Lichtenstein and M Sipser. GO is polynomial-space hard. *Journal of the ACM*, 27:393–401, 1980.
- [11] Michael Littman and Peter Stone. A polynomial-time nash equilibrium algorithm for repeated games. *ACM-EC*, San Diego, CA, 2003.
- [12] R D Luce and H Raiffa. *Games and Decisions*. John Wiley and Sons, 1957.
- [13] Andreu Mas-Colell, Michael Whinston, and Jerry R. Green. *Microeconomic Theory*. Oxford University Press, 1995.
- [14] Andrew McLennan. The expected number of Nash equilibria of a normal form game. *Econometrica*, 1999.
- [15] Andrew McLennan and In-Uck Park. Generic 4x4 two person games have at most 15 Nash equilibria. *Games and Economic Behavior*, pages 26–1,111–130, 1999.
- [16] John Nash. Equilibrium points in n-person games. *Proc. of the National Academy of Sciences*, 36:48–49, 1950.
- [17] J Orlin. The complexity of dynamic languages and dynamic optimization problems. In *STOC*, pages 218–227, 1981.
- [18] C. H. Papadimitriou. On players with a bounded number of states. *Games and Economic Behavior*, pages 122–131, 1992.
- [19] Christos Papadimitriou. Algorithms, games and the Internet. *Annual ACM Symposium on the Theory of Computing (STOC)*, pages 749–253, 2001.
- [20] Lloyd S Shapley. Stochastic games. *Proc. of the National Academy of Sciences*, 39:1095–1100, 1953.
- [21] M Sobel. Noncooperative stochastic games. *Annals of Mathematical Statistics*, 42:1930–1935, 1971.
- [22] Bernhard von Stengel. Computing equilibria for two-person games. In R Aumann and S Hart, eds, *Handbook of game theory*, vol 3. North Holland. Forthcoming.
- [23] L Stockmeyer and A Meyer. Word problems requiring exponential time. In *STOC*, pages 1–9, 1973.
- [24] Leslie Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.
- [25] William R. Zame and John H. Nachbar. Non-computable strategies and discounted repeated games. *Economic Theory*, pages (1)103–122, 1996.

Local-Effect Games

Kevin Leyton-Brown[†] and Moshe Tennenholtz[‡]

[†] kevinlb@stanford.edu; Computer Science Department, Stanford University, Stanford CA 94305, USA

[‡] moshet@ie.technion.ac.il; Faculty of Industrial Engineering and Management, Technion, Haifa 32000, Israel

Abstract

We present a new class of games, local-effect games (LEGs), which exploit structure in a different way from other compact game representations studied in AI. We show both theoretically and empirically that these games often (but not always) have pure-strategy Nash equilibria. Finding a potential function is a good technique for finding such equilibria. We give a complete characterization of which LEGs have potential functions and provide the functions in each case; we also show a general case where pure-strategy equilibria exist in the absence of potential functions. In experiments, we show that myopic best-response dynamics converge quickly to pure strategy equilibria in games not covered by our positive theoretical results.

1 Introduction

Games have long been studied in AI as a model of competitive multiagent interactions. In particular, many researchers in AI have been interested in finding Nash equilibria (c.f. [5; 2]; for an introduction to games and equilibrium concepts, see e.g. [3]). Recently, there has been a lot of work on compact representations of games with large numbers of players, and games for which the computation of equilibria is tractable [8; 15; 6; 7; 14; 9]. Much work in this vein has been based on the exploitation of one of two kinds of locality. First, some approaches exploit unconditional independencies between players' abilities to affect each other's payoffs [8; 15; 6; 1]. Second, some approaches exploit symmetry in utility functions along with context-specific independencies between players' effects on each other; more precisely, games in which players' abilities to affect each other depend on the actions they choose. Here we study games in this second class, because we believe that this sort of context-specific independence is more common in real-world games.

Although compact representation has not been a primary motivation for economists, some work from economics does fall into the framework defined above. Most influentially, Rosenthal defined congestion games [13]. In these games each agent i selects a subset S_i from an available set of resources R ; where n_r is the number of agents who choose resource $r \in R$, and F_r are arbitrary functions for each r , agent i pays:

$$p_i(S_i, n) = \sum_{r \in S_i} F_r(n_r) \quad (1)$$

Observe that agent payoffs exhibit no unconditional independencies: all agents are given the same action choices, and so all agents can affect each other's payoffs. On the other hand, context-specific independence does exist when two agents choose non-intersecting resource subsets. Rosenthal's main result in [13] was that congestion games always have pure-strategy Nash equilibria (PSNE). This is important because, although all games have mixed-strategy Nash equilibria [11], there are few known classes of interesting games with pure strategy equilibria. At the same time, pure strategy equilibria are attractive: they can be more likely to arise in play as they are more intuitive than mixed-strategy equilibria for many players; they can be easier for agents to coordinate to; as there are a finite number of pure strategy profiles in a given game, they can be easier to compute than mixed strategy equilibria.

Rosenthal's work was extended by Monderer and Shapley [10], who showed that the class of congestion games is equivalent to the class of games with potential functions. Potential functions map agents' joint actions to a real number, with the property that if X and Y are strategy profiles differing only in the action choice of one agent i , $P(X) - P(Y)$ is equal to the difference in i 's payoff from selecting the two actions. This result is useful because it means that the construction of a potential function is sufficient for showing the existence of a pure-strategy equilibrium. Potential functions can also be used to compute equilibria: the set of pure-strategy Nash equilibria is equivalent to the set of strategy profiles that maximize P .

Recent work in computer science and AI has explored classes of games inspired by and extending congestion games. For example, Kearns et al. examined games with bounded effects [7], and Roughgarden studied a nonatomic variant [14]. In this paper we propose a new class, which we call local-effect games.

2 Local-Effect Games

In congestion games, whenever two agents affect each other's payoff, they each do so by the same amount. Local-effect games (LEGs) model situations where agents' effects on each other are potentially asymmetric. Generally, action A locally affects action B if the utility of agents taking action B depends on some function $F_{A,B}$ of the number of agents taking action A , but the utility of agents taking action A depends on a different function $F_{B,A}$ of the number of agents taking action B .

There are many natural settings which are modeled by such locally-affecting actions. One problem domain that has been

studied in Economics for a three quarters of a century is the location problem [4]. These problems model situations where agents must choose a location to operate their business in the presence of other competing agents, and each agent's profit depends on how far she is from her competitors. The canonical example concerns ice cream vendors who must choose a spot on the beach to set up a kiosk, with agents' utility depending on how many other ice cream sellers have located themselves in the same or adjacent areas. Work from Economics on this problem has usually dealt with one-dimensional continuous spaces and has not modeled local effects explicitly; also, game theoretic analyses have typically considered only a small (e.g. 2 or 3) number of agents (c.f. [12]). It is easy to think of many variants on the location problem: ice cream sellers arranging themselves around a lake (ring structure); vendors opening coffee houses in a city (grid structure); pairs at a cocktail party trying to pick a quiet room, with noise proportional to the number of people in the room, and noise also emanating from nearby rooms (arbitrary structure).

Another natural domain modeled by LEGs is a role formation game, where agents can take on one of a set of partially-substitutable roles. Agents are rewarded according to the amount of work they do, so their payoff is reduced as other agents adopt the same or similar roles.

Formally, let $G = \langle \mathcal{A}, \mathcal{F}, n \rangle$ be a local-effect game for n agents. $\mathcal{A}$ is the set of actions available to each player in the game. Let D denote the distribution of players across actions, and $D(a)$ denote the number of players who chose action $a \in \mathcal{A}$. For every pair of actions $a, a' \in \mathcal{A}$, $\mathcal{F}_{a,a'} : \mathbb{Z}^+ \rightarrow \mathbb{R}^+$ is the cost function expressing the cost due to the local effect from action a to action a' , and depending on the number of agents having chosen action a . From this we can build the cost function of an agent a who chose action $i \in \mathcal{A}$:

$$\text{cost}(a) = \mathcal{F}_{i,i}(D(i)) + \sum_{j \in \mathcal{A}, j \neq i} \mathcal{F}_{j,i}(D(j)). \quad (2)$$

We assume that in all local-effect games $\mathcal{F}$ is **strictly monotonic**: that $\forall a, a' \neq a \in \mathcal{A}$ $\mathcal{F}_{a,a'}(x)$ either increases strictly monotonically with x or is always 0. Furthermore we assume $\forall a, a' \neq a \in \mathcal{A}$ $\mathcal{F}_{a,a'}(0) = 0$.

It is useful to think of a directed graph representing the actions and their local effects. We create a node for every action, and draw an edge from node i to node j if $\forall x$ $\mathcal{F}_{i,j}(x) = 0$ is false. We will sometimes denote functions of the form $\mathcal{F}_{i,i}(x)$ as *node functions*, and functions of the form $\mathcal{F}_{i,j}(x)$, $i \neq j$ as *edge functions*. Let $\text{neigh}(i)$ denote the set of nodes to which there are directed edges originating at node i .

We make one assumption about this graph's connectivity:

$$\forall A, B \in \mathcal{A}, B \in \text{neigh}(A) \rightarrow A \in \text{neigh}(B). \quad (3)$$

That is, each pair of nodes in the graph are either both or neither neighbors of each other, though they might influence each other according to different local-effect functions.

Definition 1 A local-effect game is a **bidirectional local-effect game (B-LEG)** when $\forall a \in \mathcal{A}, \forall a' \neq a \in \mathcal{A}$ $\mathcal{F}_{a,a'}(x) = \mathcal{F}_{a',a}(x)$.

For B-LEGs local-effect functions between pairs of actions are always the same in both directions; note however that for a given distribution of agents the *magnitude* of the local effects between a pair of actions may be different. The graphical representation of actions and local effects in B-LEGs is undirected.

Definition 2 A local-effect game is a **uniform local-effect game (U-LEG)** when $\forall A, B, C \in \mathcal{A}$ ($B \in \text{neigh}(A) \wedge C \in \text{neigh}(A)$) $\rightarrow \forall x$ $\mathcal{F}_{A,B}(x) = \mathcal{F}_{A,C}(x)$.

That is, if action A has any effect on nodes B and C then the same function governs its effect on both. We define notation for the uniform effect from node A : $\mathcal{F}_A^u(x) = \mathcal{F}_{A,\cdot}(x)$.

3 Theoretical Results

3.1 Nonexistence of Pure Strategy Equilibria

Rosenthal was able to show that congestion games always have a PSNE. For local-effect games, we can find counterexamples where exhaustive enumeration of strategies shows the absence of any PSNE, demonstrating that such a sweeping general result is impossible. One example (found experimentally, and confirmed by exhaustive search) is the B-LEG $G = \langle \{A, B, C\}, \mathcal{F}, 11 \rangle$, with $\mathcal{F}_{A,A}(x) = 2.79x$, $\mathcal{F}_{B,B}(x) = 4.72x$, $\mathcal{F}_{C,C}(x) = 1.5x$, $\mathcal{F}_{A,B}(x) = 0.64 \log x$, $\mathcal{F}_{A,C}(x) = 0.32 \log x$, $\mathcal{F}_{B,C}(x) = 2.77 \log x$.

3.2 Pure Strategy Equilibria: Potential Functions

In this section we show that two interesting classes of local-effect games have potential functions, meaning that they always have pure-strategy Nash equilibria. Although these results show regions of overlap between the class of congestion games and the class of local-effect games, the potential functions themselves are interesting as their construction is nontrivial. Also, these results are useful because they make it possible for the games to be described in the more intuitive local-effect game framework.

Theorem 1 Bidirectional local-effect games have pure strategy Nash equilibria if $\forall i, \forall j \neq i$ $\mathcal{F}_{i,j}(x) = m_{i,j}x$.

Sketch of Proof. Here we prove the result by giving a potential function:

$$P = \sum_{i=1}^n \sum_{j=1}^{D(i)} \mathcal{F}_{i,i}(j) + \frac{1}{2} \sum_{i=1}^n \sum_{j \in \text{neigh}(i)} D(i) m_{j,i} D(j) \quad (4)$$

The first term is the standard congestion game potential function. A game with only functions of the form $\mathcal{F}_{i,i}(x)$ is a congestion game, and so must have the congestion game potential function. The relationship between each $\mathcal{F}_{i,j}(x)$ function and the agent's cost function is additive, and potential functions are only used for taking differences. Thus if we can find a potential function P' for a game with only local effects and all $\mathcal{F}_{i,i}(x) = 0$, the potential function for a general B-LEG will be the sum of the congestion game potential function and P' .

Thus it remains to argue that our second term is this P' : that it captures changes in utility arising from local effects.

Consider the sum of the contribution of local effects to each agent's utility: $s = \sum_{i=1}^n \sum_{j \in \text{neigh}(i)} D(i)m_{j,i}D(j)$. When a single agent a deviates, s increases by twice the amount of the change to a 's utility, because all $\mathcal{F}_{i,j}(x)$ are linear and bidirectional. That is, there is a change both in the amount of local effect acting on agent a , and new local effect caused by agent a , and bidirectionality and linearity imply that these two amounts are the same. Thus the desired result is obtained by adding $\frac{1}{2}s$ to the congestion game potential function. $\square$

Observe that Theorem 1 holds for B-LEGs with non-linear functions $\mathcal{F}_{i,i}(x)$ —what is required is linearity of the local-effect functions.

Theorem 2 *Uniform local-effect games have pure strategy Nash equilibria if the local-effect graph is a clique.*

Sketch of Proof. Again we provide a potential function:

$$P = \sum_{i=1}^n \sum_{j=1}^{D(i)} \mathcal{F}_{i,i}(j) - \sum_{i=1}^n \sum_{j=1}^{D(i)-1} \mathcal{F}_i^u(j) \quad (5)$$

As argued in Theorem 1, to construct a potential function it is sufficient to add the standard congestion game potential function with a function that accounts for changes in utility due to local effects. This explains the first term.

Let distributions X and Y be identical except that $D_X(A) = \alpha$ and $D_X(B) = \beta$, while $D_Y(A) = \alpha - 1$ and $D_Y(B) = \beta + 1$. Assuming $\mathcal{F}_{A,A}(\cdot) = \mathcal{F}_{B,B}(\cdot) = 0$, $P(X) - P(Y) = \mathcal{F}_B^u(\beta) - \mathcal{F}_A^u(\alpha - 1)$. This is precisely the change in utility for an agent deviating from A in X to B in Y : the agent will be spared the local effect $\mathcal{F}_B^u(\beta)$ since he moves to B and is no longer subject to its local effect; however, since he moves away from A and the graph is a clique, he is now subject to the local effect $\mathcal{F}_A^u(\alpha - 1)$. Because the graph is a clique, and because the game is a U-LEG, the argument holds no matter which pair of nodes is chosen as A and B . $\square$

3.3 LEGs and Potential Functions

Finding potential functions is an effective way of proving the existence of pure-strategy equilibria; however, there are many LEGs for which potential functions can be shown not to exist. In this section we give a complete characterization of the class of LEGs which have potential functions.

Lemma 1 *A local-effect game has a potential function if $\forall a, a' \in \mathcal{A} \mathcal{F}_{a,a'}(\cdot) = 0$.*

Sketch of Proof. Trivially, a LEG without any local effects is a congestion game. $\square$

Lemma 2 *The class of potential games does not contain the class of local-effect games for which $\exists A, B, C \in \mathcal{A}$ where $B \in \text{neigh}(C)$ and not $A \in \text{neigh}(B)$ and not $A \in \text{neigh}(C)$ and $(\mathcal{F}_{B,C} \neq \mathcal{F}_{C,B}$ or $\mathcal{F}_{B,C}$ is nonlinear).*

Sketch of Proof. Assume for contradiction that every LEG in the class has a potential function P . We will consider three distributions of agents in order to derive properties of P . Without loss of generality, we take A, B and C to be the

first three actions in the game, and we take the total number of actions to be n . For more compact notation in what follows, let $\alpha = D(A)$, $\beta = D(B)$ and $\gamma = D(C)$. Define the following three distributions: $X = (\alpha, \beta, \gamma, D(4), \dots, D(n))$, $Y = (\alpha - 1, \beta + 1, \gamma, D(4), \dots, D(n))$ and $Z = (\alpha, \beta + 1, \gamma - 1, D(4), \dots, D(n))$. Without making any assumptions about the local effects between actions A, B and C and any of the other $n - 3$ actions, and for $x \in \{A, B, C\}$, let:

$$U_x(D(x), D(4), \dots, D(n)) \\ = \mathcal{F}_{x,x}(D(x)) + \sum_{a' \in \{4, \dots, n\}} \mathcal{F}_{a',a}(D(a')) \quad (6)$$

That is, $U_x(D(x), D(4), \dots, D(n))$ denotes the (negative) utility contributed to each agent taking action $x \in \{A, B, C\}$ by those agents also taking action x , and by those agents taking the 4th through n th actions. For compactness we will abbreviate $U_x(D(x), D(4), \dots, D(n))$ as $u_x(D(x))$ below.

If distribution X were the case and an agent playing action A switched to action B , then distribution Y would be the result. Thus:

$$P(X) - P(Y) = [u_A(\alpha)] - [\mathcal{F}_{C,B}(\gamma) + u_B(\beta + 1)] \quad (7)$$

If X were the case and an agent playing action C switched to action B , then Z would be the result. Thus:

$$P(X) - P(Z) = [\mathcal{F}_{B,C}(\beta) + u_C(\gamma)] - [\mathcal{F}_{C,B}(\gamma - 1) + u_B(\beta + 1)] \quad (8)$$

If Y were the case and an agent playing action C switched to action A , then Z would be the result. Thus:

$$P(Y) - P(Z) = [\mathcal{F}_{B,C}(\beta + 1) + u_C(\gamma)] - [u_A(\alpha)] \quad (9)$$

From equations (7) and (8), we can infer:

$$P(Y) - P(Z) = P(Y) - P(Z) + [P(X) - P(X)] \\ = [P(X) - P(Z)] - [P(X) - P(Y)] \\ = [[\mathcal{F}_{B,C}(\beta) + u_C(\gamma)] - [\mathcal{F}_{C,B}(\gamma - 1) \\ + u_B(\beta + 1)]] - [[u_A(\alpha)] \\ - [\mathcal{F}_{C,B}(\gamma) + u_B(\beta + 1)]] \quad (10)$$

Intersect equation (10) with equation (9) and rearrange. Observe that $u_A(\alpha)$, $u_B(\beta + 1)$ and $u_C(\gamma)$ all cancel out, demonstrating that this proof does not depend on what edges exist between A, B, C and the rest of the graph, or on node effects. Define $\mathcal{I}_{a,a'}(x) = \mathcal{F}_{a,a'}(x) - \mathcal{F}_{a,a'}(x - 1)$: the incremental cost on the local effect between a and a' of adding the x th agent to a . We then get:

$$\mathcal{I}_{C,B}(\gamma) - \mathcal{I}_{B,C}(\beta + 1) = 0 \quad (11)$$

Clearly, equation (11) will not be satisfied for all β, γ unless $\mathcal{F}_{B,C} = \mathcal{F}_{C,B}$ and $\mathcal{F}_{B,C}$ is linear. This contradicts our assumption that a potential function exists for every LEG in the class. $\square$

Lemma 3 *The class of potential games does not contain the class of local-effect games for which $\exists A, B, C \in \mathcal{A}$ where $B \in \text{neigh}(C)$ and $A \in \text{neigh}(B)$ and not $A \in \text{neigh}(C)$ and $(\mathcal{F}_{B,C} \neq \mathcal{F}_{C,B}$ or $\mathcal{F}_{B,C}$ is nonlinear or $\mathcal{F}_{A,B} \neq \mathcal{F}_{B,A}$ or $\mathcal{F}_{A,B}$ is nonlinear).*

Sketch of Proof. This proof follows the proof of Lemma 2 and uses the same setting and definitions, except that (as stated in the theorem) $A \in \text{neigh}(B)$. Using the same arguments about distributions X, Y and Z we can derive:

$$[\mathcal{I}_{C,B}(\gamma) - \mathcal{I}_{B,C}(\beta+1)] + [\mathcal{I}_{B,A}(\beta+1) - \mathcal{I}_{A,B}(\alpha)] = 0 \quad (12)$$

Clearly, equation (12) will not be satisfied for all α, β, γ unless $\mathcal{F}_{A,B} = \mathcal{F}_{B,A}, \mathcal{F}_{B,C} = \mathcal{F}_{C,B}$, and both $\mathcal{F}_{A,B}$ and $\mathcal{F}_{B,C}$ are linear. This contradicts our assumption that a potential function exists for every LEG in the class. $\square$

Lemma 4 *The class of potential games does not contain the class of local-effect games for which $\exists A, B, C \in \mathcal{A}$ where $B \in \text{neigh}(C)$ and $A \in \text{neigh}(B)$ and $A \in \text{neigh}(C)$ and $(\mathcal{F}_{B,C} \neq \mathcal{F}_{C,B}$ or $\mathcal{F}_{B,C}$ is nonlinear or $\mathcal{F}_{A,B} \neq \mathcal{F}_{B,A}$ or $\mathcal{F}_{A,B}$ is nonlinear or $\mathcal{F}_{A,C} \neq \mathcal{F}_{C,A}$ or $\mathcal{F}_{A,C}$ is nonlinear) and $(\mathcal{F}_{A,B} \neq \mathcal{F}_{A,C}$ or $\mathcal{F}_{B,A} \neq \mathcal{F}_{B,C}$ or $\mathcal{F}_{C,A} \neq \mathcal{F}_{C,B})$.*

Sketch of Proof. This proof follows the proof of Lemmas 2 and 3 and uses the same setting and definitions, except that (as stated in the theorem) $A \in \text{neigh}(B)$ and $A \in \text{neigh}(C)$. Using the same arguments about distributions X, Y and Z we can derive:

$$[\mathcal{I}_{A,C}(\alpha) - \mathcal{I}_{A,B}(\alpha)] + [\mathcal{I}_{B,A}(\beta+1) - \mathcal{I}_{B,C}(\beta+1)] + [\mathcal{I}_{C,B}(\gamma) - \mathcal{I}_{C,A}(\gamma)] = 0 \quad (13)$$

Equation (13) may be rewritten as:

$$[\mathcal{I}_{A,C}(\alpha) - \mathcal{I}_{C,A}(\gamma)] + [\mathcal{I}_{B,A}(\beta+1) - \mathcal{I}_{A,B}(\alpha)] + [\mathcal{I}_{C,B}(\gamma) - \mathcal{I}_{B,C}(\beta+1)] = 0 \quad (14)$$

From equation (13) we can see that the contradiction does not obtain for all α, β, γ when $\mathcal{F}_{A,B} = \mathcal{F}_{A,C}, \mathcal{F}_{B,A} = \mathcal{F}_{B,C}$, and $\mathcal{F}_{C,A} = \mathcal{F}_{C,B}$. From equation (14) we can see that the contradiction does not obtain for all α, β, γ when $\mathcal{F}_{A,B} = \mathcal{F}_{B,A}, \mathcal{F}_{B,C} = \mathcal{F}_{C,B}, \mathcal{F}_{A,C} = \mathcal{F}_{C,A}$ and $\mathcal{F}_{A,B}, \mathcal{F}_{A,C}$ and $\mathcal{F}_{B,C}$ are all linear. If neither condition holds, our assumption a potential function exists for every LEG in the class is contradicted. $\square$

We can now completely characterize those classes of LEGs that have potential functions.

Theorem 3 *The class of potential games contains the class of local-effect games for which any of the following hold:*

1. *the local-effect graph contains fewer than three nodes*
2. *the local-effect graph contains no edges*
3. *the game is a bidirectional local-effect game and all local-effect functions are linear*

4. *the game is a uniform local-effect game and the local-effect graph is a clique*

No other local-effect games have potential functions.

Sketch of Proof. First, it is clear that a game with only a single action has a potential function. A LEG with only two actions is trivially a U-LEG and has a local-effect graph which is a clique, so by Theorem 2 it has a potential function. This proves statement 1 in the theorem, and leaves us to consider LEGs which have 3 or more actions. We will do a case analysis considering all possible local-effect graph structures for these LEGs. Clearly, all graphs with 3 or more nodes are included if we consider all graphs with no edges, all cliques, and all graphs containing subgraphs having three nodes and either exactly one or exactly two edges.

Lemma 1 proves that if the local-effect graph has no edges then the LEG is a congestion game, proving statement 2.

If the local-effect graph contains a subgraph with three nodes and exactly one edge, and is not a B-LEG with linear functions, Lemma 2 shows that it does not have a potential function. If it is a B-LEG with linear functions, Theorem 1 shows that it has a potential function, proving statement 3.

If the local-effect graph contains a subgraph with three nodes and exactly two edges, and is not a B-LEG with linear functions (again, statement 3), Lemma 3 shows that it does not have a potential function.

If the local-effect graph is a clique, it contains a clique of size three as a subgraph. If the graph is not a U-LEG, Lemma 4 shows that it does not have a potential function. If it is a U-LEG, Theorem 2 shows that it does have a potential function, proving statement 4. $\square$

3.4 Other Pure-Strategy Equilibria

We are also able to prove the existence of pure-strategy Nash equilibria for classes of graphs, and node and edge functions that Theorem 3 shows cannot have potential functions. The following constructive proof has classes of B-LEGs and U-LEGs as special cases:

Theorem 4 *If a local-effect game satisfies*

1. $\forall A \in \mathcal{A}, \forall B \in \text{neigh}(A), \forall x, \mathcal{F}_{A,A}(x) \leq \mathcal{F}_{A,B}(x)$
2. $\forall A, B \in \mathcal{A}, \forall x \geq 1, \mathcal{F}_{A,B}(x+1) - \mathcal{F}_{A,B}(x) \leq \mathcal{F}_{A,B}(x) - \mathcal{F}_{A,B}(x-1)$,

then there exists a pure-strategy Nash equilibrium in which agents choose nodes that constitute an independent set.

Sketch of Proof. This proof proceeds by induction, building up a Nash equilibrium one agent at a time, and with each agent making a myopic best response to the previous distribution. In the case of a single agent, it is clearly an equilibrium for him to select the best node. Define D_i as the distribution of agents at induction step i . Assume that $n-1$ agents have each selected the best node in turn, resulting in a distribution D_{n-1} which is a Nash equilibrium and also an independent set. We must show that when an additional agent n chooses the best node the resulting distribution D_n is still an independent set, and still a Nash equilibrium.

First, we show that the new distribution is an independent set. Assume for the purposes of contradiction that it was best

for n to choose a node that does not belong to the independent set. Then it must be the case that the selected node has at least one neighbor which has been chosen by one or more other agents. Let the node selected by n be A , and let B be some neighboring node. From condition 2 in the theorem (linearity/sublinearity), we can infer that:

$$F_{B,B}(D_{n-1}(B) + 1) \leq F_{B,B}(D_{n-1}(B)) + F_{B,B}(1) \quad (15)$$

From condition 1 in the theorem (functional dominance), we know that:

$$F_{B,B}(D_{n-1}(B)) \leq \mathcal{F}_{B,A}(D_{n-1}(B)) \quad (16)$$

Thus we can use equation (16) to weaken the bound in equation (15) to get:

$$F_{B,B}(D_{n-1}(B) + 1) \leq \mathcal{F}_{B,A}(D_{n-1}(B)) + F_{B,B}(1) \quad (17)$$

Define the utility at inductive step i for an agent taking action X , and disregarding any local effect from action Y :

$$U_{X,\sim Y}^i(z) = \mathcal{F}_{X,X}(z) + \sum_{W \in \text{neigh}(X) | W \neq Y} \mathcal{F}_{W,X}(D_i(W)).$$

At some step i in the induction, $D_i(B) = 0$ and $D_i(A) = 0$, but $D_{i+1}(B) = 1$. From the fact that the distribution of agents resulted from myopic choices (stated in the induction hypothesis), we know that:

$$F_{B,B}(1) \leq U_{A,\sim B}^i(1) \quad (18)$$

We can use $U_{A,\sim B}(1)$ in equation (18) because $D_i(B) = 0$ anyway. From the monotonicity of local-effect functions, and the fact that $i \leq n$ we can write:

$$U_{A,\sim B}^i(1) \leq U_{A,\sim B}^n(1) \quad (19)$$

We can use equation (19) to weaken the bound given in equation (18):

$$F_{B,B}(1) \leq U_{A,\sim B}^n(1) \quad (20)$$

Finally, we can use equation (20) to further weaken the bound given in equation (17). This gives us:

$$F_{B,B}(D_{n-1}(B) + 1) \leq \mathcal{F}_{B,A}(D_{n-1}(B)) + U_{A,\sim B}^n(1) \quad (21)$$

Equation (21) contradicts our assumption that agent n would myopically choose A over B ; therefore D_n must be an independent set.

Now we show that D' is a Nash equilibrium. Let C be the node that the new agent i selected in making his myopic response to the distribution D . From symmetry of cost functions we know that no agent can profitably deviate from node C : if so, i would have chosen a different node in the first place. Consider an agent j who chose a node $V \neq C$. Agent j 's payoff does not change from distribution D to distribution D' , because D' is an independent set, and so $\mathcal{F}_{C,V}(\cdot) = 0$ (there are no local effects between nodes C and V). Since distribution D was a Nash equilibrium (inductive hypothesis) j will not deviate from a new distribution D' that differs only in that node C is more costly. $\square$

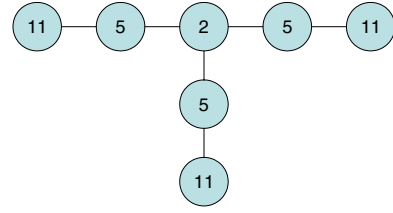


Figure 1: $T - k_n = 3$, 50 agents

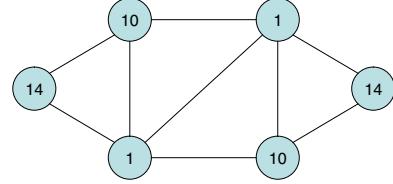


Figure 2: Arbitrary - $k_n = 3$, 50 agents

4 Empirical Findings

Section 3 shows that there are many cases in which local-effect games have pure-strategy Nash equilibria. Myopic best response has been shown to be an effective technique for computing pure strategy equilibria in a variety of settings [10]. In this section we show that this simple algorithm can compute pure strategy equilibria for very large local-effect games that are not covered by any of the positive results in section 3 and that do not have potential functions. We present five different graph structures with similar local-effect functions, and show sample equilibria. We should note that we have been able to find equilibria experimentally for most B-LEGs¹ with different graph structure and different local-effect functions that we have tried, and that convergence occurs within a second in most cases. As with our theoretical results, we do not claim that these equilibria are unique; indeed, because agents' cost functions are symmetric, a new equilibrium can always be constructed from a given equilibrium by swapping action choices between pairs of agents. Furthermore, we have observed many cases where multiple structurally different equilibria exist in the same local-effect game.

All games shown here are B-LEGs with $\forall A, \forall B \mathcal{F}_{A,B}(x) = k_{A,B} \log(x + 1)$. We use one k_n for all node functions and another k_e for all edge functions (i.e., $\forall A, \forall B \neq A \ k_{A,A} = k_{B,B}$ and $\forall A, \forall B \neq A, \forall C \notin \{A, B\} \ k_{A,B} = k_{A,C}$). We hold $k_e = 1$ throughout, and vary k_n to highlight some of the more interesting equilibria. These equilibria are representative of average runs, and were found with a minimum of parameter manipulation. Each node is labelled with the number of agents choosing the node in equilibrium. Fig. 1 shows a T structure representative of a simple location problem. Fig. 2, which we call 'arbitrary' in what follows,

¹So far, we have only experimented with B-LEGs because undirected local-effect graphs are easier to specify and generate, and because we consider them to be among the most natural LEGs. We expect to experiment with other classes of LEGs in our future work.

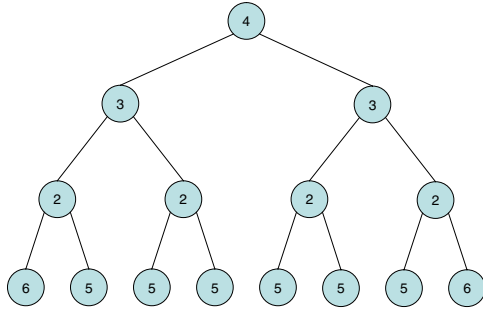


Figure 3: Binary Tree – $k_n = 6$, 60 agents

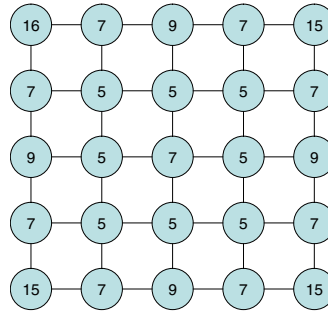


Figure 4: Grid – $k_n = 4$, 200 agents

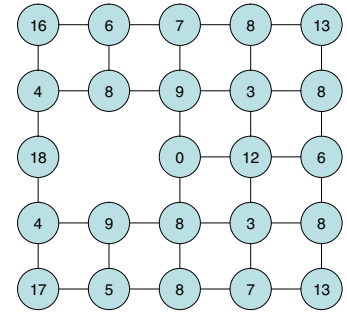


Figure 5: Modified Grid – $k_n = 4$, 200 agents

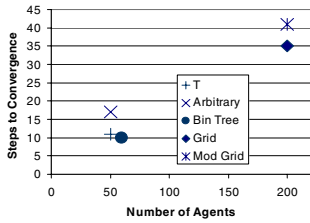


Figure 6: Steps to convergence for the five graphs

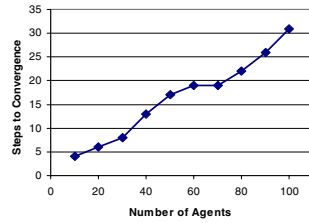


Figure 7: Steps to convergence for the arbitrary graph

is interesting because there are 2 nodes with 2 neighbors, 2 with 3 neighbors and 2 with 4 neighbors. This setting could represent a role formation game. Fig. 3 shows a binary tree structure; observe that most agents select leaf nodes because they have only one neighbor, and thus the parents of leaves are chosen by few agents. Fig. 4 shows a two-dimensional grid, representative of our coffee house location problem. Observe that the corners are most desirable, as they have only two neighbors; nodes neighboring corners are thus under-populated, leading to another concentration of agents in the middle of each edge. It is also interesting that agents concentrate in the central node, even though it has four neighbors, because its neighbors are relatively under-populated. Fig. 5 shows what happens to the game from Fig. 4 when we remove a single node (consider the same location problem when one node becomes unavailable). Observe that agents generally cluster around the missing node, except for one neighboring node that is entirely unpopulated, as a result of the large local effects acting upon it.

The amount of time it took to reach convergence in each graph is shown in Fig. 4, starting in each case with a uniform distribution of agents across the actions. Finally, since the ‘arbitrary’ graph in Fig. 2 took the longest to converge (34% of the agents moved before convergence occurred) we examine this graph in more detail in Fig. 4. Observe that as we vary the number of agents, the number of steps required for convergence increases roughly linearly.

5 Conclusion

Local-effect games exploit context-specific independence between players’ payoff functions. Finding a potential func-

tion is a good technique for finding equilibria; we identify all the local-effect games for which potential functions exist, and provide the potential function in each case. We also give a positive theoretical result for a broad class of games that do not have potential functions. Furthermore, we can show that myopic best response dynamics converge quickly in other cases which do not have potential functions, and are also not covered by our positive theoretical results.

References

- [1] B. Blum, C. Shelton, and D. Koller. A continuation method for Nash equilibria in structured games. In *IJCAI*, 2003.
- [2] V. Conitzer and T. Sandholm. Complexity Results about Nash Equilibria. Technical Report CM-CS-02-135, CMU, 2002.
- [3] D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, 1991.
- [4] H. Hotelling. Stability in competition. *Economic Journal*, 39:41–57, 1929.
- [5] J. Hu and M.P. Wellman. Multi-agent reinforcement learning: Theoretical framework and an algorithm. In *ICML*, 1998.
- [6] M. Kearns, M. Littman, and S. Singh. Graphical models for game theory. In *UAI*, 2001.
- [7] M. Kearns and Y. Mansour. Efficient Nash computation in large population games with bounded influence. In *UAI*, 2002.
- [8] D. Koller and B. Milch. Multi-agent influence diagrams for representing and solving games. In *IJCAI*, 2001.
- [9] Pierfrancesco La Mura. Game networks. In *UAI-00*, pages 335–342, 2000.
- [10] D. Monderer and L.S. Shapley. Potential games. *Games and Economic Behavior*, 14:124–143, 1996.
- [11] J.F. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences of the United States of America*, 36:48–49, 1950.
- [12] M.J. Osborne and C. Pitchik. Equilibrium in Hotelling’s model of competition. *Econometrica*, 55:911–922, 1987.
- [13] R.W. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2:65–67, 1973.
- [14] T. Roughgarden and E. Tardos. Bounding the inefficiency of equilibria in nonatomic congestion games. Technical Report TR2002-1866, Cornell, June 2001.
- [15] D. Vickrey and D. Koller. Multi-agent algorithms for solving graphical games. In *AAAI*, 2002.

MULTIAGENT SYSTEMS

NONMANIPULABILITY AND FAULT-TOLERANCE

Universal Voting Protocol Tweaks to Make Manipulation Hard*

Vincent Conitzer and Tuomas Sandholm

Carnegie Mellon University
Computer Science Department
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
{conitzer,sandholm}@cs.cmu.edu

Abstract

Voting is a general method for preference aggregation in multiagent settings, but seminal results have shown that all (nondictatorial) voting protocols are manipulable. One could try to avoid manipulation by using voting protocols where determining a beneficial manipulation is hard computationally. A number of recent papers study the complexity of manipulating existing protocols. This paper is the first work to take the next step of *designing new protocols that are especially hard to manipulate*. Rather than designing these new protocols from scratch, we instead show how to *tweak* existing protocols to make manipulation hard, while leaving much of the original nature of the protocol intact. The tweak studied consists of adding one elimination preround to the election. Surprisingly, this extremely simple and universal tweak makes typical protocols hard to manipulate! The protocols become NP-hard, #P-hard, or PSPACE-hard to manipulate, depending on whether the schedule of the preround is determined before the votes are collected, after the votes are collected, or the scheduling and the vote collecting are interleaved, respectively. We prove general sufficient conditions on the protocols for this tweak to introduce the hardness, and show that the most common voting protocols satisfy those conditions. These are the first results in voting settings where manipulation is in a higher complexity class than NP (presuming $\text{PSPACE} \neq \text{NP}$).

1 Introduction

Often, a group of agents has to make a common decision, yet they have different preferences about which decision is made. Thus, it is of central importance to be able to aggregate the preferences, that is, to make a socially desirable decision as to which *candidate* is chosen from a set of candidates. Such candidates could be potential presidents, joint plans, allocations

of goods or resources, etc. Voting is the most general preference aggregation scheme, and has been used in several multiagent decision making problems in AI, such as collaborative filtering (e.g. [Pennock *et al.*, 2000]) and planning among automated agents (e.g. [Ephrati and Rosenschein, 1991; 1993]).

A key problem voting mechanisms are confronted with is that of *manipulation* by the voters. An agent is said to vote strategically when it does not rank the alternatives according to its true preferences, but differently so as to manipulate the outcome to be more favorable to the agent. For example, if an agent prefers Nader to Gore to Bush, but knows that Nader has too few other supporters to win, while Gore and Bush are close to each other, the agent would be better off by declaring Gore as its top candidate. Manipulation is an undesirable phenomenon. For one, because social choice schemes are tailored to aggregate preferences in a socially desirable way, and if the agents reveal their preferences insincerely, a socially undesirable candidate may be chosen.

A seminal negative result, the *Gibbard-Satterthwaite theorem*, shows that if there are three or more candidates, then in any nondictatorial voting scheme, there are preferences under which an agent is better off voting strategically [Gibbard, 1973; Satterthwaite, 1975]. (A voting scheme is called dictatorial if one of the voters dictates the social choice no matter how the others vote). In automated group decision making where the voters are software agents, the manipulability of protocols is even more problematic, for at least two reasons. First, the algorithms they use to decide how to vote must be coded explicitly. Given that the voting algorithm needs to be designed only once (by an expert), and can be copied to large numbers of agents (even ones representing unsophisticated human voters), it is likely that rational strategic voting will increasingly become an issue, unclouded by irrationality, emotions, etc. Second, software agents have more computational power and are more likely to find effective manipulations.

We take the following tack toward avoiding manipulation: *ensuring that finding a beneficial manipulation is so hard computationally that it is unlikely that voters will be able to manipulate*. So, unlike in most of computer science, here high computational complexity is a desirable property. The harder it is to manipulate, the better.

Prior work on the complexity of manipulating elections has focused on existing protocols [Bartholdi *et al.*, 1989; Bartholdi and Orlin, 1991; Conitzer and Sandholm, 2002a].

*The material in this paper is based upon work supported by the National Science Foundation under CAREER Award IRI-9703122, Grant IIS-9800994, ITR IIS-0081246, and ITR IIS-0121678.

This paper is the first to take the next step of designing new protocols that are especially hard to manipulate. Rather than designing these protocols from scratch, we show how to tweak existing protocols to make manipulation computationally much more difficult, while leaving much of the original nature of the protocol intact, for the following reasons:

- Results on the computational complexity induced by a tweak typically apply to a large family of protocols.
- Many of the original protocol's nice theoretical properties are preserved by the tweak.
- In practice, it will be much easier to replace a currently used protocol with a tweaked version of it, than with an altogether new protocol.

The type of tweak we study in this paper is the following. All the candidates are paired in a *preround*; of each pair of candidates, only the winner of their *pairwise election* survives. (The winner of the pairwise election between two candidates is the candidate that is ranked above the other more often in the votes.) After the preround, the original protocol is executed on the remaining candidates. The *schedule* of the preround (i.e., who faces who) can be determined before the votes are collected; after the votes are collected; or while the votes are collected (the processes are interleaved). We study these three cases in Sections 4, 5, and 6, respectively.¹

2 Definitions

2.1 Elections and voting protocols

An *election* consists of a set of candidates C ; a set of voters V ; and a protocol for deciding on a winner $w \in C$ given all the voters' votes. (Here, a vote is a total ordering of the candidates.) A *deterministic* protocol is a function from the set of all combinations of votes to C . (All our results hold even for unweighted voters, so in this paper this function will always treat the voters symmetrically.) A *randomized* protocol is a function from the set of all combinations of votes to probability distributions over C . An *interleaved* protocol is a procedure for alternating between collecting parts of the voters' votes (e.g. whether they prefer candidate a to candidate b) and drawing and publishing random variables (such as parts of the schedule for an election), together with a function from the set of all combinations of votes and random variables to C . (Collecting only parts of the voters' preferences is also known as *elicitation*.)

¹The high complexity results obtained in this paper are dependent on the number of candidates growing. This places them in line with all the early results in this area [Bartholdi *et al.*, 1989; Bartholdi and Orlin, 1991], but in contrast with more recent results [Conitzer and Sandholm, 2002a] that show high complexity of manipulation with a constant number of candidates for some protocols. Having high complexity of manipulation occur with a constant number of candidates already is certainly preferable to having it occur only with a growing number. On the other hand, unlike in that paper, the results here hold even when the voters all have equal weight, and even when manipulation is attempted by an individual rather than a coalition, making the results in this paper stronger in that sense, so there is a tradeoff.

The standard definitions of most voting protocols allow for the possibility of ties between candidates, in which case a tie-breaking rule is required to fully specify the protocol. All our results hold for any tie-breaking rule, so we do not need to specify such rules here.

In this paper we apply our technique to the most common voting protocols (in these definitions, whenever points are defined, the candidate with the most points wins):

- *Plurality*. A candidate receives 1 point for every voter that ranks it first. (Thus, the voters effectively only vote for one candidate.)
- *Borda*. For each voter, a candidate receives $m - 1$ points if it is the voter's top choice, $m - 2$ if it is the second choice, ..., 0 if it is the last.
- *Maximin*. A candidate's score in a *pairwise election* is the number of voters that prefer it over the opponent. A candidate's number of points is the lowest score it gets in any pairwise election.
- *Single Transferable Vote (STV)*. The winner determination process proceeds in rounds. In each round, a candidate's score is the number of voters that rank it highest among the remaining candidates, and the candidate with the lowest score drops out. The last remaining candidate wins. (A vote "transfers" from its top remaining candidate to the next highest remaining candidate when the former drops out.)

2.2 Preround

The tweaks we study in this paper all involve the addition of a preround. We will now define how this works.

Definition 1 Given a protocol P , the new protocol obtained by adding a preround to it proceeds as follows:

1. The candidates are paired. If there is an odd number of candidates, one candidate gets a bye.
2. In each pairing of two candidates, the candidate losing the pairwise election between the two is eliminated. A candidate with a bye is never eliminated.
3. On the remaining candidates, P is executed to produce a winner. For this, the implicit votes over the remaining candidates are used. (For example, if a voter voted $a \succ b \succ c \succ d \succ e$, and b and c were eliminated, the voter's implicit vote is $a \succ d \succ e$.)

The pairing of the candidates is also known as the schedule for the preround. If the schedule is decided and published before the votes are collected, we have a deterministic preround (DPRE), and the resulting protocol is called $DPRE + P$. If the schedule is drawn completely randomly after the votes are collected, we have a randomized preround (RPRE), and the resulting protocol is called $RPRE + P$. Finally, if the votes are elicited incrementally, and this elicitation process is interleaved with the scheduling-and-publishing process (which is again done randomly), as described in detail in Section 6, we have an interleaved preround (IPRE), and the resulting protocol is called $IPRE + P$.

2.3 Manipulation

We now define the computational problem of manipulation. Because all our hardness results hold even when the voters are unweighted, only a single voter is trying to manipulate, and all the other voters' votes are known to the manipulator, we will only define this simple setting here. Any hardness results in this simple setting immediately imply hardness in all more general settings.

Definition 2 (CONSTRUCTIVE-MANIPULATION)

We are given a protocol P , a candidate set C , a preferred candidate p , and a set of votes S corresponding to all the other voters' votes. The manipulator has yet to decide on its vote, and wants to make p win. Then the constructive manipulation question is:

- (For deterministic protocols) Can the manipulator cast its vote to make p win under P ?
- (For randomized protocols) Can the manipulator cast its vote to make the probability of p winning under P at least some given $k \in [0, 1]$?
- (For interleaved protocols) Given the initial random choices (if any) by the protocol, is there a contingency plan (based on the random decisions the protocol takes between eliciting parts of the votes) for the manipulator to answer the queries to make the probability of p winning under P at least some given $k \in [0, 1]$?

3 Complexity of manipulating untweaked protocols

In this section, we briefly review the complexity of manipulating voting protocols, as a benchmark for our results. CONSTRUCTIVE-MANIPULATION is in P for the Plurality, Borda, and Maximin voting protocols [Bartholdi *et al.*, 1989]. The only voting protocol for which CONSTRUCTIVE-MANIPULATION is known to be NP-hard is the STV protocol [Bartholdi and Orlin, 1991].²

4 NP-hardness when scheduling precedes voting

In this section, we examine the complexity induced by the preround when the voters know the schedule before they vote.

4.1 A sufficient condition for NP-hardness

We present a sufficient condition under which adding a preround with a preannounced schedule makes manipulation NP-hard. The condition can be thought of as an NP-hardness reduction template. If it is possible to reduce an arbitrary SAT instance to a set of votes satisfying certain properties under the given voting protocol, that protocol—with a preround—is NP-hard to manipulate.

Theorem 1 *Given a voting protocol P , suppose that it is possible, for any Boolean formula ϕ in conjunctive normal form*

²CONSTRUCTIVE-MANIPULATION is NP-hard also for the Second Order Copeland protocol [Bartholdi *et al.*, 1989], but the hardness is driven solely by the tie-breaking rule.

(i.e., a SAT instance), to construct in polynomial time a set of votes over a candidate set containing at least $\{p\} \cup C_L$ where $C_L = \{c_l : l \in L\}$ (L is the set of literals $\{+v : v \in V\} \cup \{-v : v \in V\}$, where V is the set of variables used in ϕ), with the following properties:

- (Property 1a) If we remove, for each $v \in V$, one of c_{+v} and c_{-v} , p would win an election under protocol P against the remaining candidates if and only if for every clause $k \in K$ (where K is the set of clauses in ϕ), there is some $l \in L$ such that c_l has not been removed, and l occurs in k . This should hold even if a single arbitrary vote is added.
- (Property 1b) For any $v \in V$, c_{+v} and c_{-v} are tied in their pairwise election after these votes.

Then CONSTRUCTIVE-MANIPULATION in $DPRE + P$ is NP-hard (and NP-complete if P is deterministic and can be executed in polynomial time).

Proof: Consider the following election under $DPRE + P$. Let the candidate set be the set of all candidates occurring in the votes constructed from ϕ (the "original candidates"), plus one dummy candidate for each of the original candidates besides those in C_L . To each of the constructed votes, add all the dummy candidates at the bottom; let the resulting set of votes be the set of the nonmanipulators' votes. A single manipulator's vote is yet to be added. Let the schedule for the preround be as follows: for each v , c_{+v} and c_{-v} face each other in the preround; and every other original candidate faces (and, because of the dummy candidates' position in the votes, defeats) a dummy candidate. Thus, the set of candidates that make it through the preround consists of, for each $v \in V$, one of c_{+v} and c_{-v} ; and all the other original candidates. The manipulator's vote will decide the winner of every c_{+v} vs. c_{-v} match-up, because by property 1b, all these pairwise elections are currently tied. Moreover, it is easy to see that the manipulator can decide the winner of each of these match-ups independently of how it decides the winners of the other match-ups. Thus, we can think of this as the manipulator giving the variables truth-values: v is set to *true* if c_{+v} survives, and to *false* if c_{-v} survives. By property 1a it then follows that p wins if and only if the manipulator's assignment satisfies all the clauses, i.e. is a solution to the SAT instance. Hence there is a successful constructive manipulation if and only if there is a solution to the SAT instance, and it follows that CONSTRUCTIVE-MANIPULATION in $DPRE + P$ is NP-hard. (It is also in NP if P is deterministic and can be executed in polynomial time, because in this case, given a vote for the manipulator, it can be verified in polynomial time whether this vote makes p win). ■

4.2 Examples

We now show how to apply Theorem 1 to the well-known protocols we discussed, thus showing that each of these protocols—with a preround—is NP-hard to manipulate.

Theorem 2 *There exists a reduction that satisfies properties 1a and 1b of Theorem 1 under the Plurality protocol.*

When it does not matter for our proofs whether a given vote is $a \succ b \succ c$ or $b \succ a \succ c$, we write $\{a, b\} \succ c$.

Proof: Given the formula ϕ , let the candidate set be the minimally required candidates $\{p\} \cup C_L$, plus a set of candidates corresponding to the set of clauses K of ϕ , $C_K = \{c_k : k \in K\}$. Then, let the set of votes be as follows: $4|K| + 2$ votes ranking the candidates $p \succ C_L \succ C_K$; for each $k \in K$, $4|K|$ votes ranking the candidates $c_k \succ \{c_{cl} \in C_K : cl \neq k\} \succ C_L \succ p$; and for each $k \in K$, 4 votes ranking the candidates $\{c_l \in C_L : l \in k\} \succ c_k \succ \{c_l \in C_L : l \notin k\} \succ \{c_{cl} \in C_K : cl \neq k\} \succ p$. Additionally, we require that these votes are such that after counting them, for each $v \in V$, c_{+v} and c_{-v} are tied in their pairwise election, so that property 1b is satisfied. (This is possible because the total number of votes is even, and the majority of the votes do not yet have any restrictions on the order of the C_L .) We now show property 1a is satisfied. We first observe that regardless of which of the candidates corresponding to literals are removed, p will get $4|K| + 2$ votes. Now, if for some $k \in K$, all the candidates c_l with $l \in L, l \in k$ are removed, then c_k will get at least $4|K| + 4$ votes and p will not win. On the other hand, if for each $k \in K$, at least one candidate c_l with $l \in k$ remains, then each of the c_k will get precisely $4|K|$ votes. Because each remaining c_l can get at most $4|K|$ votes as well, p will win. In both cases there is a "margin" of at least 2, so a single additional vote will not change this. Thus, property 1a is satisfied. ■

Theorem 3 *There exists a reduction that satisfies properties 1a and 1b of Theorem 1 under the Borda protocol.*

Proof: Given the formula ϕ , let the candidate set be the minimally required candidates $\{p\} \cup C_L$; plus a set of candidates corresponding to the set of clauses K of ϕ , $C_K = \{c_k : k \in K\}$, which we order in some arbitrary way to get $\{c_1, \dots, c_{|K|}\}$. Let M be the total number of candidates this defines. Then, let the set of votes be as follows: for every $c_i \in C_K$, $4M$ votes ranking the candidates $c_{i+1} \succ c_{i+2} \succ \dots \succ c_{|K|} \succ p \succ c_1 \succ c_2 \succ \dots \succ c_{i-1} \succ \{c_l \in C_L : l \in c_i\} \succ c_i \succ \{c_l \in L : l \notin c_i\}$; (here, the slight abuse of notation $l \in c_i$ means that l occurs in the clause corresponding to c_i); $4M$ votes ranking the candidates $c_1 \succ c_2 \succ \dots \succ c_{|K|} \succ p \succ C_L$; one vote $c_1 \succ c_2 \succ \dots \succ c_{|K|} \succ C_L \succ p$; one vote $c_{|K|} \succ c_{|K|-1} \succ \dots \succ c_1 \succ C_L \succ p$; and finally, $4|K|M$ votes ranking the candidates $p \succ c_1 \succ c_2 \succ \dots \succ c_n \succ C_L$, and $4|K|M$ votes ranking the candidates $c_n \succ c_{n-1} \succ \dots \succ c_1 \succ p \succ C_L$. Additionally, we require that these votes are such that after counting them, for each $v \in V$, c_{+v} and c_{-v} are tied in their pairwise election, so that property 1b is satisfied. (This is possible because the total number of votes is even, and the majority of the votes do not yet have any restrictions on the order of the c_l .) We now show property 1a is satisfied. It is easy to see that none of the c_l can win, regardless of which of them are removed. Thus, we only need to consider the c_i and p . The last $8|K|M$ votes will have no net effect on the relative scores of these candidates, so we need not consider these here. After the first $4(|K| + 1)M$ votes, any c_k for which all the c_l with $l \in k$ have been removed will be tied with p ,

and any other c_k will be at least $4M$ points behind p . Finally, from the last remaining two votes, any c_k ($k \in K$) will gain $2M - 2|V| - |K| - 1$ points on p . It follows that p wins if and only if for every clause $k \in K$, there is some $l \in L$ with $l \in k$ such that c_l has not been removed. In both cases there is a "margin" of at least $M - |V|$ points, so a single additional vote will not change this. Thus, property 1a is satisfied. ■

Theorem 4 *There exists a reduction that satisfies properties 1a and 1b of Theorem 1 under the Maximin protocol.*

Proof: Given the formula ϕ , let the candidate set be the minimally required candidates $\{p\} \cup C_L$, plus a set of candidates corresponding to the set of clauses K of ϕ , $C_K = \{c_k : k \in K\}$. Then, let the set of votes be as follows: $8|K|$ votes ranking the candidates $p \succ C_L \succ C_K$, $8|K|$ votes ranking the candidates $C_L \succ C_K \succ p$, and $8|K|$ votes ranking the candidates $C_K \succ p \succ C_L$; $4|K|$ votes ranking the candidates $C_L \succ p \succ C_K$, $4|K|$ votes ranking the candidates $C_K \succ C_L \succ p$, and, for each $k \in K$, 4 votes ranking the candidates $p \succ \{c_{cl} \in C_K : cl \neq k\} \succ \{c_l \in C_L : l \in k\} \succ c_k \succ \{c_l \in C_L : l \notin k\}$; and finally, 2 votes ranking the candidates $p \succ C_K \succ C_L$, and 2 votes ranking the candidates $C_K \succ p \succ C_L$. Additionally, we require that these votes are such that after counting them, for each $v \in V$, c_{+v} and c_{-v} are tied in their pairwise election, so that property 1b is satisfied. (This is possible because the total number of votes is even, and the majority of the votes do not yet have any restrictions on the order of the c_l .) We now show property 1a is satisfied. Regardless of which of the candidates corresponding to literals are removed, p 's worst score in a pairwise election is against any of the c_k , namely $16|K| + 2$. Any c_k for which all the c_l with $l \in k$ have been removed will get its worst pairwise election score against any of the C_L , namely $16|K| + 4$. Finally, any other c_k will get its worst pairwise election score against one of the c_l with $l \in k$, namely, $16|K|$. It follows that p wins if and only if for every clause $k \in K$, there is some $l \in k$ such that c_l has not been removed. In both cases there is a "margin" of at least 2, so a single additional vote will not change this. Thus, property 1a is satisfied. ■

Theorem 5 *There exists a reduction that satisfies properties 1a and 1b of Theorem 1 under the STV protocol.*

Proof: We omit the proof due to limited space. ■

Theorem 6 *In any of DPRE + Plurality, DPRE + Borda, DPRE + Maximin, and DPRE + STV³, CONSTRUCTIVE-MANIPULATION is NP-complete.*

Proof: NP-hardness is immediate from the previous theorems. The problem is in NP because these protocols can be

³The NP-completeness of manipulating DPRE + STV is, in itself, not that interesting, because STV is already NP-hard to manipulate without the preround as we discussed. Nevertheless, our method highlights a different aspect of the NP-hardness of manipulating DPRE + STV. We build on this reduction later to prove PSPACE-hardness of manipulating STV with a preround when the scheduling of the preround is interleaved with the vote elicitation.

executed in polynomial time. ■

In the next sections, we will raise the bar and bring the problem of manipulating elections to higher complexity classes by abandoning the assumption that the schedule for the preround should be known in advance.

5 #P-hardness when voting precedes scheduling

In this section, we will examine the complexity induced by the preround when the schedule is drawn completely (uniformly) randomly after all the votes have been collected.

5.1 A sufficient condition for #P-hardness

We present a sufficient condition for a voting protocol to become #P-hard⁴ to manipulate in this setting. Again, this condition can be thought of as a reduction template. If it is possible to reduce an arbitrary PERMANENT instance to a set of votes satisfying certain properties under the given voting protocol, that protocol is #P-hard to manipulate when a randomized preround is added to it. (In the PERMANENT problem, we are given a bipartite graph B with the same number of vertices k in both parts, and are asked how many matchings there are. This problem is #P-complete [Valiant, 1979].)

Theorem 7 *Given a voting protocol P , suppose that it is possible, for any bipartite graph B with the same number of vertices k in both parts (labeled 1 to k in one part, $k+1$ to $2k$ in the other), to construct in polynomial time a set of votes over the candidate set $\{c_1, \dots, c_{2k}, p\}$ (where c_i corresponds to vertex i in B) with the following properties:*

- (Property 2a) *If we remove k of the c_i , p would win an election under protocol P against the remaining c_i if and only if the removed c_i are exactly all the c_i with $k+1 \leq i \leq 2k$;*
- (Property 2b) *p loses its pairwise election against all c_i with $k+1 \leq i \leq 2k$;*
- (Property 2c) *For any $1 \leq i \leq k$ and $k+1 \leq j \leq 2k$, c_i defeats c_j in their pairwise election if and only if in B , there is an edge between vertices i and j .*
- (Property 2d) *All the previous properties still hold with any additional single vote.*

Then CONSTRUCTIVE-MANIPULATION in $RPRE + P$ is #P-hard.

Proof: Given the set of votes constructed on the basis of an arbitrary B , let us compute the probability that p wins under the protocol $RPRE + P$ with only these votes. In the preround, there are k matches and one bye. By property 2a, p will win the election if and only if the k candidates eliminated in this preround are precisely all the c_i with $k+1 \leq i \leq 2k$. By property 2b, p could not win a preround match against any of these, so p will win the election if and only if it gets the bye, and each of the c_j with $k+1 \leq j \leq 2k$ faces one of the c_i with $1 \leq i \leq k$ that defeats it in the preround. Then, by property

⁴#P is the class of problems where the task is to count the number of solutions to a problem in NP.

2c, it follows that p wins if and only if the preround pairing corresponds to a matching in B . Thus the probability of p winning is $\frac{m_B}{e(2k, 2k+1)}$, where m_B is the number of matchings in B and $e(2k, 2k+1)$ is the number of different ways to pair $2k$ of the $2k+1$ candidates in the preround (which is straightforward to compute). Thus, evaluating p 's chances of winning in this election is at least as hard as counting the number of matchings in an arbitrary B , which is #P-hard. Moreover, because we can compute p 's chances of winning solely on the basis of properties 2a, 2b, and 2c, and by property 2d, these properties are maintained for any single additional vote, it follows that a manipulator cannot affect p 's chances of winning. Thus, CONSTRUCTIVE-MANIPULATION in this case simply comes down to computing p 's chances of winning, which is #P-hard as demonstrated. ■

5.2 A broadly applicable reduction

In this subsection we present a single broadly applicable reduction which will satisfy the preconditions of Theorem 7 for many voting protocols, including all of the protocols discussed in this paper, thus proving them #P-hard to manipulate when the voting precedes the preround scheduling.

Definition 3 *We label the following reduction R_1 . Given a bipartite graph B with the same number of vertices k in both parts (labeled 1 to k in one part, $k+1$ to $2k$ in the other), we construct the following set of $12k^3 + 2k^2$ votes:*

- $6k^3$ votes that rank the candidates $c_{k+1} \succ c_{k+2} \succ \dots \succ c_{2k} \succ p \succ c_1 \succ c_2 \succ \dots \succ c_k$;
- $3k^2$ votes that rank the candidates $p \succ c_k \succ c_{k-1} \succ \dots \succ c_1 \succ c_{2k} \succ c_{2k-1} \succ \dots \succ c_{k+1}$;
- $6k^3 - 3k^2$ votes that rank the candidates $c_k \succ c_{k-1} \succ \dots \succ c_1 \succ c_{2k} \succ c_{2k-1} \succ \dots \succ c_{k+1} \succ p$;
- For each edge (i, j) in B ($1 \leq i \leq k$, $k+1 \leq j \leq 2k$), one vote that ranks the candidates $c_i \succ c_j \succ p \succ c_1 \succ c_2 \succ \dots \succ c_{i-1} \succ c_{i+1} \succ \dots \succ c_k \succ c_{k+1} \succ c_{k+2} \succ \dots \succ c_{j-1} \succ c_{j+1} \succ \dots \succ c_{2k}$, and another one that ranks them $c_{2k} \succ c_{2k-1} \succ \dots \succ c_{j+1} \succ c_{j-1} \succ \dots \succ c_{k+1} \succ c_k \succ c_{k-1} \succ \dots \succ c_{i+1} \succ c_{i-1} \succ \dots \succ c_1 \succ p \succ c_i \succ c_j$ (i.e., the inverse of the former vote, apart from c_i and c_j which have maintained their order);
- For each pair i, j without an edge between them in B ($1 \leq i \leq k$, $k+1 \leq j \leq 2k$), one vote that ranks the candidates $c_j \succ c_i \succ p \succ c_1 \succ c_2 \succ \dots \succ c_{i-1} \succ c_{i+1} \succ \dots \succ c_k \succ c_{k+1} \succ c_{k+2} \succ \dots \succ c_{j-1} \succ c_{j+1} \succ \dots \succ c_{2k}$, and another one that ranks them $c_{2k} \succ c_{2k-1} \succ \dots \succ c_{j+1} \succ c_{j-1} \succ \dots \succ c_{k+1} \succ c_k \succ c_{k-1} \succ \dots \succ c_{i+1} \succ c_{i-1} \succ \dots \succ c_1 \succ p \succ c_j \succ c_i$ (i.e., the inverse of the former vote, apart from c_j and c_i which have maintained their order).

We now have to show that this reduction satisfies the preconditions of Theorem 7. We start with the properties that are protocol-independent.

Theorem 8 *R_1 satisfies properties 2b and 2c of Theorem 7 (under any protocol P , because these properties are independent of P), even with a single additional arbitrary vote.*

Proof: In the pairwise election between p and any one of the c_i with $k+1 \leq i \leq 2k$, p is ranked higher in only $4k^2$ votes, and thus loses the pairwise election. So property 2b is satisfied. For a pairwise election between some c_i and c_j ($1 \leq i \leq k$ and $k+1 \leq j \leq 2k$), the first $12k^3$ votes' net contribution to the outcome in this pairwise election is 0. Additionally, the two votes associated with any pair q, r ($1 \leq q \leq k$ and $k+1 \leq r \leq 2k$) also have a net contribution of 0, if either $q \neq i$ or $r \neq j$. The only remaining votes are the two associated with the pair i, j , so c_i wins the pairwise election by 2 votes if there is an edge (i, j) in B , and c_j wins the pairwise election by 2 votes otherwise. So property 2c is satisfied. Because both are satisfied with a "margin" of at least 2, a single additional vote will not change this. ■

Finally, because property 2a is protocol-dependent, we need to prove it for our reduction on a per-protocol basis. This is what the following four theorems achieve.

5.3 Examples

Theorem 9 R_1 satisfies property 2a of Theorem 7 under the Plurality protocol. This holds even when there is a single additional arbitrary vote.

Proof: If at least one of the c_i with $k+1 \leq i \leq 2k$ is not removed, p can get at most $5k^2$ votes, whereas the lowest-indexed remaining candidate among the c_i with $k+1 \leq i \leq 2k$ will get at least $6k^3$ votes, so p does not win. On the other hand, if all the c_i with $k+1 \leq i \leq 2k$ are removed, p will get at least $6k^3 + 3k^2$ votes, which is more than half the votes, so p wins. In both cases there is a "margin" of at least 2, so a single additional vote will not change this. ■

Theorem 10 R_1 satisfies property 2a of Theorem 7 under the Borda protocol. This holds even when there is a single additional arbitrary vote.

Proof: If at least one of the c_i with $k+1 \leq i \leq 2k$ is not removed, consider the highest-indexed remaining candidate among the c_i with $k+1 \leq i \leq 2k$; call it h . The first $12k^3$ votes will put h at least $9k^3 - 3k^2$ points ahead of p . ($12k^3 - 3k^2$ of them rank h above p , and the $3k^2$ others can give p an advantage of at most k each.) The $2k^2$ remaining votes can contribute an advantage to p of at most k each, and it follows that h will still have at least $7k^3 - 3k^2$ more points than p . So p does not win. On the other hand, if all the c_i with $k+1 \leq i \leq 2k$ are removed, then there are two groups of $6k^3 - 3k^2$ among the first $12k^3$ votes which (over the remaining candidates) are each other's exact inverses and hence have no net effect on the scores. Also, the last $2k^2$ votes, which are organized in pairs, have no net effect on the score because (over the remaining candidates) the votes in each pair are each other's exact inverse. The remaining votes all rank p highest among the remaining candidates, so p wins. In both cases the "margin" is big enough that a single additional vote will not change this. ■

Theorem 11 R_1 satisfies property 2a of Theorem 7 under the Maximin protocol. This holds even when there is a single additional arbitrary vote.

Proof: If at least one of the c_i with $k+1 \leq i \leq 2k$ is not removed, then in any pairwise election between such a candidate and p , p will get at most $5k^2$ votes. However, the lowest-indexed remaining candidate among the c_i with $k+1 \leq i \leq 2k$ will get at least $6k^3$ votes in every one of its pairwise elections. So p does not win. On the other hand, if all the c_i with $k+1 \leq i \leq 2k$ are removed, p will get at least $6k^3 + 3k^2$ votes in every one of its pairwise elections, which is more than half the votes; so p wins. In both cases there is a "margin" of at least 2, so a single additional vote will not change this. ■

Theorem 12 R_1 satisfies property 2a of Theorem 7 under the STV protocol. This holds even when there is a single additional arbitrary vote.

Proof: If at least one of the c_i with $k+1 \leq i \leq 2k$ is not removed, consider the lowest-indexed remaining candidate among the c_i with $k+1 \leq i \leq 2k$; call it l . l will hold at least $6k^3$ votes as long as it is not eliminated, and p can hold at most $5k^2$ votes as long as l is not eliminated. It follows that p will be eliminated before l , so p does not win. On the other hand, if all the c_i with $k+1 \leq i \leq 2k$ are removed, p will hold at least $6k^3 + 3k^2$ votes throughout, which is more than half the votes; so p cannot be eliminated and wins. In both cases there is a "margin" of at least 2, so a single additional vote will not change this. ■

Theorem 13 In any of $RPRE + \text{Plurality}$, $RPRE + \text{Borda}$, $RPRE + \text{Maximin}$, and $RPRE + \text{STV}$, CONSTRUCTIVE-MANIPULATION is #P-hard.

Proof: Immediate from the previous theorems. ■

6 PSPACE-hardness when scheduling and voting are interleaved

In this section, we increase the complexity of manipulation one more notch, to PSPACE-hardness,⁵ by interleaving the scheduling and vote elicitation processes.

We first discuss the precise method of interleaving required for our result. The method is detailed and quite complicated. Nevertheless, this does *not* mean that the interleaving should always take place in this particular way in order to have the desired hardness. If the interleaving method used for a particular election is (say, randomly) chosen from a wider (and possibly more naturally expressed) class of interleaving methods containing this one, our hardness result still goes through, as hardness carries over from the specific to the general. Thus, our goal is to find the most specific method of interleaving for which the hardness still occurs, because this gives us the most information about more general methods. We only define the method for the case where the number of candidates is a multiple of 4 because this is the case that we will reduce to (so it does not matter how we generalize the protocol to cases where the number of candidates is not a multiple of 4).

Definition 4 $IPRE$ proceeds as follows:

⁵PSPACE is the class of problems solvable in polynomial space.

1. Label the matchups (a matchup is a space in the pre-round in which two candidates can face each other; at this point they do not yet have candidates assigned to them) 1 through $\frac{|C|}{2}$;
2. For each matchup i , assign one of the candidates to play in it, and denote this candidate by $c(i, 1)$. Thus, one of the candidates in each matchup is known.
3. For some k which is a multiple of 4, for each i with $1 \leq i \leq k$, assign the second candidate to play in matchup i , and denote this candidate $c(i, 2)$. Thus, we have k fully scheduled matchups.
4. For each pair of matchups $(2i-1, 2i)$ with $i > \frac{k}{2}$, assign two more candidates to face the candidates already in these two matchups, and denote them $c((2i-1, 2i), 1)$ and $c((2i-1, 2i), 2)$. (Thus, at this point, all that still needs to be scheduled is, for each i , which of these two faces $c(2i-1, 1)$ and which $c(2i, 1)$.)
5. For $i = \frac{k}{2} + 1$ to $\frac{|C|}{4}$:
 - Randomly decide which of $c((2i-1, 2i), 1)$ and $c((2i-1, 2i), 2)$ faces $c(2i-1, 1)$, and which faces $c(2i, 1)$. Denote the former $c(2i-1, 2)$, the latter $c(2i, 2)$,
 - Ask all the voters whether they prefer $c(i - \frac{k}{2}, 1)$ or $c(i - \frac{k}{2}, 2)$. (We observe that, even if the number of already scheduled matchups is $k = 0$, the elicitation process trails behind the scheduling process by a factor 2.)
6. Elicit the remainder of all the votes.

One important property of this elicitation process is that the voters are treated symmetrically: when a query is made, it is made to all of the voters in parallel. Thus, no voter gets an unfair advantage with regard to knowledge about the schedule. Another important property is that the elicitation and scheduling process at no point depends on how the voters have answered earlier queries. Thus, voters cannot make inferences about what other voters replied to previous queries on the basis of the current query or the current knowledge about the schedule. These two properties guarantee that many issues of strategic voting that may occur with vote elicitation [Conitzer and Sandholm, 2002b] in fact do not occur here.

We are now ready to present our result.

Theorem 14 *Given a voting protocol P , suppose that it is possible, for any Boolean formula ϕ in conjunctive normal form (i.e., a SAT instance) over variables $V = X \cup Y$ with $|X| = |Y|$ (and corresponding literals L), to construct in polynomial time a set of votes over a candidate set containing at least $\{p\} \cup C_L \cup \{c_y^1 : y \in Y\}$ with the following properties:*

- (Property 3a) *If we remove, for each $v \in V$, one of c_{+v} and c_{-v} , p would win an election under protocol P against the remaining candidates if and only if for every clause $k \in K$ (where K is the set of clauses in ϕ), there is some $l \in L$ such that c_l has not been removed, and l occurs in k . This should hold even if a single arbitrary vote is added.*
- (Property 3b) *For any $x \in X$, c_x and c_{-x} are tied in their pairwise election after these votes.*

- (Property 3c) *For any $y \in Y$, c_y and c_{-y} are both losing their pairwise elections against c_y^1 by at least 2 votes (so that they will lose them regardless of a single additional vote).*

Then CONSTRUCTIVE-MANIPULATION in $IPRE + P$ is PSPACE-hard (and PSPACE-complete if P can be executed in polynomial space).

Proof: Consider the following election under $IPRE + P$. Let the candidate set be the set of all candidates occurring in the votes constructed from ϕ (the "original candidates"), plus one dummy candidate for each of the original candidates besides the c_{+v} and c_{-v} . To each of the constructed votes, add all the dummy candidates at the bottom; let the resulting set of votes be the set of the nonmanipulators' votes, according to which they will answer the queries posed to them. The manipulator has yet to decide on its strategy for answering queries. After step 4 (according to Definition 4) of $IPRE + P$ (up to which point the manipulator will not have had to make any decisions), let the situation be as follows:

- The number of already fully scheduled matchups is $k = \frac{|C|}{2} - 2|Y|$. In matchup i ($1 \leq i \leq |X|$), c_{+x_i} faces c_{-x_i} . In the remaining fully scheduled matchups, candidates not corresponding to a literal face a dummy candidate.
- Matchups $k + 2i - 1$ and $k + 2i$ ($1 \leq i \leq |Y|$) already have candidates c_{+y_i} and c_{-y_i} in them, respectively. The other two candidates to be assigned to these rounds are $c_{y_i}^1$ and a dummy candidate.

Thus, what will happen from this point on is the following. For i ranging from 1 to $|X|$, first the protocol will schedule which of c_{+y_i} and c_{-y_i} face which of $c_{y_i}^1$ and the dummy candidate. The c_l facing the dummy will move on, and the other will be defeated by $c_{y_i}^1$, by property 3c. Second, everyone will be asked which of c_{+x_i} and c_{-x_i} is preferred, and because the nonmanipulators will leave this pairwise election tied by property 3b, the manipulator's vote will be decisive. Thus, we can think of this as nature and the manipulator alternately giving the variables in Y and X respectively truth-values: v is set to *true* if c_{+v} survives, and to *false* if c_{-v} survives. By property 3a it then follows that p wins if and only if the resulting assignment satisfies all the clauses, i.e. is a solution to the SAT instance. Thus, the manipulator's strategy for setting variables should aim to maximize the chance of the SAT instance being satisfied eventually. But this is exactly the problem STOCHASTIC-SAT, which is PSPACE-complete [Papadimitriou, 1985].

If P can be executed in polynomial space, the manipulator can enumerate all possible outcomes for all possible strategies in polynomial space, so the problem is also in PSPACE. ■

Because the preconditions of Theorem 14 are similar to those of Theorem 1, we can build on our previous reductions to apply this theorem to the well-known protocols.

Theorem 15 *For each of Plurality, Borda, Maximin, and STV, there exists a reduction that satisfies properties 3a, 3b and 3c of Theorem 14. Thus, In any of $IPRE + \text{Plurality}$, $IPRE + \text{Borda}$, $IPRE + \text{Maximin}$, and $IPRE + \text{STV}$, CONSTRUCTIVE-MANIPULATION is PSPACE-complete.*

Sketch of Proof: We can modify the reductions from Section 4 to satisfy the preconditions of Theorem 14. This is done by adding in the c_y^1 in such a way as to achieve property 3c (ranking them just above their corresponding c_y and c_{-y} in slightly more than half the votes), while preserving property 3a (by ranking them as low as possible elsewhere). ■

7 Conclusions

Voting is a general method for preference aggregation in multiagent systems, but seminal results have shown that all (non-dictatorial) voting protocols are manipulable. One could try to avoid manipulation by using voting protocols where determining a beneficial manipulation is hard computationally. A number of recent papers study the complexity of manipulating existing protocols.

This paper is the first work to take the next step of *designing new protocols that are especially difficult to manipulate*. Rather than designing these new protocols from scratch, we instead showed how to *tweak* existing protocols to make manipulation hard, while leaving much of the original nature of the protocol intact. The tweak studied in this paper consists of adding one preround to the election, where candidates face each other one against one. The surviving candidates continue to the original protocol. Surprisingly, this extremely simple and universal tweak makes typical protocols hard to manipulate! The protocols become NP-hard, #P-hard, or PSPACE-hard to manipulate, depending on whether the schedule of the preround is determined before the votes are collected, after the votes are collected, or the scheduling and the vote collecting are interleaved, respectively. We proved general sufficient conditions on the protocols for this tweak to introduce the hardness, and showed that the most common voting protocols satisfy those conditions. These are the first results in voting settings where manipulation is in a higher complexity class than NP (presuming PSPACE $\neq$ NP).

References

- [Bartholdi and Orlin, 1991] John J. Bartholdi, III and James B. Orlin. Single transferable vote resists strategic voting. *Social Choice and Welfare*, 8(4):341–354, 1991.
- [Bartholdi *et al.*, 1989] John J. Bartholdi, III, Craig A. Tovey, and Michael A. Trick. The computational difficulty of manipulating an election. *Social Choice and Welfare*, 6(3):227–241, 1989.
- [Conitzer and Sandholm, 2002a] Vincent Conitzer and Thomas Sandholm. Complexity of manipulating elections with few candidates. AAAI, pages 314–319, Edmonton, Canada, 2002.
- [Conitzer and Sandholm, 2002b] Vincent Conitzer and Thomas Sandholm. Vote elicitation: Complexity and strategy-proofness. AAAI, pages 392–397, Edmonton, Canada, 2002.
- [Ephrati and Rosenschein, 1991] Eithan Ephrati and Jeffrey S Rosenschein. The Clarke tax as a consensus mechanism among automated agents. AAAI, pages 173–178, Anaheim, CA, 1991.

- [Ephrati and Rosenschein, 1993] Eithan Ephrati and Jeffrey S Rosenschein. Multi-agent planning as a dynamic search for social consensus. IJCAI, pages 423–429, Chambéry, France, 1993.
- [Gibbard, 1973] A Gibbard. Manipulation of voting schemes. *Econometrica*, 41:587–602, 1973.
- [Papadimitriou, 1985] C Papadimitriou. Games against nature. *Journal of Computer and System Sciences*, 31:288–301, 1985.
- [Pennock *et al.*, 2000] David M Pennock, Eric Horvitz, and C. Lee Giles. Social choice theory and recommender systems: Analysis of the axiomatic foundations of collaborative filtering. AAAI, pages 729–734, Austin, TX, 2000.
- [Satterthwaite, 1975] M A Satterthwaite. Strategy-proofness and Arrow’s conditions: existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10:187–217, 1975.
- [Valiant, 1979] Leslie Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.