To appear in *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems* (AAMAS 12), Valencia, Spain, June 2012.

Leading Ad Hoc Agents in Joint Action Settings with Multiple Teammates*

Noa Agmon and Peter Stone
Department of Computer Science
The University of Texas at Austin
{agmon,pstone}@cs.utexas.edu

ABSTRACT

The growing use of autonomous agents in practice may require agents to cooperate as a team in situations where they have limited prior knowledge about one another, cannot communicate directly, or do not share the same world models. These situations raise the need to design *ad hoc* team members, i.e., agents that will be able to cooperate without coordination in order to reach an optimal team behavior. This paper considers the problem of leading N-agent teams by an agent toward their optimal joint utility, where the agents compute their next actions based only on their most recent observations of their teammates' actions. We show that compared to previous results in two-agent teams, in larger teams the agent might not be able to lead the team to the action with maximal joint utility, thus its optimal strategy is to lead the team to the best possible *reachable* cycle of joint actions. We describe a graphical model of the problem and a polynomial time algorithm for solving it. We then consider other variations of the problem, including leading teams of agents where they base their actions on longer history of past observations, leading a team by more than one ad hoc agent, and leading a teammate while the ad hoc agent is uncertain of its behavior.

Categories and Subject Descriptors

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

General Terms

Algorithms

Keywords

Agent Cooperation::Teamwork, coalition formation, coordination ; Economic paradigms::Game theory (cooperative and non-cooperative)

1. INTRODUCTION

Teams of agents have been studied for more than two decades, where the general assumption is that the agents coordinate their actions to increase the team's performance. The growing popularity of agents in domains such as e-commerce, has raised the need for cooperation between agents that are not necessarily programmed similarly, and might not have the same communication protocols or world models. Nevertheless, these agents might be required to perform as a coordinated team. When designing such systems, one cannot assume the team members engage in a known team strategy, but each agent must adjust to the current circumstances while adopting a strategy that aims to optimize the team's utility. Such systems are called *Ad-Hoc Teams*.

In many cases, such as robotic teamwork, it might be impossible to change the design of agents in a team. This work attempts to provide theoretical results (model and solution, and bound on existence of a solution) for the possible influence of a new added agent (one or more) on the team performance. Consider the case where several robots are deployed on Mars; you designed them (thus know their behavior), but once they are there - you cannot re-code them. Suppose that as time passes, you have more knowledge about the world. Will it be worthwhile to send a new robot to change the team behavior to a new, improved, one? If so - how should it do so? These questions motivate our research, and this paper makes progress towards answering them.

Specifically, we consider the problem of leading a team to the optimal joint action. In this problem, the team members do not communicate explicitly, but are assumed to choose their actions based on their observations of their teammates' previous actions (one or more), i.e., the agents behave as *best response agents*.[1] The problem is represented as a simultaneous repeated game. The ultimate goal is to have all team members act in a way that will maximize the joint utility of the team. Assume we design a team member that joins the team, the *ad hoc team member*. Our goal is, therefore, to determine the optimal strategy for the ad hoc team member such that it will lead the team to their optimal joint action while minimizing the system's cost while doing so.

This problem was introduced by Stone *et al.* [11] for systems of two agents: one ad hoc agent, and one best response agent. They describe an algorithm for determining the optimal strategy for the ad hoc agent that leads, in minimal cost, the best response agent to perform the action yielding optimal joint utility. In this paper we consider the more general problem of leading N-agent teams, $N \geq 2$, toward their optimal joint utility. In such systems, the possible

*This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (IIS-0917122), ONR (N00014-09-1-0658), and the FHWA (DTFH61-07-H-00030).

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

[1] We consider best response agents for simplicity, however our results can equally be applied to the more general case of agents that base their decisions on a fixed history window of the ad hoc agents' past actions, rather than on their own previous actions.

influence of one agent on the team is relatively limited compared to the two-agent teams. As a result, we show that in N-agent teams, the optimal joint utility might not be reachable, regardless of the actions of our agent. In such cases, our agent's optimal strategy is to lead the team to the best possible *reachable* joint utility, with minimal cost.

In order to find the optimal strategy for the ad hoc team player, we describe a graphical model of the possible joint set of actions integrating the possible transitions between the system's states (agents' joint action), and the resulting costs of those transitions. Using this model, we first determine the set of joint actions resulting in maximal joint utility, and find the lowest cost path from the initial joint action of the agents to this optimal set of joint actions. We then consider other variations of the problem, and evaluate them using the suggested graphical model. These variations include leading best-response agents with memory size greater than one, leading a team by a team of coordinated ad hoc agents, and leading a two-agent team by an ad hoc agent where uncertainty exists on the behavior of its teammate.

2. PROBLEM DESCRIPTION

We consider the problem of leading a team of best response players by one ad hoc team member towards the optimal joint utility of the team. In this section we describe the general problem, as well as notations that will be used throughout the paper.

The problem of finding a policy for leading team members to the optimal joint utility was introduced in [11] for a team of two agents, A and B, where agent A is the ad hoc agent and agent B is the best response agent. Agent A was designed to lead agent B to perform an action that will result in the optimal joint utility, denoted by m^*. This is done *without using explicit communication or prior coordination*, where agent B chooses an action that maximizes the joint utility based on its observation of agent A's previous action (but with both players having knowledge of the game). This is designed as a simultaneous repeated game, i.e., the players choose their actions simultaneously, where current actions influence the future decisions of the players.

The assumption is that agent B's behavior is known to agent A, but is fixed and unchangeable. This represents one of the simplest cases of ad hoc teamwork, where there is no uncertainty about behaviors. Nevertheless, it poses some interesting challenges, as shown previously in [11], and reinforced in this paper. Relaxation of this assumption is discussed in Section 6.

Agent A has x possible actions $\{a_0, \ldots, a_{x-1}\}$, and agent B has y possible actions $\{b_0, \ldots, b_{y-1}\}$. The team's utility is represented by an $x \times y$ payoff matrix M, where an entry $M(i, j) \in M$ is the joint utility when A performs action a_i and B performs b_j. The *cost* of a joint action $(a_i, b_j), 0 \leq i \leq x-1, 0 \leq j \leq y-1$, denoted by $C(a_i, b_j)$, is defined as $m^* - M(i, j)$, i.e., the distance from the optimal joint utility. The system is initialized in the joint action (a_0, b_0).

It can be assumed, without loss of generality, that m^* is the joint utility obtained when A performs action a_{x-1} and B performs b_{y-1}. Therefore m^* is necessarily reachable from (a_0, b_0) in at most two stages: A picks a_{x-1} and B will adjust in the next stage and choose action b_{y-1}, thus a possible sequence to the optimal joint action m^* is $\langle (a_0, b_0), (a_{x-1}, b_0), (a_{x-1}, b_{y-1}) \rangle$, after which A and B will continue performing actions a_{x-1} and b_{y-1} (respectively). However, this might not be the only possible sequence, and moreover - there could be a sequence of joint actions leading to m^* that has lower cost. The question answered by Stone *et al.* [11] was, therefore, how to reach m^* with minimal cost. They describe a dynamic programming algorithm for finding the optimal solution in polynomial time. Their solution is based on the knowledge of the longest possible optimal sequence, bounding the depth of the recursive algorithm.

In this paper we consider the more general case of leading N-agent teams, $N \geq 2$, by one ad hoc team player. We do so by first examining the problem of leading three-agent teams, and then describe the generalization to N agent teams.

The three-agent team consists of agent A - the ad hoc team member, and the best response agents B and C. The set of actions available for the agents is $\{a_0, \ldots, a_{x-1}\}$, $\{b_0, \ldots, b_{y-1}\}$ and $\{c_0, \ldots, c_{z-1}\}$ for agents A, B, and C, respectively. The payoff matrix of the team is a 3-D matrix M, that can be conveniently written as x matrices of size $y \times z$, $M_0, \ldots, M_{x-1}$ (see Figure 1), where entry (b_i, c_j) in matrix M_k, denoted by $M_k(i, j)$, $(0 \leq k \leq x-1,\ 0 \leq i \leq y-1,\ 0 \leq j \leq z-1)$, is the payoff of the system when agent A performs action a_k, B performs b_i and C performs c_j. Denote the maximal joint payoff in the system by m^*, and assume, without loss of generality, that the agents initially perform actions (a_0, b_0, c_0). Similarly to the two-agent case, the agents do not coordinate explicitly, but agents B and C are assumed to choose their next move according to their current observation of their teammates' actions. Therefore the next action of agent B (denoted by BR_B) is based on its current observation of the actions of agents A and C, and similarly the next action of C (denoted by BR_C) is based on the actions of A and B. Formally, $BR_B(a_i, c_k) = \text{argmax}_{0 \leq j \leq y-1}\{M_i(j, k)\}$ (similarly for BR_C). Therefore if the current actions of the agents are (a_i, b_j, c_k), then the next joint action would be $(a_{i'}, BR_B(a_i, c_k), BR_C(a_i, b_j))$ for $0 \leq i, i' \leq x-1$, $0 \leq j \leq y-1$ and $0 \leq k \leq z-1$.

a_0	c_0	c_1
b_0	5	2
b_1	1	4

a_1	c_0	c_1
b_0	8	6
b_1	4	**20**

Figure 1: An example for unreachable $m^* = (a_1, b_1, c_1)$

Compared to the two-agent team, in a three-agent team the control of agent A on the world is relatively limited. Specifically, there are cases in which m^* remains unreachable, regardless of the actions of agent A. An example for such a case is given in Figure 1. In this example, $x = y = z = 2$. According to these payoff matrices, $BR_B(a_i, c_0) = b_0$ $BR_C(a_i, b_0) = c_0, i \in \{0, 1\}$, thus agents B and C will continue to choose actions (b_0, c_0) in both matrices, and A will not be able to lead them to joint utility of $m^* = M_1(1, 1) = 20$. Therefore the goal of agent A is to lead the team to the *best possible reachable* joint action or cycle of joint actions. In this example, A will choose action a_1, leading to the maximal possible payoff of 8, and all agent will continue choosing the same action yielding maximal possible joint utility.

Definition A *Steady Cycle* is a sequence of t joint actions $s_0, s_1, \ldots, s_{t-1}$ such that if $s_l = (a_i, b_j, c_k)$, then $s_{l+1} = (a_{i'}, BR_B(a_i, c_k), BR_C(a_i, b_j))$, $(0 \leq l \leq t-1,\ 0 \leq i; i' \leq x-1,\ 0 \leq j \leq y-1,\ 0 \leq k \leq z-1)$, and $s_t = s_0$, i.e., the sequence is cyclic. The *Optimal Steady Cycle*, denoted by OSC, is a steady cycle with minimal average cost, i.e, $1/t \times \sum_{i=1}^{t} C(s_i)$ is minimal.

Note that if m^* is reachable, the optimal steady cycle consists of only the joint action yielding payoff m^*.

3. LEADING A TEAM BY A SINGLE AGENT

In this section we examine the problem of leading a team by a single agent, initially concentrating on three-agent teams. We describe a graphical model for representing the system, and a polynomial time algorithm for determining the optimal possible set of joint actions and how to reach it with minimal cost to the team. We later (Subsection 3.4) generalize the representation and solution to an $N-$agent teams.

3.1 Problem Definition

The three-player team consists of three agents: agent A, our designed ad-hoc team player, and agents B and C, which are the original team players.

We define the 3-***Player Lead to Best Response Problem*** (3LBR) as follows.

Given a three-agent team, A, B, C, where agent A is an ad-hoc team player and agents B and C are best response players, and a 3-D payoff matrix representing the team payoff for every joint action of the agents, determine the set of actions of agent A that will lead the team to the optimal steady cycle reachable from (a_0, b_0, c_0) in minimal cost.

3.2 Graphical Representation

In this section we describe a graphical model of the state space, used to find an optimal solution to the 3LBR problem. We create a graph $G = (V, E)$, where V includes of all possible joint actions, i.e., each vertex $v_{ijk} \in V$ corresponds to a set of joint actions (a_i, b_j, c_k) $(0 \le i \le x-1, 0 \le j \le y-1, 0 \le i \le z-1)$. The directed edges in E are defined as follows: an edge $e = (v_{ijk}, v_{i'j'k'}) \in E\ \forall i', 0 \le i' \le x-1$, if $j' = BR_B(a_i, c_k)$ and $k' = BR_C(a_i, b_j)$. In words, an edge is added where it is possible to move from one set of joint actions to the other—either by A repeating the same action $(a_i = a_{i'})$ or by it switching to another action $a_i \neq a_{i'}$. The cost of an edge $e = (v_{ijk}, v_{i'j'k'})$ is set to $m^* - M_{i'}(b'_j, c'_k)$. The total number of vertices in G is xyz, and the number of edges is $x \times |V| = x^2yz$.

Figure 2 illustrates two sets of payoff matrices and their corresponding graphical representations. On the right, m^* is reachable, hence the optimal steady cycle is of size $t = 1$ and includes only v_{111}. The optimal path to the optimal steady cycle is the shortest path (corresponding to the path with lowest cost) between v_{000} to v_{111}, which is $v_{000}, v_{101}, v_{111}$, meaning that the minimal cost sequence is $\langle(a_0, b_0, c_0), (a_1, b_0, c_1), (a_1, b_1, c_1)\rangle$ with a total minimal cost of 21 (there is a "startup cost" of 15 for the first play, that is added to all paths from vertex 000, as indicated by the incoming edge to that vertex). The dashed lines represent the transitions which are determined by A's choices if it changes its action, leading to a change in M_i. The solid lines represent the outcome if A did not change its action.

3.3 Algorithm for Solving the 3LBR Problem

The solution to the 3LBR problem, described in Algorithm 1, is divided into two stages:

1. Find the optimal reachable steady cycle.
2. Find the sequence of actions for agent A that leads the team to the optimal steady cycle with minimal cost.

In order to find the optimal reachable steady cycle, we first remove all vertices that do not belong to the connected component that includes v_{000}. This can be done by a simple BFS tour starting at v_{000} (linear in the graph size). Finding the

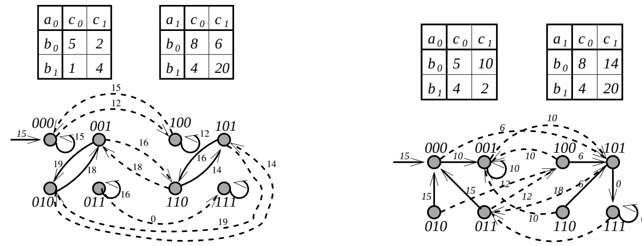

Figure 2: An example for the graphical representation of the state transition. On the left - the representation of the the payoff matrix from Figure 1 where m^* is unreachable, and on the right a case in which m^* is reachable.

optimal steady cycle corresponds to finding the Minimum Cycle Mean (introduced by [8]), that can be computed using dynamic programming in time $\mathcal{O}(|V| \times |E|) = \mathcal{O}(x^3y^2z^2)$.

Finding the sequence of actions taken by agent A that will lead the team to the optimal steady cycle with minimal cost is equivalent to finding the shortest path from v_{000} to any vertex in the cycle yielding the minimum cycle mean of G. Recall that the number of vertices in the cycle yielding the minimum cycle mean of G is denoted by t. Therefore finding the shortest path from v_{000} to one of the vertices of that cycle can be done by Dijkstra's algorithm, resulting in time complexity of $\mathcal{O}(t|E|\log|V|) = \mathcal{O}(tx^2yz\log(xyz))$. The total time complexity of the algorithm is, therefore, $\mathcal{O}(tx^2yz\log(xyz) + x^3y^2z^2)$.

Comparing the time complexity of our algorithm to the algorithm presented by Stone *et al.* [11] for two-player games, we note that finding the optimal sequence of joint actions for a two-player game is a special case of the three-player game in which $z = 1$, and the optimal reachable steady cycle is known to be v_{110}. Thus the time complexity of finding the optimal sequence using our algorithm is $\mathcal{O}(x^2y\log(xy))$, compared to $\mathcal{O}(x^2y)$ of the algorithm described in Stone *et al.* [11]. However, as they have shown, there is no point in returning to a set of joint actions in this scenario, thus while constructing the graph, edges closing a cycle will not be added, yielding a directed acyclic graph (DAG). In DAGs, the shortest path can be found in $\mathcal{O}(|E| + |V|)$ (using topological ordering), therefore the time complexity is similar to the one described in [11], i.e., $\mathcal{O}(x^2y)$.

Algorithm 1 Algorithm Lead3Team(M)

```
Create graph G = (V, E) as follows
for 0 ≤ i ≤ x − 1 ; 0 ≤ j ≤ y − 1 ; 0 ≤ k ≤ z − 1 do
    Add v_{i,j,k} to V
end for
for 0 ≤ i ≤ x − 1 ; 0 ≤ j ≤ y − 1 ; 0 ≤ k ≤ z − 1 do
    for 0 ≤ i' ≤ x − 1 do
        Add e = (v_{i,j,k}, v_{i',BR_B(i,k),BR_C(i,j)}) to E
        Set w(e) = m* − M(i', BR_B(i,k), BR_C(i,j))
    end for
end for
Compute G' ⊆ G by a BFS tour on G starting from v_{0,0,0}
Compute the optimal steady cycle OSC ⊆ G' = {v^0, ..., v^{k−1}}
(Minimum Cycle Mean)
P ← argmin_{0≤i<k} {Shortest path from v_{0,0,0} to v^i ∈ OSC}
```

3.4 Leading N-agent teams

Generalizing 3LBR, we define the N-*Player Lead to Best Response* Problem (NLBR) as follows.

Let $\{a^0, \ldots, a^{N-1}\}$ be a team of N agents, where agent a_0

is an ad-hoc team player. The set of actions for each team member a^i ($0 \leq i \leq N-1$) is $\{a^i_0, a^i_1, \ldots, a^i_{r_i}\}$, and we are given an N-D payoff matrix M representing the team payoff for every combination of actions of the agents. Determine the set of actions of agent a_0 that will lead the team to the optimal steady cycle reachable from $(a^0_0, a^1_0, \ldots, a^{N-1}_0)$ in minimal cost.

In order to generalize the solution to the 3LBR problem to the NLBR problem, it is necessary to define its graphical representation. Once the graphical model is set, finding the optimal solution to the problem becomes similar to the three-agent case, i.e., we find the optimal steady cycle, then we find the shortest path to that cycle. The main difference from the three-agent case lies, therefore, in the creation of the graph, which in turn affects the time complexity.

The graph $G = (V, E)$ is built similarly to the 3-player game, where $v_{i_0 i_1 \ldots i_{N-1}} \in V$ for each set of joint actions $(a^0_{i_0}, a^1_{i_1}, \ldots, a^{N-1}_{i_{N-1}})$ corresponding to an entry in the payoff matrix M_{i_0}, and $e = (v_{i_0 i_1 \ldots i_{N-1}}, v_{i'_0 i'_1 \ldots i'_{N-1}}) \in E$ if $\forall 1 \leq j \leq N-1$, $a^j_{i'_j} = BR_j(a^0_{i_i}, \ldots, a^{j-1}_{i_{j-1}}, a^{j+1}_{i_{j+1}}, \ldots, a^{N-1}_{i_{N-1}})$, $\forall 0 \leq i' \leq r_0 - 1$.

The number of vertices in G is $\prod_{i=0}^{N-1} r_i$, and the number of edges is $r_0 \prod_{i=0}^{N-1} r_i$, leading to a time complexity of $\mathcal{O}(tr_0^2 \prod_{i=1}^{N-1} r_i \log(\prod_{i=0}^{N-1} r_i) + r_0 \prod_{i=0}^{N-1} r_i^2)$ for solving the NLBR problem (t is the length of the optimal steady cycle).

4. LEADING A TEAM WITH MEMORY > 1

Until this point, we have assumed that the teammates optimize their choices based on their most recent observation (best response). We now consider the case in which team members have memory greater than one, i.e., each agent computes its best response to the maximum likelihood distribution corresponding to the last mem joint actions it observed. We describe the analysis for three-agent teams; the solution for the general N-agent team follows directly.

Denote the number of times agent A, B and C performed action a_i, b_j and c_k during the previous set of mem joint actions by N^a_i, N^b_j and N^c_k, correspondingly. Formally, for a three-agent team, the best response of agents B and C are defined (BR_B and BR_C, correspondingly) as follows: $BR_B = \mathrm{argmax}_{\{0 \leq l \leq y-1\}} \{\sum_{i=0}^{x-1} \frac{N^a_i}{mem} \sum_{k=0}^{z-1} \frac{N^c_k}{mem} M_i(l,k)\}$ (BR_C is defined similarly). Let $BR_B(s_1, \ldots, s_{\mathsf{mem}})$ ($BR_C(s_1, \ldots, s_{\mathsf{mem}})$) be the best response of agent B (C) based on the last mem observed joint actions $s_1, \ldots, s_{\mathsf{mem}}$.

The graph representation $G = (v, e)$ in case $\mathsf{mem} > 1$ is as follows. A vertex $v \in V$ represents a global state which includes a sequence of mem consecutively executed joint actions, $\{s_0, \ldots, s_{\mathsf{mem}}\}$. An edge $e = (v, u) \in E$ exists from a vertex $v = \{s_0, \ldots, s_{\mathsf{mem}}\}$ to vertex $u = \{s'_0, \ldots, s'_{mem}\}$ if $\forall 0 \leq l \leq \mathsf{mem} - 2; \forall 0 \leq i \leq x-1$, $s'_l = s_{l+1}$, $s'_{\mathsf{mem}} = \{a_i, BR_B(s_0, \ldots, s_{\mathsf{mem}}), BR_C(s_0, \ldots, s_{\mathsf{mem}})\}$.

As shown by Stone *et al.* [11], even if m^* was played once, it does not necessarily mean that the system will stay there. Specifically, it could be the case that the team was lead to m^*, however the next best response of some agent (one or more) would be to switch actions. This was denoted as *unstable states.* Assume the joint action yielding m^* is (a^*, b^*, c^*). As a result, we define the terminal vertex of the system to be $\langle (a^*, b^*, c^*), \ldots, (a^*, b^*, c^*) \rangle$ (mem times). Clearly, this vertex is stable.

4.1 On the time complexity of handling high memory

Finding the optimal steady cycle and the optimal path to that cycle in case the team members have memory size greater than one can be computed similarly to the solution to the 3LBR and the NLBR problems (Algorithm Lead3Team). In order to determine the time complexity of reaching an optimal solution, it is necessary to calculate the size of the graph representation, i.e., $|V|$ and $|E|$. The number of combinations of mem sets of joint actions is $|V|^{\mathsf{mem}}$. However, not all combinations of sets of joint actions are feasible (the system cannot reach every vertex from every vertex, but only x vertices from each vertex), hence the upper bound on the number of vertices is $xyz \times x^{\mathsf{mem}-1}$, i.e., $x^{\mathsf{mem}} yz$ (exponential in mem). The number of edges from each vertex remains x, hence the total number of edges is $x^{\mathsf{mem}+1} yz$.

This provides an *upper bound* on the time complexity of reaching a solution with $\mathsf{mem} \geq 2$. However, we would like to examine whether this bound is tight or not, i.e., can we guarantee that the time complexity will practically be lower than that? We do so by pursuing two different directions. First, we check whether there is a connection between the relevant graph size (connected component that includes the initial vertex) with teammates having memory of size $\mathsf{mem} - 1$ and the relevant graph size when their memory is mem. For example, if the connected component only got smaller as memory increased, then we could bound the graph size by the size of the connected component when mem=1. Second, we check whether we can efficiently bound the possible length of the optimal path from the initial joint action to the optimal (cycle of) joint action(s). If so, that would allow us to restrict the construction of our state space to states within that maximal length from the initial state. Unfortunately, the investigations in both directions are not promising. We show that there is no connection between the size of the relevant connected component as the memory size grows (it could increase or decrease). We also show that even in the simplest case of $N = 2$ and $\mathsf{mem} = 2$ determining the maximal size of an optimal path from $v_{0,0}$ to m^* is $\mathcal{NP}$-Hard.

4.1.1 *Graph size as memory size changes*

We investigated the influence of the number of vertices in the connected component in case $\mathsf{mem} = 1$ to the number of components when $\mathsf{mem} = 2$ in order to determine a tight bound on the number of vertices we need to explore as the memory grows. Unfortunately, we have shown that there is no relation between the number of vertices in the connected components between different memory sizes. In this section, we describe two opposite cases: one in which the connected component grows, and one in which it becomes smaller.

We show, by the following example, that the connected component originating at $v = (0,0,0)$ with $\mathsf{mem} = 1$ can grow as mem grows to include vertices corresponding to joint actions that were unreachable with smaller memory size. Moreover, Figure 3 shows a case in which with $\mathsf{mem} = 1$ m^* is unreachable, yet with $\mathsf{mem} = 2$ it becomes reachable. On the other hand, as shown in Figure 4, the number of reachable joint actions may decrease, possibly causing m^* to become unreachable.

These two examples demonstrate that a tight bound on the number of vertices that need to be explored as memory grows does not exist, i.e., it is not sufficient to explore only

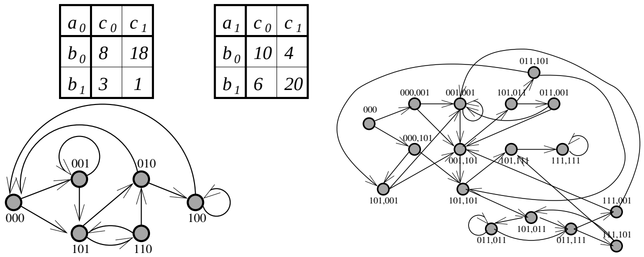

Figure 3: An example for a case where m^* was unreachable for mem $= 1$ (left), and became reachable with mem $= 2$ (right).

the main connected component of mem $= 1$, but it is necessary to explore the entire main connected component in the current memory model.

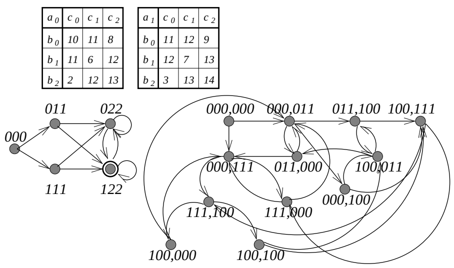

Figure 4: An example for a case where m^* was reachable for mem $= 1$ (left), and became unreachable with mem $= 2$ (right).

4.1.2 $\mathcal{NP}$-Hardness of maximal optimal length determination

In order to evaluate whether the graph size can be reduced to guarantee a more realistic upper bound on the time complexity that is not exponential in the graph size, we examined the possibility of limiting the construction of the graph based on the maximal possible length of an optimal path from the initial vertex to the optimal steady cycle. In [11] it was shown that in two agent teams, consisting of one ad hoc team member (agent A) and one best-response team member (agent B), if mem $= 1$ then the maximal size of an optimal path is at most $\min\{x, y\}$. However, we prove here that even in this simple case where m^* is known to be reachable (i.e., the optimal steady cycle is known in advance), determining the maximal size of an optimal path is $\mathcal{NP}$-Hard when agent B has mem ≥ 2 (note that this was assumed in [11], yet was not proven there).[2]

Denote the maximal length of an optimal path starting at (a_0, b_0) to m^* by S^* (note that since it is the two agent game, such a path always exists).

THEOREM 1. *In the two-agent case, finding S^* when* mem ≥ 2 *is $\mathcal{NP}$-Hard.*

PROOF. The problem is proven to be NP-hard by a reduction from the Hamiltonian Path problem:Given an n-node unweighted, undirected graph G, an initial node and a destination node, is there a simple path from initial to destination of length n? That is, can we visit each node exactly once? This decision problem is known to be $\mathcal{NP}$-Complete [4].

[2] We thank ML for his help in formalizing the proof

We will show that if it were possible to find S^* for a given matrix M with agent B's mem > 1 in polynomial time, then it would also be possible to find a Hamiltonian path in polynomial time. To do so, we assume that we are given a graph $G = (V, E)$ such that $|V| = n$. We construct a matrix M in a particular way such that if there is a path through the matrix of cost no more than a target value of $n*(n^4-1)$, then there is a Hamiltonian Path in graph G. Note that, as required, the construction of the matrix can be done in time polynomial in all the relevant variables.

Let agent B's mem $= n$. We construct the joint payoff matrix M as follows.

- Agent A has $(n-1)*n+2$ actions. The first action is "start", and agent B's memory is initialized to n copies of that action. Each of the next $(n-1)*n$ actions represents a combination (i, t) of a vertex $v_i \in V(G)$ and a time step $t \geq 2$. M's payoffs will be constructed so that if the sequence satisfying the maximum cost requirement in M (if any) includes action (i, t), then the corresponding Hamiltonian path passes through v_i on time step t. Finally, there is a "done" action to be taken at the end of the path.
- Agent B has $n^2 + n + 1$ actions. The first n^2 actions are similar to agent A's: one for each combination of $v_j \in V(G)$ and $t \geq 1$. If the satisfying sequence through M includes agent B taking action (j, t), then the Hamiltonian path visits v_j at time t. The next n actions are designed as "trap" actions which agent B will be induced to play if agent A ever plays two actions corresponding to the same node in the graph: actions (i, s) and (i, t). There is one trap action for each vertex, called action j. Finally, the last action is the "done" action to be played at the end of the sequence.
- M's payoffs are constructed as follows, with the actions named as indicated in the bullets above. The initial vertex in the Hamiltonian path (the one visited on time step 1) is called "initial."

$$
\begin{array}{llcll}
\text{a)} & M[(i,t+1),(j,t)] & = & 1 & \text{if } (v_i,v_j) \in E(G) \\
\text{b)} & M[(i,t+1),(j,t)] & = & -n^5 & \text{if } (v_i,v_j) \notin E(G) \\
\text{c)} & M[(i,t),(i,t)] & = & tn & \\
\text{d)} & M[(i,t),(j,s)] & = & -n^5 & \text{if } t \geq s \\
\text{e)} & M[(i,t),(j,s)] & = & 0 & \text{if } t < s \\
\text{f)} & M[(i,t),i] & = & tn - \frac{1}{3n} & \\
\text{g)} & M[(i,t),j] & = & 0 & \\
\text{h)} & M[(i,t),\text{done}] & = & 0 & \\
\text{i)} & M[\text{start},(\text{initial},1)] & = & 1 & \\
\text{j)} & M[\text{start},\text{initial}] & = & \frac{1}{2} & \\
\text{k)} & M[\text{start},\text{done}] & = & -n^4 & \\
\text{l)} & M[\text{start},j] & = & 0 & \forall \text{ action } j \notin \{\text{initial,done}\} \\
\text{k)} & M[\text{done},(j,n)] & = & 1 & \\
\text{l)} & M[\text{done},(j,t)] & = & -n^5 & \text{if } t < n \\
\text{m)} & M[\text{done},\text{done}] & = & n^4 &
\end{array}
$$

Following a path through the matrix that corresponds to a Hamiltonian path (if one existed) would give payoffs of 1 at every step until reaching m^* (n^4) and staying there forever. Thus the cost of the n-step path would be $n*(n^4-1)$.

As there is no positive payoff in the matrix greater than n^2, any path longer than n steps must have cost of at least $(n+1)(n^4-n^2) = n^5+n^4-n^3-n^2 > n^5-n = n*(n^4-1)$. In other words, if there is a path through the matrix corresponding to a Hamiltonian path in the graph, then any longer path through M must have higher cost.

Furthermore, the matrix is carefully constructed such that any diversion from the path corresponding to a Hamiltonian path either will get a payoff of $-n^5$ on at least one step (which by itself makes the target cost impossible to reach), will prevent us from getting one of the 1's, or else will make it so that the path to (done,done) will require more than n total steps. In particular, if agent A ever takes two actions that lead agent B to select a trap action, then agent B will not take a different action until the $n+1$st step after the first action that led to the trap, causing the path to (done,done) to be at least $n+2$ steps long. Therefore, if we could find the optimal sequence through any matrix in polynomial time, then we could use this ability to also solve the Hamiltonian path problem, concluding our proof. $\square$

5. LEADING A TEAM BY A TEAM

Until now, we have considered the case of one ad-hoc agent leading a team of best response agents towards the optimal set of joint actions. However, it could be possible to deploy a *team* of coordinated ad-hoc agents to lead the best response agents. The two interesting questions that arise here are: (a.) What is the time complexity of determining the optimal path? (b.) What is the influence of the addition of a new team member to the group with respect to the optimal steady cycle and the cost of the path towards it?

Also in this case we adopt the graphical model in order to find the optimal steady cycle, and the shortest path towards it. Let $\{a^0, \ldots, a^{N-1}\}$ be a team of N agents, where agents $a^0, \ldots a^{k-1}$ are the ad-hoc team players, and $a^k, \ldots, a^{N-1}$ are the best response team members. Each agent a^i has a set of possible actions $\{a^i_0, \ldots, a^i_{r_i}\}$. Therefore the number of possible joint actions (hence the number of vertices in the representing graph), similar to the $N-$agent teams discussed in Section 3.4, is $\prod_{i=0}^{N-1} r_i$. The difference between the $N-$agent teams led by one agent and the $N-$agent teams led by k agents lies in the edges in the graph. Formally, the graph $G = (V, E)$ representing the system is built as follows. A vertex $v_{i_0 i_1 \ldots i_{N-1}} \in V$ exists for each joint action $(a^0_{i_0}, \ldots, a^{N-1}_{i_{N-1}})$. An edge $e = (v_{i_0 \ldots i_{N-1}}, v_{i'_0 \ldots i'_{N-1}}) \in E$ if $\forall k \leq j \leq N-1$, $a^j_{i'_j} = BR_j(a^0_{i_i}, \ldots, a^{j-1}_{i_{j-1}}, a^{j+1}_{i_{j+1}}, \ldots, a^{N-1}_{i_{N-1}})$, $\forall 0 \leq i'_l \leq r_{l-1}; 0 \leq l \leq k-1$. Therefore the number of outgoing edges from each vertex is $\prod_{i=0}^{k-1} r_i$, since each ad hoc team member's choice of action influences the possible outcome, hence $|E| = \prod_{i=0}^{k-1} r_i^2 \prod_{j=k}^{N-1} r_j$. An example for a graphical representation of leading a team by more than one agent is shown in Figure 5.

Finding the optimal steady cycle is done on this graph as described previously. The time complexity of determining the optimal steady cycle is $\mathcal{O}(|E| \times |V|) = \mathcal{O}(\prod_{i=0}^{k-1} r_i^3 \prod_{j=k}^{N-1} r_i^2)$. Assuming the number of joint actions in the steady cycle is $t \geq 1$, finding the optimal path from the initial vertex to the optimal steady cycle is done in time $\mathcal{O}(t|E| \log(|V|)) = \mathcal{O}(t \prod_{i=0}^{k-1} r_i^2 \prod_{j=k}^{N-1} r_i \log(\prod_{i=0}^{N-1} r_i))$

Clearly, as more agents are involved in leading the team, their influence on the outcome (the optimal reachable steady cycle) is higher relative to the case of a single leading agent. For example, in Figure 1, if agents B and C were leading the team, then m^* becomes reachable. However, in the general case, having more than one leader still might not have the power to lead the system towards m^*. Consider, as an example, the case in which $k = 2$ and $N = 4$, i.e., two agents lead the other two agents towards the optimal steady cycle. The payoff matrix can be considered as $r_0 \times r_1$ matrices of size $r_2 \times r_3$. If each of these matrices is, for example, a replica

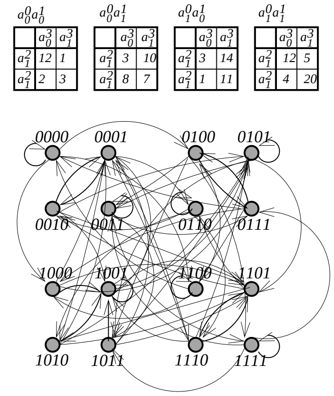

Figure 5: An example for the graphical representation where there is more than one ad hoc agent. The team has 4 agents: 2 ad hoc (a^0 and a^1) and 2 best response (a^2 and a^3), each having 2 actions.

of the matrix described in Figure 1, then agents a^0 and a^1 will never be able to lead the team to m^* (unless starting at m^*), as regardless of their actions, a^2 and a^3 will never jointly choose actions a^2_1 and a^3_1 (respectively).

This example leads to the following corollary:

Corollary 2. *In a team of N agents, for every $N \geq 3$, where at least two team members are best response agents, m^* may remain unreachable.*

6. LEADING WITH UNCERTAINTY

It is possible, perhaps likely, that our designed agent will be uncertain about the behaviors of its teammates. We consider in this section a two-agent team, agents A and B, where agent A is an ad hoc agent, and agent B can be either an ad hoc agent or a best response agent. Clearly, if B is an ad hoc agent and the agents are fully coordinated, then their choice of actions is well defined—both will choose together the joint action with utility m^*. Similarly, in larger teams where there is more than one ad hoc agent and they are coordinated, then the solution is similar to the one described in Section 5. However, uncertainty may arise when, for example, communication fails, and agent A has to decide which action to take in order to lead the team to m^* (recall that in the two-agent team case, m^* is always reachable). In cases where there exists a probability distribution over the possible identities of the teammate, the ad hoc agent can choose actions that result in a minimal cost path towards m^*, and also possibly choose actions that will reveal the true identity of the teammate (if necessary).

If B is a best response agent, then like in previous sections (and in [11]) it assumes that A will continue performing its previous action, and given that action it will choose an action maximizing the joint utility. If B is an ad hoc agent, we assume that it will act in one of two ways: it will either believe that A is an ad hoc player, thus will choose action b_{y-1} (assuming A will choose a_{x-1}, reaching m^* immediately), or it will assume that A is a best response player, and choose its actions appropriately. Modeling the identity of the other agent can be done recursively infinitely (what A believes B will do, that believes what A will do, and so on), however we are motivated by RMM [5] in modeling only a recursion of depth two (note that although RMM does recommend a recursive depth 2, it is usually shown to be beneficial to model a bounded depth, and we leave the general discussion of depth in this setting to future work). Therefore we model three possible cases: (*i*) B is a best response agent, denoted by br ; (*ii*) B is an ad hoc agent, believing that A is an ad hoc agent, denoted by ah/ah; (*iii*) B is an ad hoc agent believing that A is a best response agent, denoted by ah/br.

In order to determine the best action for our agent A, we build a graph for each identity of the teammate. Note that since the vertices of the graphs are identical, we can use one graph with three different types of edges: E_i for br agent, E_{ii} for ah/ah agent and E_{iii} for the ah/br agent.

Generally, when uncertainty arises with respect to the environment (in our case the identity of the teammate), it is common to gain more information about the environment while exploring it. In our case, we need agent A to both gain information about B, and to allow agent B to reevaluate its belief about A. Specifically, if agent B is ah/br, we want it to realize that A is *not* a best response agent, thus allowing the team to reach m^* more efficiently. This happens if A diverts from the expected best response behavior. Note that if agent B is an ah/ah agent, it will choose to perform action b_{y-1} even after realizing that A did not perform a_{x-1}, since diverting from that action will necessarily cause the path to m^* to be longer (even in the case where A is best response, its next action would be a_{x-1}, leading to m^*).

As a first step we construct E_i, E_{ii} and E_{iii}. E_i is constructed as in Section 3. E_{ii} and E_{iii} can theoretically be complete graphs, however given the beliefs of B, most of the edges will not exist. Since B believes A to be an ad hoc agent, it will choose action b_{y-1}, thus E_{ii} includes x edges from v_{00} to $v_{i(y-1)}$, $\forall 0 \leq i \leq x-1$, and $x-1$ more edges from $v_{i(y-1)}$ to m^*. E_{iii} is constructed as follows. The only outgoing edge out of v_{00} is to some v_{ij} such that $a_i = BR_A(b_0)$, and $b_j \in SP_B(v_{00}, m^*)$, where SP_B denotes the shortest path from v_{00} to m^*, calculated by B (according to the weights and path that are calculated similarly to the description in Section 3). In addition, there are $x-1$ edges from v_{00} to v_{kj}, $0 \leq k \leq x-1$. From now on, for each vertex v, if there is no incoming edge from some vertex u such that v is the best response of u with respect to A's behavior as a best response agent, then we add an edge (v, m^*), otherwise the only edges added are from v to v_{ij}, $0 \leq i \leq x-1$, $b_j \in SP_B(v_{00}, m^*)$ (we omit the formal algorithm due to space constraints). Choosing the shortest path from v_{00} to m^* will determine whether it is more efficient to choose an edge that in the short term might not be beneficial, but will reveal enough information to the teammate that will force it to realize it is teamed with an ad hoc agent, making it choose a shorter path towards m^*.

Given the constructed edges, one can compute the shortest path along each set of edges, starting from v_{00} towards m^*. Each of these path has a total cost, and given a probability distribution over the possible identities of the teammates, the first action is chosen such that the expected cost is minimized. If a_{x-1} is chosen, then the path towards m^* is easily determined (regardless of the true identity of B). If it was not chosen, then A follows the path it chose, while making adjustments throughout the execution. Namely, if it chose an action that will teach B that it's an ad hoc agent, yet B acts as a best response (rather than ad hoc), it will find the shortest path towards m^* according to E_i (as it is clear that B is best response). Otherwise, it will follow the shortest path by E_{iii} towards m^*.

	b_0	b_1	b_2	b_3	b_4
a_0	10	15	0	0	0
a_1	0	16	17	0	5
a_2	0	0	18	19	15
a_3	0	0	0	20	21
a_4	0	0	0	0	22

Figure 6: Example for the advantage of relying on expected cost.

Clearly, if A assumes some identity of B and it is not mistaken, then this yields the best possible path towards m^*. Consider the example in Figure 6. If A prepares for ah/ah and is teamed with an ah/ah, the cost of path towards m^* is 0. If it expects B to be ah/br and B is indeed ah/br, the cost of the path is 6. Finally, if it expects B to be br and is indeed teamed with a br agent, then the cost is 12. Being wrong in the identity of the teammate is costly. Note that no matter if A expects a weak opponent and is teamed with a strong one or the opposite - the consequences (in terms of path cost) are high. In this example (considering a uniform probability distribution between the identities) assuming a br teammate has minimal expected cost of path towards m^*: assuming an ah/ah teammate has expected cost of 14.6, assuming an ah/br teammate has expected cost of 18.6 and br has expected cost of 14. Therefore, according to the algorithm, B should be assumed to be a br agent. This choice of action yields a *worst case* cost of mistaken identity of 17 (if B is actually ah/ah). On the other hand, if we would not have consulted the algorithm and chosen to believe that B is ah/ah, the worst cost of mistaken identity would be 22, and similarly ah/br yields worst cost of 28 upon mistaken identity. Note that generally even if the maximal cost of mistaken identity is not higher in the unchosen belief, still the expected cost is lower, thus in the average case, by using this algorithm, A minimizes its cost.

When leading a team with more than one agent where one of them might be an ad hoc agent, the identity of the teammate can be crucial, and determines whether m^* is reachable or not. This case is broad and complicated and is thus left out of the scope of this paper.

7. RELATED WORK

Stone *et al.* [10] introduced a formalism for *Ad-Hoc teamwork*, which deals with teamwork behavior without prior coordination. They raised a challenge *"To create an autonomous agent that is able to efficiently and robustly collaborate with previously unknown teammates on tasks to which they are all individually capable of contributing as team members"*. This paper answers one aspect of the challenge raised there, namely leading teams of agents, with no a-priori coordination and explicit communication to the optimal possible joint-utility, in a simultaneous-action setting.

Bowling and McCracken [1] suggested two techniques for incorporating a single agent into an unknown team of existing agents: adaptive and predictive. In their work, they are concerned with the task allocation of the agent (which role should it choose, and what is its teams' believed behavior), where their agent might adapt its behavior to what it observes by the team. Jones *et al.* [7] examined the problem of team formation and coordination without prior knowledge in the domain of *treasure hunt*. They considered a team composed of heterogenous robots, each with different capabilities required for various aspects of searching an unknown environment and extracting a hidden treasure. Their architecture was based on role selection using auctions. In contrast to these approaches, in our work we examine how our agent can influence the behavior of the team by leading the team to an optimal behavior.

Stone and Kraus [12] considered the problem of ad hoc teamwork by two agents, agent A (also known as the teacher), and agent B in the k-armed bandit problem. The question they asked was: Assuming that agent B observes the actions of agent A and its consequences, what actions should agent A choose to do (which bandit to pull) in order to maximize the team's utility. It was shown that in some cases, agent A should act as a teacher to agent B by pulling a bandit that will not yield optimal immediate payoff, but will result in teaching agent B the optimal bandit it should pull. In our work we also control the actions of agent A, but the payoff is determined by the joint actions of the team players, not by individual actions of each teammate.

Han *et al.* [6] examined a closely related problem of controlling the collective behavior of self-organized multi-agent system by one agent. They consider self organized teams of physically interacting agents, concentrating on flocking of birds, where their goal is to design an agent, denoted as a *shill agent*, that will be able to gradually change the heading of the entire team to a desired heading. They evaluate the system in terms of physical capabilities of the shill agent and the team (velocity, initial heading) and provide theoretical and simulation results showing that it is possible, under some conditions, for one agent to change the heading of the entire team. Different from out approach, they do not consider game theoretic evaluation of the individual actions and their impact on the team behavior.

Young [13] introduced the notion of *adaptive games*, where N agents base their current decisions on a finite (small) horizon of observations in *repeated games*, and search for agents' actions yielding a stochastically stable equilibrium using shortest paths on graphs. In our work, we do not assume the agents play repeatedly (allowing to adjust to a strategy), but we aim to guarantee that our agent leads the team to the optimal possible joint action(s) while minimizing the cost paid by the team along the way.

Numerous research studies exist in the area of normal form games, where the agents' payoffs are described in a matrix (similar to our case) and depend on the chosen joint actions. In the normal form games framework, a related topic is the problem of learning the best strategy for a player in repeated games. Powers and Shoham [9] considered the problem of normal form games against an adaptive opponent with bounded memory. Chakraborty and Stone [2] examine optimal strategies against a memory bounded learning opponent. Cho and Kreps [3] introduced signaling games, a sequential Bayesian game in which a player chooses its actions based on messages transferred from a second player with unknown type. Our work is inherently different from these approaches, since in our case the agents are collaborating as a team, hence they aim to maximize the *joint* payoff and not the individual payoff, which raises different questions and challenges as for the optimality of the joint action and the way to reach this optimal joint action.

8. CONCLUSIONS AND FUTURE WORK

In this paper we examine the problem of leading a team of $N \geq 2$ agents by one or more ad hoc team members to the team's joint actions yielding optimal payoff. We show that it may not be possible to lead the team to the optimal joint action, and offer a graphical representation of the system's state and a polynomial time algorithm that determines the optimal *reachable* set of joint actions, and finds the path with minimal system cost to that set. We examine the case in which the team members base their next action on more than one previous joint action, describe an algorithm—using the same graphical representation—that calculates the optimal strategy for the ad hoc team member in time exponential in the teammates' memory size, and show that it is not likely that there exists an algorithm that solves the problem in better time complexity. We further use the graphical representation to consider the case in which the ad hoc agent is teamed with more than one ad hoc agent in coordination, and also the case in which it might be teamed with one agent with uncertain nature—ad hoc or best response.

There are various directions to be addressed as future work. First, the question of uncertainty can be examined not only in the identity of the agent, but also in many other aspects, such as the knowledge of the payoff matrix, namely, the ad hoc agent might not have full knowledge of the payoff matrix, but some incomplete knowledge or a probability distribution over possible values. Similar uncertainty may exist in the knowledge of the best response agents on the payoff matrix, which might result in nondeterministic choice of actions. Uncertainty may exist also in the type of teammates, acting not necessarily as best response agents, but adapting other (known or unknown) behaviors.

9. REFERENCES

[1] M. Bowling and P. McCracken. Coordination and adaptation in impromptu teams. In *Proc. of AAAI'05*, pages 53–58, 2005.

[2] D. Chakraborty and P. Stone. Online multiagent learning against memory bounded adversaries. In *Machine Learning and Knowledge Discovery in Databases*, volume 5212 of *Lecture Notes in Artificial Intelligence*, pages 211–26, September 2008.

[3] I. Cho and D. M. Kreps. Signaling games and stable equilibria. *The Quarterly Journal of Economics*, 102(2):179–221, 1987.

[4] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., 1990.

[5] P. J. Gmytrasiewicz, E. H. Durfee, and D. K. Wehe. A decision-theoretic approach to coordinating multiagent interactions. In *IJCAI*, pages 62–68, 1991.

[6] J. Han, M. Li, and L. Guo. Soft control on collective behavior of a group of autonomous agents by a shill agent. *Systems Science and Complexity*, 19(1), 2006.

[7] E. Jones, B. Browning, M. Dias, B. Argall, M. Veloso, and A. Stentz. Dynamically formed heterogeneous robot teams performing tightly-coordinated tasks. In *Proc. of ICRA'06*, pages 570 – 575, 2006.

[8] R. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23, 1978.

[9] R. Powers and Y. Shoham. Learning against opponents with bounded memory. In *Proc. of IJCAI'05*, pages 817–822, 2005.

[10] P. Stone, G. Kaminka, S. Kraus, and J. Rosenschein. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Proc. of AAAI'10*, 2010.

[11] P. Stone, G. A. Kaminka, and J. S. Rosenschein. Leading a best-response teammate in an ad hoc team. In *Agent-Mediated Electronic Commerce: Designing Trading Strategies and Mechanisms for Electronic Markets (AMEC)*, 2010.

[12] P. Stone and S. Kraus. To teach or not to teach? decision making under uncertainty in ad hoc teams. In *Proc. of AAMAS'10*, 2010.

[13] P. Young. The evolution of conventions. *Econometrica*, 61(1):57–84, 1993.