Leading Multiple Ad Hoc Teammates in Joint Action Settings

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 problem of leading N-agent teams by a single 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. In these cases, the agent's optimal strategy leads 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 the problem of leading teams where the agents' base their actions on a longer history of past observations, showing that the an upper bound computation time exponential in the memory size is very likely to be tight.

Introduction

Teams of agents have been studied for more than two decades, where generally the assumption is that the agents can 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 existing agents on the 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 on the team performance. Consider the case where several robots are deployed on Mars; you designed them, 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.

In this paper 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 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 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, Kaminka and Rosenschein (2010) 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 influence of one agent in 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

[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.

the lowest cost path from the initial joint action of the agents to this optimal set of joint actions.

We then consider the problem of leading best-response agents with memory size greater than one. In such cases, the teammates compute their best response to the maximum likelihood distribution corresponding to the last joint actions they observed (in the size of their memory). We discuss the influence of the memory size on the time complexity of determining the optimal strategy for the ad hoc team member. Specifically, the upper bound on the time complexity of our suggested algorithm is exponential in the memory size. Unfortunately, we show that this upper bound is very likely to be tight, based on the size of the state space and the fact that determining the length of the optimal path even in the simple case of two agents and memory size 2 is $\mathcal{NP}$-Hard.

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 (Stone, Kaminka, and Rosenschein 2010) 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 to perform 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).

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 in this paper.

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 the matrix is the joint utility when A performs action a_i and B performs action b_j. The *cost* of each possible 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.* (2010) 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 a case in which $m^* = (a_1, b_1, c_1)$ is unreachable.

Compared to the two-player game, in the three-agent team the control of agent A on the game is relatively limited. Specifically, there are cases in which m^* will remain 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^*.

Leading a three-agent team

In this section we examine the problem of leading a three-agent team. We describe a graphical model for representing the system, and a polynomial time algorithm for determining the optimal possible joint action (cycle) and how to reach it with minimal cost to the team.

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.

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) $(0 \leq i \leq x-1, 0 \leq j \leq y-1, 0 \leq i \leq z-1)$. The directed edges in E are defined as follows: an edge $e = (v_{ijk}, v_{i'j'k'}) \in E \; \forall i', 0 \leq i' \leq x-1$, 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 a set of payoff matrices and its corresponding graphical representation. In this case, 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, leading to a change in M_i. The solid lines represent the outcome if A did not change its action.

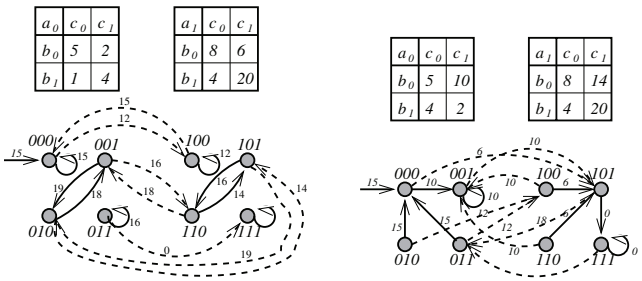

Figure 2: An example for the graphical representation of the state transition. On the left - the representation of the the payoff matrix from Figure 1 where m^* is unreachable.

Algorithm for Solving the 3LBR Problem

We divide the solution to the 3LBR problem into two stages:

1. Find the optimal reachable steady cycle.
2. Find the sequence of actions for agent A that will lead 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 optimal steady cycle corresponds to finding the Minimum Cycle Mean (introduced by (Karp 1978)), 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.* (2010) 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 $z = 1$ (thus is irrelevant) 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.* (2010). 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 (Stone, Kaminka, and Rosenschein 2010), i.e., $\mathcal{O}(x^2y)$.

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

Algorithm 1 Algorithm Lead3Team(M)

1: Create graph $G = (V, E)$ as follows
2: **for** $0 \leq i \leq x-1$; $0 \leq j \leq y-1$; $0 \leq k \leq z-1$ **do**
3: Add $v_{i,j,k}$ to V
4: **for** $0 \leq i \leq x-1$; $0 \leq j \leq y-1$; $0 \leq k \leq z-1$ **do**
5: **for** $0 \leq i' \leq x-1$ **do**
6: Add $e = (v_{i,j,k}, v_{i',BR_B(i,k),BR_C(i,j)})$ to E
7: Set $w(e) = m^* - M(i', BR_B(i,k), BR_C(i,j))$
8: Compute $G' \subseteq G$ by a BFS tour on G starting from $v_{0,0,0}$
9: Compute the optimal steady cycle OSC $\subseteq G' = \{v^0, \ldots, v^{k-1}\}$ (Minimum Cycle Mean)
10: $P \leftarrow \text{argmin}_{0 \leq i < k}$ {Shortest path from $v_{0,0,0}$ to $v^i \in$ OSC}

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 $\mathcal{O}(t r_0^2 \prod_{i=1}^{N-1} r_i \log(\prod_{i=0}^{N-1} r_i) + r_0 \prod_{i=0}^{N-1} r_i^2)$ for solving the NLBR problem (where t is the length of the optimal steady cycle).

Leading a team with memory> 1

Until this point, we assumed that the teammates optimize their choices based on their most recent observation (best response). We now consider the case in which the team members have memory greater than one, i.e., each team member 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:
$BR_B = \text{argmax}_{\{0 \leq l \leq y-1\}} \{\sum_{i=0}^{x-1} \frac{N^a_i}{mem} \sum_{k=0}^{z-1} \frac{N^c_k}{mem}(a_i, b_l, c_k)\}$ (BR_C is defined similarly). Denote the best response of agent B (C) based on the last observed *mem* sets of joint actions $s_1, \ldots, s_{mem}$ by $BR_B(s_1, \ldots, s_{mem})$ ($BR_C(s_1, \ldots, s_{mem})$).

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

As shown by Stone *et al.* (2010), 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.

On the time complexity of handling high *mem*

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 $|V|^{mem}$. 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 $xyz \times x^{mem-1}$, i.e., $x^{mem}yz$ (exponential in *mem*). The number of edges from each vertex remains x, hence the total number of edges is $x^{mem+1}yz$.

This provides an *upper bound* on the time complexity of reaching a solution with $mem \geq 2$. 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 $mem - 1$ 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 $mem = 2$

determining the maximal size of an optimal path from $v_{0,0}$ to m^* is $\mathcal{NP}$-Hard.

Graph size as memory size changes We investigated the influence of the number of vertices in the connected component in case *mem* $= 1$ to the number of components when *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 *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 *mem* $= 1$ m^* is unreachable, yet with *mem* $= 2$ it becomes reachable. On the other hand, as shown in Figure 4, the number of vertices may decrease, possibly causing m^* to become unreachable.

These two examples set a tight bound on the number of vertices that need to be explored as memory grows, i.e., it is not sufficient to explore only the main connected component of *mem* $= 1$, but it is necessary to explore the entire main connected component in the current memory model.

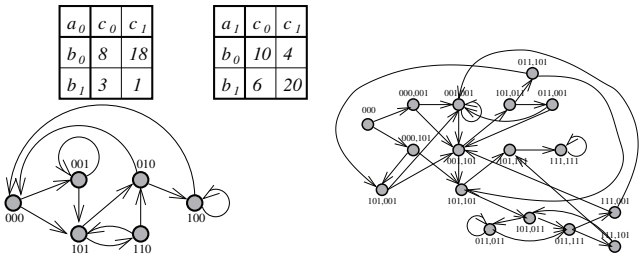

Figure 3: An example for a case where m^* was unreachable for *mem* $= 1$ (left), and became reachable with *mem* $= 2$ (right).

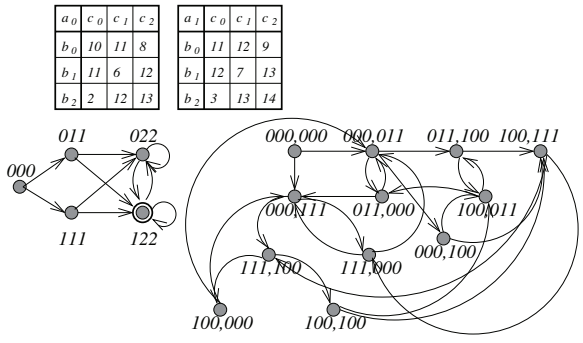

Figure 4: An example for a case where m^* was reachable for *mem* $= 1$ (left), and became unreachable with *mem* $= 2$ (right).

$\mathcal{NP}$-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 (Stone, Kaminka, and Rosenschein 2010) 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.

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 (Garey and Johnson 1990).

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."

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

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. □

Related Work

Stone *et al.* (2010) 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 (2005) 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.* (2006) 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 (2010) 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.

Young (1993) 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 (2005) considered the problem of normal form games against an adaptive opponent with bounded memory. Chakraborty and Stone (2008) examine optimal strategies against a memory bounded learning opponent. 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.

Conclusions and Future Work

In this paper we examine the problem of leading a team of $N \geq 2$ agents by an ad hoc team member 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, thus we 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 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.

There are various directions to be addressed as future work. First, we are examining the case in which the team is lead by more than one ad hoc team member. This raises various interesting questions such as the possibility of coordination among the ad hoc team members that might result in better team performance. Other directions include nondeterminism in agents' perception and uncertainty in the behavior of the team members.

Acknowledgements

This work has taken place in the Learning Agents Research Group (LARG) at the Artificial Intelligence Laboratory, The University of Texas at Austin. LARG research is supported in part by grants from the National Science Foundation (IIS-0917122), ONR (N00014-09-1-0658), and the Federal Highway Administration (DTFH61-07-H-00030).

References

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

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

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

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

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

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

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

Stone, P.; Kaminka, G.; Kraus, S.; and Rosenschein, J. 2010. Ad hoc autonomous agent teams: Collaboration without precoordination. In *Proc. of AAAI'10*.

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

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