To appear in *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems* (AAMAS), Paris, France, May 2014.

Modeling Uncertainty in Leading Ad Hoc Teams

Noa Agmon
Bar-Ilan University, Israel
agmon@cs.biu.ac.il

Samuel Barrett
The University of Texas at Austin, USA
sbarrett@cs.utexas.edu

Peter Stone
The University of Texas at Austin, USA
pstone@cs.utexas.edu

ABSTRACT

Ad hoc teamwork exists when a team of agents needs to cooperate without being able to communicate or use coordination schemes that were designed a-priori. Sometimes ad hoc teamwork amounts to acting so as to bring out the best in your teammates by "leading" them to the optimal joint action. Doing so can be challenging even when their behavior is fully known. In this paper, we take the challenge to the next level by considering the situation in which there is uncertainty about the teammates' behaviors. We discuss the problem of recursive modeling of the teammate's uncertain behavior in two-agent teams and conclude not only that the depth that is useful to model is bounded, but also the number of models useful to consider is linear in the number of actions (and not exponential, as expected). We then show that adopting a naive perspective might lead to negative long-term results in large teams, and thus introduce REACT, an algorithm for determining the action an agent should perform in order to maximize the team's expected utility. Finally, we show empirically that in randomly generated utility matrices, using REACT to select actions outperforms making incorrect assumptions about the identities of teammates.

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

In ad hoc teamwork [16, 18, 19], a group of agents needs to cooperate as a team without having the opportunity to coordinate their behaviors a priori, or use explicit communication for coordination during task execution. Ad hoc teamwork is a new, developing area that emerged due to the growing use of cooperating agents in different domains, such as e-commerce and robotic search and rescue. In such cases, the agents—that were not necessarily designed or programmed similarly—share a joint goal, and each action results in different joint utility of the team. The goal of each agent is, therefore, to choose an action that yields optimal joint utility, given the actual (not necessarily ideal) teammate's behavior.

Sometimes ad hoc teamwork amounts to trying to bring out the best in your teammates by "leading" them to the optimal joint action. In the problem of leading teams in ad hoc settings, one agent (or more) is designed as an *ad hoc* (ah) agent, that has better awareness of the team's possible actions and knowledge of the resulting joint utilities. The other team members act as *best response* (br) agents, i.e., they choose their next action based on their current view of the world, assuming the world and specifically the other agents, will continue performing the same actions that are currently performed. The behavior of the teammates (specifically, their best response reactions) are given and cannot be controlled. The only control we have on the system is via the ad hoc agent. The ad hoc agent aims to use its knowledge to lead the team members into choosing actions that result in higher team utility compared to what could be achieved without its intervention. The problem is designed as a simultaneous repeated game, where at each time step all agents choose their next action based on their world view.

Leading teams in ad hoc settings can be challenging even when teammate behaviors are fully known, as studied in [18, 2]. There, the research concentrated on determining the optimal set of actions an ad hoc agent should perform in order to lead the team to the reachable cyclic set of joint actions with maximal utility. In this paper, we take the challenge of leading teams in ad hoc settings to the next level by considering the situation in which there is uncertainty about the teammates' behaviors.

Modeling an unknown behavior of a teammate is commonly done using recursive modeling, e.g. [6, 13], where the depth of the recursion determines the level of consciousness of the teammate's perception of the world we want to model, and—in our case—of the ad hoc agent. We therefore examine the influence of the depth and the width of the recursion on the state space of the possible actions in two agent teams where the teammate can be either an ad hoc or a best response agent, and show that these are bounded in such cases.

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*
Copyright © 2014, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

We demonstrate that when leading teams with uncertain behaviors, assuming a naive teammate (br agent) can have crucial negative consequences to the team's utility when they are actually ah. We therefore introduce Reducing Expected Action Costs for Teamwork (REACT), an algorithm for determining the optimal actions for the ad hoc agent, that takes into account both long-term utility (set of reachable joint actions) and immediate cost (cost of reaching that optimal long-term solution). REACT can be applied to any possible type of teammate. Finally, we show that misidentifying teammate types can have arbitrarily high costs for long-term utility and empirically evaluate REACT in a variety of situations, including ones in which it must consider uncertainty over several types of teammates.

2. BACKGROUND

In this section we briefly describe the problem of leading ad hoc teams, along with the graphical representation of the domain that aids in solving the problem of leading teams efficiently. The problem of leading in ad hoc teamwork was initially presented by Stone *et al.* [18], as a simultaneous repeated game between two agents: the ad hoc agent (agent A) and the other agent (agent B), which acts as a *best response* - *br* agent. 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 shared joint utilities gained by the team for joint actions is represented by a utility matrix M, where $M(i,j)$ is the joint utility achieved when A and B perform actions a_i and b_j (respectively). The maximal possible joint utility is denoted by m^*, and without loss of generality $m^* = M(x-1, y-1)$. In addition, it is assumed that the system is initialized with A performing a_0 and B performing b_0. The br agent B views the current state of the world, specifically the previous action performed by its teammate, and chooses an action that maximizes the joint utility of the team assuming its teammate continues performing its current action. The *cost* of performing a joint action (a_i, b_j) is defined as $m^* - M(i,j)$.

In a 2-agent team, it is trivially possible to lead the br agent into performing b_{y-1}, yielding maximal utility m^*, simply by A choosing action a_{x-1} and B adapting to the optimal action. The set of joint actions leading from the initial state to m^*, in this case, is $\{(a_0, b_0) \rightarrow (a_{x-1}, b_0) \rightarrow (a_{x-1}, b_{y-1}) = m^*\}$. However, it could be possible to lead B into performing b_{y-1} along a set of joint actions with lower cost compared to the trivial set. The question posed by Stone *et al.* was *how to lead the team to m^* with minimal cost?*

a_0	c_0	c_1
b_0	5	10
b_1	4	2

a_1	c_0	c_1
b_0	8	14
b_1	4	20

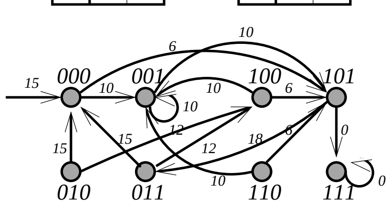


Figure 1: *Example of utility matrices representing a 3-agent team with the corresponding graphical representation: A is the ah agent with actions a_0, a_1, B and C are br with actions b_0, b_1 and c_0, c_1 (respectively)*

In an N-agent team, where the team consists of one ah agent and $N-1$ br agents, it was shown that m^* may become unreachable [2]. In addition, they have shown that the optimal behavior includes a cyclic set of joint actions, and this cyclic set was denoted by the *optimal steady cycle* (osc): a cyclic set of joint actions with minimal *average* cost. The question, in this case, is *how to lead the team to the* osc*?*. Note that this problem is solved in two steps: Finding the osc (minimizes the average cost) and finding the set of joint actions leading to the osc (minimizing the sum of costs).

The method used for solving the problem was by transferring the utility matrices into a graphical representation, where each vertex v in the weighted directed graph $G = (V, E)$ corresponds to a joint action, each edge $(u, v) \in E(G)$ corresponds to a possible transition between the actions (by the br assumption), and the cost of $(u, v) \in E(G)$ is the cost of the joint action corresponding to v. Finding an osc is equivalent to finding a Minimum Cycle Mean in G, thus can be solved in polynomial time in the size of G [11]. An example for a utility matrix and its corresponding graphical representation is shown in Figure 1.

Note that the problem considered in this paper, as well as the problems investigated by Stone *et al.* [18] and Agmon and Stone [2], can be seen as a Partially Observable Markov Decision Process (POMDP). However, as opposed to solving a POMDP that might be intractable, the complexity of finding an optimal solution in this scenario is polynomial (as described later in the paper), based on the modeling we suggest and the bounds on the growth of the environment. This problem can also be considered as a special case of an Interactive POMDP (I-POMDP), which models adversarial interactions of agents by examining the beliefs of an agent on other agents' beliefs, and their beliefs on other agents (e.g. [9]). While the I-POMDP framework could have been used also here, it suffers from the same intractability problem in finding the optimal solution that exists with POMDPs.

In this paper we concentrate on leading teams in ad hoc settings, where the teammates could be either ah or br agents. We restrict ourselves to the case in which no coordination was done a-priori, and the agents cannot communicate explicitly in order to verify their true identity. However, if noticing that a teammate has diverged from the expected br behavior, then clearly it is an ah agent. The question here is *how to lead the team to the* osc *with minimal cost, while taking into account the possible types of the unknown agent(s)?* This question is much more representative of the true ad hoc setting compared to past work in this area.

3. RECURSIVE MODELING

A common method for modeling an opponent in adversarial environments that was shown to be effective in many cases is recursive modeling (e.g. [6]), which simulates the opponent's search of actions in order to exploit it. Other environments that benefit from opponent modeling include negotiation and economics. Constructing a recursive model (RM), commonly represented as a decision tree, has two components: the width of the tree, influenced by the number of possible actions (or choices) of the opponent, and the depth of the tree. In a RM of depth one, the only assumption our agent makes is that the other agent will perform some action or be of some type. In a RM of depth two, agent A will include the understanding of the other agent B has of itself (agent A), and so on. This RM representation tree includes an exponential number of components in terms of the input size.

In [2], the authors suggest a limited RM of depth two, resulting in three possible scenarios: 1) Agent A believes that B is a br agent, 2) A believes that B is ah believing that A is ah, or 3) A believes that B is ah believing that A is br. In this section we discuss the general problem of recursive modeling, concentrating on the two main characteristics of the RM representation: How wide should we spread, and how deep should we go? Fortunately, we are able to show that the results are promising in both directions, specifically that both the width and the depth of the recursion are linear in the number of actions.

Recall that in a 2-agent team, agent A is the ah agent and agent B is the agent with uncertain type. The agents have possible actions $\{a_0, \ldots, a_{x-1}\}$, $\{b_0, \ldots, b_{y-1}\}$ (respectively), where without loss of generality $m^* = (a_{x-1}, b_{y-1})$, and the agents start at joint action (a_0, b_0). We define the *recursive beliefs* as follows. Let $T = \{T_1, \ldots, T_m\}$ be a set of teammate types (in our case $T = \{\mathsf{ah}, \mathsf{br}\}$). If Agent A believes that B is of type T_i, it is noted as $A(B = T_i)$. In recursion of depth two: A believes that B is of type T_i that believes that A is of type T_j is noted as $A(B = T_i(A = T_j))$. Similarly for deeper recursion levels, A believes that B is of type T_i that believes that A is of type T_j that believes that B is of type T_k that believes that A is of type T_l and so on, is noted as $A(B = T_i(A = T_j(B = T_k(A = T_l \ldots))))$

In Lemma 2 we prove that in a recursion of any depth, if the agents believe their teammate is ah, then the outcome is the same: The agents choose (a_{x-1}, b_{y-1}).[1] For example, if $A(B = \mathsf{ah}(A = \mathsf{ah}(B = \mathsf{ah})))$ (recursion of depth 3), then the optimal action of A is equivalent to $A(B = \mathsf{ah})$ (recursion of depth 1). Practically, the lemma restricts the width of the RM. We start with a trivial proposition:

Proposition 1. *Once an agent assumes the other is* br*, the recursion stops.*

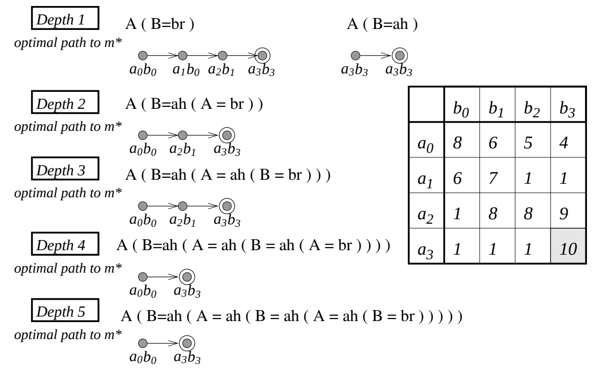
The proposition follows directly from the fact that by the definition of a br agent, it does not have any beliefs of its teammate other than the fact that its next action is consistent with its previous one(s). Therefore, once one agent assumes the other is br the recursion stops, hence the only appearance of a belief in a br teammate is at the bottom of the recursion, for example: $A(B = \mathsf{ah}(A = \mathsf{br}))$.

Lemma 2. *For a recursion of depth N, if the beliefs of the agents are restricted to being* ah*, then $\forall N > 0$ the optimal decision of agent A remains the same as its decision in a recursion of depth one.*

The following theorem concludes that the depth of the recursion worthwhile to examine is bounded. The rationale behind the proof is that when going deeper into the recursion the agents improve their actions resulting in a shorter, better, path towards m^* until there is no room for improvement.

Theorem 3. *The optimal path towards m^* is fixed in all recursions of depth $N' \geq \tilde{N}$, where $\tilde{N}$ is linear in $\min\{x, y\}$. Moreover, the path towards m^* is equivalent $\forall N'' = N' + 2i, i \geq 0$.*

[1]The proofs of all lemmas and theorems are available as appendices in the extended version [1].



	b_0	b_1	b_2	b_3
a_0	8	6	5	4
a_1	6	7	1	1
a_2	1	8	8	9
a_3	1	1	1	10

Figure 2: *Illustrating the influence of the recursion depth on the optimal action of the* ah *agent A.*

We demonstrate in Figure 2 the influence of the depth of the recursion on the optimal action of the ah agent. In the first level either $A(B = \mathsf{ah})$, in which case they both reach m^* immediately; or $A(B = \mathsf{br})$, in which case the shortest path is as calculated by [2, 18]. In the second level (depth = 2), $A(B = \mathsf{ah}(A = \mathsf{br}))$. Starting the interpretation bottom-up, $B = \mathsf{ah}(A = \mathsf{br}))$. Therefore B will assume that A's first action would be a_1, thus to optimally lead A to m^*, B will choose to perform first action b_1 (again, at this point we assume B calculates the shortest path as suggested in [18]). A, being an ah knowing that B's first action would be b_1, will adapt its action to lead the team with lower cost to m^*, thus will choose to perform a_2. By our basic assumption, this choice will reveal to B that A is not br (but an ah), thus they will reach m^* immediately after that. For depth=3 (again, working bottom-up), A thinks that B is br, thus will choose initially action a_1 (as in depth 1). B is an ah agent, thus knowing so will maximize the team's utility by choosing to perform b_1. A, in the next level, knows that B will choose b_1 thus will choose to perform a_2, leading directly after that to m^*. In this example, in the next step of the recursion the agents will transfer directly from a_0, b_0 to m^*.

4. LEADING WITH UNCERTAIN AGENT BEHAVIOR

In the previous section we have established the options for modeling a teammate's behavior. Now, given these models (or others, as shown in Section 5.4), the main question is *how should the ad hoc agent decide what next action to perform?*

4.1 The risk in assuming worst case

When an agent does not have enough information about the environment in order to plan its actions optimally, a common method for solving the problem is to assume a worst case scenario. By doing so, it is possible to guarantee a lower bound on the performance of the system, since if the agent is actually faced with better conditions, its performance could only increase. This method is commonly used in adversarial planning (e.g., [14]), but also in other problems involving single agent planning such as robotic navigation [12].

However, assuming the worst case when leading ad hoc teams with uncertainty might lead to an unbounded utility loss. When leading a team to an optimal behavior, the process is divided into two steps: determining the optimal behavior, in our case the reachable osc (the set of joint actions yielding maximal team utility), and finding the set of joint actions leading to the osc with minimal cost. An agent could always assume the worst teammate, in our case a br agent, and if actually facing an ah agent, it will adjust its actions to jointly lead the team to the reachable osc. In 2-agent teams [2] and in 3-agent teams (Lemma 4), the osc will remain reachable, and in the worst case - the cost of reaching it will increase. However, in the general case of N-agent teams, the consequences of assuming a br agent might cause the team to be unable to reach the osc, thus forcing them into choosing an osc with lower team utility. This results in a **long-term negative consequence of sub-optimal behavior**. Note that here we consider just two possible teammate types, br and ah (and not deeper RMs).

LEMMA 4. *Given a 3-agent team consisting of one ah agent and two teammates: one br and the other that could be either a br or an ah agent. If the ah agent assumes both agents are br, then if the other agent is ah, m^* remains reachable.*

As stated above, in the general N-agent case the situation changes. Here, the power of the ah agent is relatively limited, so every mistake counts: if assuming an uncertain agent to be br, it might divert the system from possibly reaching m^*, even if the uncertain agent is discovered eventually to be an ah agent. An example for a 4-agent team can be viewed in Figure 3. In this example, there are 4 agents: A, B, C, D, where A is our ah agent, B is unknown (either ah or br), and C and D are known to be br. Agents A and B have two possible actions each: a_0, a_1 and b_0, b_1, respectively. Agents C and D have 5 possible actions each, $c_0, \ldots, c_4$ and $d_0, \ldots, d_4$, respectively. If A knows for sure that B is br, then the possible set of joint actions leading to m^* is $(a_0, b_0, c_0, d_0) \rightarrow (a_1, b_1, c_1, d_1) \rightarrow (a_1, b_1, c_4, d_1) = m^*$, with cost $19 + 14 + 0 = 33$. On the other hand, if A knows for sure that B is ah, then it would continue performing a_0, leading to the joint actions $(a_0, b_0, c_0, d_0) \rightarrow (a_0, b_0, c_1, d_1) \rightarrow (a_0, b_0, c_4, d_1) = m^*$, with cost $19 + 13 + 0 = 32$. If A assumes the worst case perspective (a br teammate), but is teamed with an ah agent, then B would perform b_1, yet A will choose a_0. The result would be a set of joint actions never leading to m^*: $(a_0, b_0, c_0, d_0) \rightarrow (a_0, b_1, c_1, d_1) \rightarrow (a_0, b_1, c_2, d_2) \rightarrow (a_0, b_1, c_3, d_3)$. From this point, no matter what A and B do, C and D will continue playing c_2, c_3 and d_2, d_3, respectively. Thus the team will stay at the osc consisting of the latter two states, with joint utility of $(1 + 6)/2 = 3.5$ instead of $m^* = 20$.

a_0b_0	d_0	d_1	d_2	d_3	d_4
c_0	1	3	1	1	1
c_1	3	7	1	1	1
c_2	1	1	1	2	1
c_3	1	1	3	1	1
c_4	1	20	1	1	1

a_0b_1	d_0	d_1	d_2	d_3	d_4
c_0	2	3	1	1	1
c_1	3	8	9	1	1
c_2	1	9	1	10	1
c_3	1	1	10	6	1
c_4	1	1	1	1	1

a_1b_0	d_0	d_1	d_2	d_3	d_4
c_0	1	2	1	1	1
c_1	2	5	6	1	1
c_2	1	6	1	7	1
c_3	1	1	7	4	1
c_4	1	1	1	1	1

a_1b_1	d_0	d_1	d_2	d_3	d_4
c_0	1	4	1	1	1
c_1	2	6	1	1	1
c_2	1	1	1	2	1
c_3	1	1	2	1	1
c_4	1	20	1	1	1

Figure 3: *Payoff matrices representing a case in which the "worst case" perspective can lead to sub optimal joint utility (m^* states are shaded).*

This example shows the importance of having an accurate teammate model, but more than that: if an accurate model does not exist, relying on assuming a naive (br) teammate might harm the team in a way that might not be reversible. Therefore another method for choosing actions, that is more sophisticated than simply assuming a br agent, is required.

4.2 How to Lead a Team with Uncertainty

In the previous sections we have shown that it is worthwhile to examine a bounded recursive model of the unknown teammate and established the fact that assuming the naive teammate as a strategy can be very harmful from the team's perspective. But the question remains: what action should the ad hoc agent take in order to best suit the team's objectives, i.e., maximize the team's utility? In this case, we adopt a risk-averse strategy, in which we prioritize the possible consequences of each choice of action: first by the joint utility of the osc, then by the cost of reaching that point. This prioritization is chosen because getting to the wrong steady cycle has long-term negative consequences with effectively infinite cost, while getting to the right steady cycle the wrong way has finite, one-shot consequences. Happily, as shown in the previous section, the number of options of the teammate's type we have to consider is limited due to the bound on the recursive model. It is important to note that the algorithm described in this section is general and can be applied to any teammate type, as shown in 5.4.

We use the graphical model introduced in [2] in order to identify the points where the actions taken by our agent might change the outcome, considering the cost of the optimal steady cycle. These points in the graph will be denoted as pnr: *Point of No Return.* Upon reaching a pnr, the ah agent will choose (if possible) an action that will lead it to a steady cycle with higher utility, with higher probability (the expected utility is computed with respect to the utility of the steady cycle, not the cost of reaching it). Decisions in other vertices along the way will be made according to the cost of reaching the steady cycle.

In the example presented in Figure 3, a pnr will exist in the vertex corresponding to state (a_0, b_0, c_0, d_0). If, for example, the probability of B being ah is 0.8 and the probability of it being br is 0.2, then the expected long-run steady cycle utility for agent A from choosing action a_0 is $0.8 * 20 + 0.2 * 3.5 = 16.7$, while its expected utility from choosing a_1 is $0.2 * 20 + 0.8 * 3.5 = 6.8$. Thus A should choose to perform action a_0. Note that if the expected utility from a_0 and a_1 were the same, then the decision on which action to choose would have been based on the expected cost towards the osc.

To address this problem, we introduce the Reducing Expected Action Costs for Teamwork (REACT) algorithm. Algorithm 1 describes the working of REACT, specifically the calculation of the pnr, along with the decision on what action to choose at each step. The number of possible identities for the uncertain agent is denoted by r. The inputs to the algorithm are as follows: The directed graph G corresponding to the best response actions of the teammates, along with all its possible actions as an ah agent; the set of paths $\mathcal{O}$ calculated as optimal for each of the r identities—as well as the osc for that option (denoted by $\mathsf{osc_i}$); the probability distribution P over the r different identities of the teammates.

The time complexity of Algorithm 1 is composed of two factors: (1) The complexity of determining an osc and the path leading to it for each of its individual possible behaviors, which by [2] is polynomial in the number of actions, and (2) the complexity of determining the set of actions yielding maximal expected utility given those paths. This is also polynomial in the number of actions, since determining the pnr vertices and the optimal expected utility is by going over all vertices in the graph and comparing the possible next edges of the shortest paths determined in the previous step. If there are multiple uncertain identities, i.e., in an N-agent team, the ah agent is uncertain about the behavior of $M \leq N - 1$ teammates, then such a graph should

Algorithm 1 REACT($G, \mathcal{O} = \{O_0, \ldots, O_{r-1}\}, P = \{P_0, \ldots, P_{r-1}\}$)

for every vertex $v \in V$ **do**
  for every $O_i \in \mathcal{O}$ **do**
    Store cost of travel $C_i(v)$ to the osc along with the utility of the osc_i, $U(i)$
  end for
end for
for every vertex $v \in V(G)$ **do**
  if $\exists u \in V(G)$ s.t. $(v, u) \in O_i \&\& (v, u) \notin O_j$, $i \neq j$ **then**
    for every $u \in V(G)$ s.t. $(v, u) \in E(G)$ **do**
      Compute expected utility: $E_U(u) = \sum_{i=0}^{r-1} P_i U_i$
    end for
    if $\exists u^* \in V(G)$ with maximal $E_U(u^*)$ **then**
      Set next vertex in the shortest path to u
    else
      Compute expected cost of travel $E_T(u) = \sum_{i=0}^{r-1} P_i C_i$ from the set of vertices with maximal utility
      Set next vertex in the shortest path to $\hat{u}$ maximizing $E_T(\hat{u})$
    end if
  end if
end for

be constructed for each combination of identities, i.e., the method is exponential in the number of uncertainties. We leave the exploration of reduction in the number of graphs (thus, perhaps, leading to a polynomial time solution) to future work.

5. EMPIRICAL ANALYSIS

The previous section introduced the idea of the Point of No Return (pnr), explained how to handle these points, and analyzed a single hand-generated payoff matrix. However, it is important to see that pnrs are not limited to carefully handcrafted examples. This section establishes that pnrs do occur in randomly generated payoff matrices and that the long term cost of not reasoning about pnrs can be unbounded. It should be noted that the frequencies of these occurrences and their costs in random matrices may not be representative of the matrices that an ad hoc agent encounters in practice: the actual values will depend heavily on the specific problem domain. Nonetheless, these results show that even without limiting the generation of the matrices, pnrs do occur. In addition, this section presents results on more types of teammates and considers the case in which the agent must select from several possible behaviors that its teammates may be following. Run times of REACT were on average 0.58sec on a high end computing cluster, largely consisting of 2.83Ghz Xeon processors.

5.1 Occurrences in Random Matrices

In this work, we analyze 10 million randomly generated matrices with values in $[0, 1]$ with different settings of the number of agents and the number of actions per agent. In every matrix, the agents start by taking the joint action $(0, 0, \ldots, 0)$. We consider the case where every agent has the same number of actions, and the agents follow the behaviors (ah, unknown, br, br, …, br). In other words, all of the agents' types are known except for that of the second agent. The agent starts with uniform beliefs over the different behaviors the second agent may be employing.

The results in Figure 4 show the number of occurrences of pnrs that prevent the worst case assumption from reaching the osc using the black line with units given on the right y-axis. The cost per step lost in the suboptimal steady cycles averaged across the ten million random matrices is shown by the bars with units given on the left y-axis. The costs of the REACT algorithm are calculated using an agent that reasons about the pnrs and minimizes the long term expected cost of its actions when it reaches the pnrs. The costs are given using a logarithmic scale; when the cost for REACT is not shown, its cost was 0.

These results indicate that REACT's method of considering weighted costs of the reachable steady cycles allows the agent to lead the team to better cycles on average. The differences in results are statistically significant for four agents with either two or three actions using the Mann-Whitney U-test (for data that is not normally distributed) with $p < 0.001$. The other results are not significant due to the low numbers of pnrs.

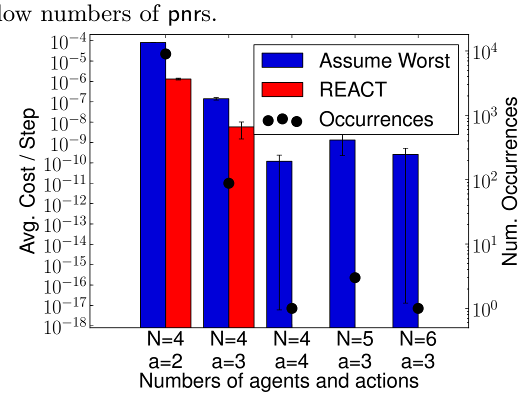


Figure 4: *Occurrences and average costs per step of reaching a steady cycle with a lower utility when assuming that an unknown teammate is br, when in fact it is ah. Results are reported for 10 million randomly generated matrices. When the cost for REACT is not shown, the cost is zero.*

A randomly generated matrix is unlikely to have a pnr, largely due to the high connectivity of a randomly generated matrix. In most cases, the ad hoc agent is able to lead its teammates to nearly every cell of the matrix, therefore allowing it to lead its teammates to the optimal joint utility. The connectivity of the resulting graph increases as the number of actions per agent increases and as the number of agents increases. Therefore, the frequency of pnrs similarly decreases as the number of actions and agents increase, for randomly generated graphs.

In addition, while the average cost of ignoring the pnr is low per time step, the loss will be infinite given an infinite repeated game. Furthermore, the randomly generated matrices may not be representative of the cost matrices that are encountered by ad hoc agents, and the cost of ignoring a pnr is unbounded per *step*. Given a payoff matrix of values varying between 0 and 1, it is possible to trivially transform any matrix with a pnr to have a cost of ≈ 1 if the ad hoc agent ignores the pnr. This transformation can be performed by scaling every value of the matrix between 0 and ε, except for cells on the osc which are set to 1. This transformation raises the cost of ignoring the pnr to $1 - \varepsilon$ per time step.

5.2 Analyzing an Example

To better understand the pnrs in randomly generated payoff matrices, we now describe a single example in the four agent, three action setting. The payoff matrix is given in Table 1, and the various paths that the ad hoc agent would take are illustrated in Figure 5. These paths represent the joint actions that will be selected based on the ad hoc agent's assumptions about its teammates and their actual behavior. The red dashed line shows the joint actions selected if the ad hoc agent correctly knows that the second agent is using the ah behavior. The blue dotted line shows the joint actions that will be chosen if the ad hoc agent correctly knows that the second agent is a br agent. Finally, the solid magenta line shows the actions taken if the ad hoc agent assumes that its teammate is br, when it is in fact ah. The ad hoc agent realizes its mistake at the starred node, but can only reach the nodes marked with a white circle. In this case, it can no longer reach the optimal payoff that was reachable if it had known its teammate was ah and must make do with the indicated cycle, losing 4.66e-3 per step compared to the optimal given a teammate of type ah.

a_0 b_0	d_0	d_1	d_2	a_0 b_1	d_0	d_1	d_2	a_0 b_2	d_0	d_1	d_2
c_0	0.690	0.363	0.061		0.854	0.002	0.280		0.061	0.276	0.815
c_1	0.653	0.709	0.564		0.167	0.140	0.588		0.845	0.516	0.803
c_2	0.672	0.540	0.775		0.161	0.589	0.491		0.682	0.754	0.952
a_1 b_0				a_1 b_1				a_1 b_2			
c_0	0.340	0.464	0.891		0.613	0.592	0.434		0.983	0.749	0.707
c_1	0.232	0.855	0.246		0.979	0.819	0.808		0.871	0.880	0.879
c_2	0.107	0.918	0.475		0.058	0.692	0.090		0.300	0.281	0.541
a_2 b_0				a_2 b_1				a_2 b_2			
c_0	0.421	0.198	0.466		0.264	0.413	0.110		0.175	0.164	0.580
c_1	0.450	0.518	0.301		0.869	0.415	0.248		0.567	0.748	0.849
c_2	0.634	0.870	0.258		0.050	0.421	0.104		0.409	0.071	0.334

Table 1: *An example of a randomly generated matrix that contains a pnr.*

5.3 Existence in Larger Matrices

Due to the high connectivity of randomly generated matrices, the frequency of pnrs decreases as the number of agents and actions increases. However, we can show by induction that pnrs can exist in matrices of any size, even though their frequency is low. Consider the case where there is a pnr in a matrix with n agents. Suppose that an $(n+1)^{\text{th}}$ agent is added where the payoff for all agents is 0 unless the new agent selects action 0. Then, the same steady cycles will remain, with the new agent selecting action 0 at every step. Similarly, look at agent a_i, which currently has m actions. If instead, it has an additional action $m+1$, where the payoffs are 0 if it selects the new action. Then, agent a_i will never select action $m+1$, leaving the steady cycles unchanged. Thus pnrs can exist in matrices of any size and that the cost of mispredicting the teammates' behaviors can be unbounded.

5.4 Additional Teammate Types

While the theoretical analysis in Section 4 focuses on the case when the agent may encounter only br and ah teammates, these results extend to any deterministic type of teammates. Therefore, we now introduce two additional types of behaviors:

- av1 – Never selects action 1, but selects the best response from the remaining options. This teammate represents an agent that may have incomplete information or less ability than its teammates.
- 2br – Selects the second best response. This type of behavior represents the case in which an teammate may have different agendas than its teammates.

Using these new teammate types, the analysis can proceed as in Section 5.2, focusing on the case with four agents and three possible actions for each agent. Then, we consider the case in which the agent is uncertain of one of its teammate's behavior. In this setting, the worst case type of teammate is easy to identify as there is an ordering in the performance of the agent types, specifically that ah $\geq$ br $\geq$ av1 $\geq$ 2br. The results in Figure 6 show that assuming the worst case about its teammate's type can lead to suboptimal performance. The differences between the approaches is statistically significant using the Mann-Whitney U-test with $p < 0.001$ in all of the settings.

These results show that for some types of teammates, making incorrect assumptions more frequently leads to suboptimal performance. The results come from teams of 4 agents, each with 3 actions, on ten million randomly generated matrices. In addition, the cost of these mistakes increase for these cases. These higher rates of occurrence and higher costs arise from the fact that the teammate is more restricted in the br, av1, and 2br cases. Therefore, it is less able to help the team recover from any mistakes, resulting in more cases of poor performance. As in Section 5.1, the results show that the REACT algorithm outperforms the naive approach of assuming the worst case.

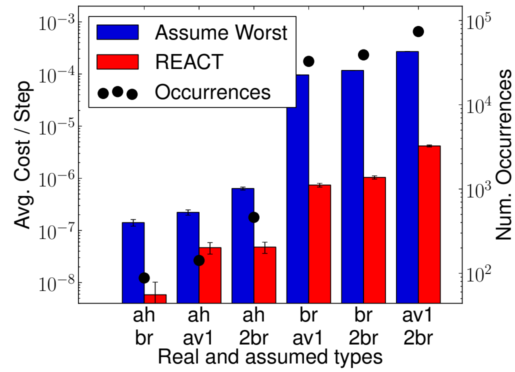


Figure 6: *Occurrences and costs of reaching a steady cycle with lower utility when assuming the worst case with more teammate types. Results are reported for 4 agents with 3 actions over 10 million randomly generated matrices.*

5.5 Multiple Types Simultaneously

The previous section introduced several new types of teammates and analyzed the performance of the agent with teams including these new types of teammates. However, the agent was only ever deciding between two possible types at any given time. In general, it is desirable for the ad hoc team agents to be able to consider teammates of many different types simultaneously. In this general setting, the ad hoc agent needs to be able to reason about the strengths of these various models and must consider what models can be eliminated by observing its teammates' actions.

Tables 2 and 3 show results when the agent must decide between all four possible types, given a prior that all types are equally likely. The agent in this case has a much harder task, as it must reason about all four models, planning how each action may affect the ability to reach a good steady cycle for each of the types. In addition, these results investigate the case where the agent assumes that its teammate is better than it really is (for example, assuming that the teammate is ahwhen it is in fact br). Similar to above, the results show that the REACT algorithm works better than making an incorrect assumption about the teammate's type. The improvement is smaller than above given the harder nature of the task, but REACT still reduces the expected cost compared to making even the best incorrect assumptions, and far outperforms making the worst assumptions. The differences between making any of the incorrect assumptions and using the REACT algorithm are statistically significant using the Mann-Whitney U-test with $p < 0.001$ except for the

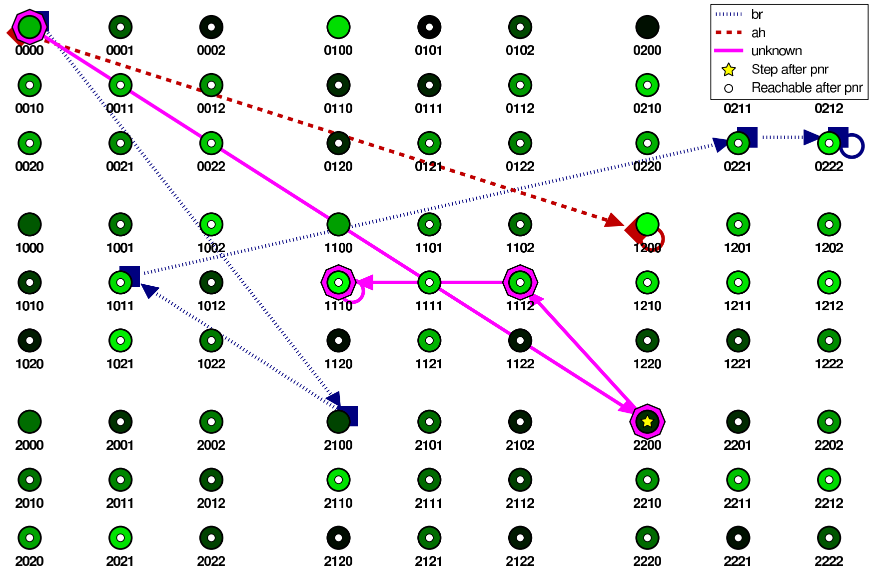


Figure 5: *Possible paths through the example random matrix. If the* ah *agent incorrectly assumes that all teammates are* br*, it will reach the starred node and follow the optimal path (solid line).*

case where the real type is ah and the assumed type is br. In the results in Figure 4, REACT was significantly better than using the incorrect assumption for this case, but the increased difficulty of choosing between four possible models reduces the improvement of the REACT algorithm in this case. Note that the computation for these results is significantly harder than in the two-model cases because there are many more pnrs to consider between each pair of models.

		Assumed Type				
		ah	br	av1	2br	REACT
True Type	ah	0	1.41e-07	2.22e-07	6.37e-07	1.13e-07
	br	9.53e-05	0	9.61e-05	1.17e-04	3.17e-05
	av1	2.09e-04	2.29e-04	0	2.69e-04	6.41e-05
	2br	2.92e-05	2.68e-05	3.03e-05	0	1.71e-05

Table 2: *Average cost per step after reaching a steady cycle compared to reaching the* osc *when the true behavior is known for 10 million randomly generated matrices.*

		Assumed Type				
		ah	br	av1	2br	REACT
True Type	ah	0	88	142	461	88
	br	27,618	0	32,730	39,097	9,709
	av1	54,072	58,990	0	73,932	16,829
	2br	4,861	4,408	5,065	0	3,059

Table 3: *Number of instances out of 10 million randomly generated matrices in which the team did not reach the* osc *due to the agent's lack of knowledge of its team.*

6. RELATED WORK

The formalism of *ad hoc teamwork* was initially introduced by Stone *et al.* [16]. In this setting, agents are engaged in teamwork behavior without prior coordination. This paper concentrates on one aspect ad hoc teamwork in which one or more agents *lead* the team of agents, with no a-priori coordination or explicit communication, to the optimal possible joint-utility, in a repeated simultaneous-action setting with unknown teammate identity.

The problem of leading teammates in ad hoc settings was introduced in [18] for a team of two agents, where the leading agent is an ad-hoc agent, and its teammate is a best response agent. The problem was later extended by Agmon and Stone [2] to leading a team of multiple robots, where they described the graphical representation used as a basis for our work. In both cases, the identity of the teammates is known, and in this work we work towards removing this assumption.

Recent work involving ad hoc teams [4, 19] concentrates on action selection for optimizing the team's utility. They do not assume that the ad hoc agent has more information about the environment, nor that it attempts to lead the team in any way, but rather acts as a team member while adjusting to the teammates' behavior. The subject of uncertainty concerning the agents' behavior is addressed using learning methods [4, 3, 8] or online planners [19].

Stone *et al.* [17] formulated a sequential decision making problem in ad hoc settings of 2-agent teams (A and B) in a k-armed bandit formulation. The question they asked was: Assuming that agent B observes the actions of agent A and its consequences, what actions should agent A select (which arm to pull) in order to maximize the team's utility. 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.

Bowling and McCracken [5] examined the problem of incorporating a single agent into an unknown team of existing agents. 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.* [10] examined the problem of team formation and coordination without prior knowledge, and suggested an architecture based on role selection using auctions for team coordination. 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.

Leading in ad hoc teamwork settings can be seen as a restrictive case of the *environment design* problem. Zhang *et al.* [21] described the problem of designing the environment (modeled as an MDP) such that it influences the behavior of an agent. As opposed to their work, where they change the MDP for optimally leading the agent, our work can be considered as if we are given the MDP and based on it, we seek an optimal policy for the leading agent.

Young [20] 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 long enough to allow adjustment 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 (e.g., [15, 7]) . 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. In particular, our setting does not require safeguarding against adversarial actions by other agents.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we examined the problem of leading teams in ad hoc settings where uncertainty exists regarding their true behaviors. We have shown that if using a recursive model of the teammate's behavior in a two-agent team, then the depth of the recursion as well as its width worthwhile considering are bounded linearly in the number of teammates actions. We then demonstrated why it might be problematic to prepare for the naive teammate, showing that it might affect not only the cost of reaching the optimal set of joint actions, but the reachable set of joint actions itself. We therefore introduce REACT to select the best action the ad hoc agent should take, based on the possible consequences of each choice (in terms of the reachable set of joint actions). In addition, we show that using REACT outperforms making incorrect assumptions about your teammates in several scenarios. Furthermore, we show that this approach extends to a variety of different types of teammates, and the empirical results show that the potential savings of reasoning about uncertainty are quite large.

The subject of leading teams in ad hoc settings is a new, emerging, research area, thus it leaves various directions for future work. For example, an important, practical, direction includes uncertainty in the utility matrix: examining the consequences of the ad hoc agent's uncertainty of the joint utility resulting from each joint action, or it has perception noise, causing it to have uncertainty of the actual action taken by the teammate.

Acknowledgments

This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (CNS-1330072, CNS-1305287) and ONR (21C184-01).

8. REFERENCES

[1] N. Agmon, S. Barrett, and P. Stone. Modeling uncertainty in leading ad hoc teams: Extended version. Technical Report UT-AI-TR-14-01, The University of Texas at Austin, Department of Computer Science, AI Laboratory, February 2014.

[2] N. Agmon and P. Stone. Leading ad hoc agents in joint action settings with multiple teammates. In *Proc. of AAMAS*, 2012.

[3] S. Albrecht and S. Ramamoorthy. Comparative evaluation of MAL algorithms in a diverse set of ad hoc team problems. In *Proc. of AAMAS*, 2012.

[4] S. Barrett, P. Stone, and S. Kraus. Empirical evaluation of ad hoc teamwork in the pursuit domain. In *Proc. of AAMAS*, 2011.

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

[6] D. Carmel and S. Markovitch. Incorporating opponent models into adversary search. In *Proc. of AAAI*, 1996.

[7] 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.

[8] D. Chakraborty and P. Stone. Cooperating with a markovian ad hoc teammate. In *Proc. of AAMAS*, 2013.

[9] P. Doshi and P. J. Gmytrasiewicz. Approximating state estimation in multiagent settings using particle filters. In *Proc. of AAMAS*, pages 320–327, 2005.

[10] 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.

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

[12] S. Koenig and R. Simmons. Solving robot navigation problems with initial pose uncertainty using real-time heuristic search. In *Proc. of AIPS*, 1998.

[13] R. E. Korf. Generalized game trees. In *Proc. of IJCAI*, pages 328–333, 1989.

[14] P. Paruchuri, M. Tambe, F. Ordonez, and S. Kraus. Security in multiagent systems by policy randomization. In *Proc. of AAMAS*, 2007.

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

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

[17] P. Stone, G. A. Kaminka, S. Kraus, J. R. Rosenschein, and N. Agmon. Teaching and leading an ad hoc teammate: Collaboration without pre-coordination. *Artificial Intelligence*, 203:35–65, October 2013.

[18] P. Stone, G. A. Kaminka, and J. S. Rosenschein. Leading a best-response teammate in an ad hoc team. In *Agent-Mediated Electronic Commerce (AMEC)*, 2010.

[19] F. Wu, S. Zilberstein, and X. Chen. Online planning for ad hoc autonomous agent teams. In *Proc. of IJCAI*, pages 439–445, 2011.

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

[21] H. Zhang, Y. Chen, and D. C. Parkes. A General Approach to Environment Design with One Agent. In *Proc. of IJCAI*, pages 2002–2009, 2009.