To appear in *Autonomous Agents and Multiagent Systems - Adaptive Learning Agents Workshop* (AAMAS - ALA), St. Paul, Minnesota, USA, May 2013.

Communicating with Unknown Teammates

Samuel Barrett
Dept. of Computer Science
The Univ. of Texas at Austin
Austin, TX 78712 USA
sbarrett@cs.utexas.edu

Noa Agmon
Dept. of Computer Science
Bar-Ilan University
Ramat Gan, 52900 Israel
agmon@macs.biu.ac.il

Noam Hazon
Dept. of Computer Science
Bar-Ilan University
Ramat Gan, 52900 Israel
hazonn@macs.biu.ac.il

Sarit Kraus
Dept. of Computer Science
Bar-Ilan University
Ramat Gan, 52900 Israel
sarit@cs.biu.ac.il

Peter Stone
Dept. of Computer Science
The Univ. of Texas at Austin
Austin, TX 78712 USA
pstone@cs.utexas.edu

ABSTRACT

Teamwork is central to many tasks, and past research has introduced a number of methods for coordinating teams of agents. However, with the growing number of sources of agents, it is likely that an agent will encounter teammates that do not share its coordination method. Therefore, it is desirable for agents to adapt to these teammates, forming an effective ad hoc team. Past ad hoc team research has focused on cases where the agents do not directly communicate. This paper tackles the problem of communication in ad hoc teams, introducing a minimal version of the multiagent, multi-armed bandit problem with communication between the agents. The theoretical results in this paper prove that this problem setting can be solved in polynomial time when the agent knows the set of possible teammates. Furthermore, the empirical results show that an agent can cooperate with a variety of teammates not created by the authors even when its models of these teammates are imperfect.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*

General Terms

Algorithms

Keywords

Ad Hoc Teams, Multiagent Systems, Teamwork

1. INTRODUCTION

Given the growing number of both software and robotic agents, effective teamwork is becoming vital to many tasks. Robots are becoming cheaper and more durable, and software agents are becoming more common for tasks including bidding in ad auctions. These agents are being developed by an increasing number of companies and research laboratories. As the number of sources of agents grows, so does the need for agents to cooperate with a variety of different teammates.

This need is addressed in the area of *ad hoc teamwork*, where agents are evaluated in their ability to cooperate with a variety of teammates. Stone et al. [20] define ad hoc teamwork problems as problems in which a team cannot precoordinate its actions, and they argue that evaluating an ad hoc team agent fundamentally depends on both the domains it may face as well as the teammates it can encounter. Then, they introduce an evaluation algorithm that includes this consideration.

One example of where ad hoc teamwork is especially applicable is in search and rescue. Currently, after a disaster, robots from a number of different laboratories are used to map the area and locate survivors. However, these robots are not designed to work with each other. With more time, these robots could be programmed to use existing coordination algorithms, but, given time constraints, this is not usually possible. Therefore, it is desirable for these robots to be able to intelligently adapt to a variety of possible teammates.

Past work on ad hoc teamwork has focused on the case where the ad hoc agent cannot directly communicate to its teammates. Instead, the focus of this work is on how an agent can influence its teammates through limited communication when a common language exists and the agent has more knowledge than its teammates. However, the ad hoc agent cannot influence how its messages are interpreted, only the messages it sends. This work has three main contributions, the first being the introduction of a minimal domain for investigating teammate communication. The second contribution is proving that several scenarios (with two Bernoulli actions and three types of messages) are solvable in polynomial time. However, for practical use, the polynomial algorithm does not scale well, so the third contribution is the evaluation of an empirical planning algorithm.

2. PROBLEM DESCRIPTION

This paper focuses on a multiagent, multi-armed bandit problem that allows limited communication. The multi-armed bandit setting is chosen as it has been well studied in the past, and it serves as a minimal decision making domain that exhibits the necessary properties for investigating communication with unknown teammates. The multi-armed bandit setting is a fundamental problem in single agent reinforcement learning [22]. Ad hoc teamwork in the bandit domain is studied in [21], in a setting where one agent teaches its teammate. However, in their formulation, the ad hoc

agent *knows* its teammates' behaviors, and the agents *cannot* communicate. Therefore, the ad hoc agent must teach its teammate solely through the arms it chooses to pull.

2.1 Ad Hoc Teamwork

While general multiagent research focuses on creating a coordinated team to perform tasks, this research focuses on ad hoc teamwork. In ad hoc teamwork, the goal is to create agents that can cooperate with a variety of possible teammates. Specifically, we assume that n of the agents are predesigned to accomplish the task as a coordinated team of $n+1$ agents. However, the remaining agent is an ad hoc agent. We treat the n agents' behaviors as fixed; we cannot change them. Instead, we design only the ad hoc agent's behavior to maximize the shared payoffs received by the team.

2.2 Bandit Setting

Formally, the bandit problem in this paper is given by the tuple $G = (\mathbb{A}, \mathbb{C}, \mathbb{P}, T)$ where $\mathbb{A}$ is a set of two Bernoulli arms $\{arm_1, arm_2\}$ that return either 0 or 1, $\mathbb{C}$ is a set of possible communications that can be sent including the broadcasting costs, $\mathbb{P}$ denotes the players in the problem with $|\mathbb{P}| = n+1$, and T is the number of rounds. Each round in the problem involves two phases: 1) a communication phase followed by 2) an action phase. In both phases, all agents act simultaneously. In the communication phase, each agent can broadcast a message of each type to its teammates, but messages have costs. Specifically, there are three messages types, and each type has an associated cost:

- **obs** – Send the agent's last selected arm and payoff
- $\mathbf{mean}_{arm}$ – Send the agent's observed mean and number of pulls for the specified arm
- $\mathbf{suggest}_{arm}$ – Suggest that the agent's teammates pull the specified arm

Other types of communication are possible, but for formal analysis, it is useful to limit the types of communication. In the action phase, each agent chooses a single arm and observes a payoff from that arm. We use arm_* to denote the arm with the highest payoff.

2.3 Behavior

Since the ad hoc agent's teammates form an existing team, we assume that they are tightly coordinated. Specifically, we assume that the team's behavior can be described as a function of the team's total number of pulls and successes of each arm, but only the ad hoc agent's pulls and successes that it has communicated. While assuming that the teammates operate as a function of the team's observations is a strong assumption, it is possible in many scenarios. For example, each agent could broadcasts all observations or if the team's actions are coordinated and they only broadcast when the chosen arm returns 1. However, this assumption is removed in Section 4, although the ad hoc agent continues planning as if this assumption were true.

This behavior consists of an action and a communication function. We denote the action function of each teammate by the function *act*, where the result of *act* is a probability distribution over the agent pulling each arm. Specifically, $\Pr(a_i^t = arm_j) = act(p_0, s_0, p_1, s_1, i, T-t, \text{sugg})$ where a_i^t is the action chosen by agent i in round t, p_k is the number of times that arm_k was pulled by the team, s_k is the number of times that arm_k returned a value of 1, $T-t$ is the number of rounds remaining, and sugg is the ad hoc agent's last suggestion. We denote the teammates' communication function by *comm*, where the result of *comm* is the probability of sending each possible message. In particular, $\Pr(c_j \in C_i^t) = comm(p_0, s_0, p_1, s_1, i, T-t, \text{sugg})$ where C_i^t is the set of messages broadcast by agent i in round t and $c_j \in \mathbb{C}$.

3. THEORETICAL ANALYSIS

To solve this general problem, we first tackle the simplest version of the problem and then progressively relax the problem's assumptions. Specifically, Sections 3.1–3.4 show that a variety of ad hoc team problems in the bandit setting can be solved in polynomial time, as summarized in Table 1. These results do not prove communication will improve the team's performance, instead proving that agents can reason about communication without moving to a higher complexity class. However, the empirical results in Section 4 show that communication does help improve the team's performance in many scenarios.

Knowledge of Teammates	Teammate Type	Knowledge of Environment	Solution Type	Section
Known	Stochastic	Known	Exact	3.1
Finite Set	Deterministic	Known	Exact	3.2
Parameterized Set	Stochastic	Known	Approx.	3.3
Parameterized Set	Stochastic	Unknown	Approx.	3.4

Table 1: A summary of the ad hoc team problems that are proven to be solvable in polynomial time.

3.1 Known Teammates and Arms

In this setting, the ad hoc agent knows the true distributions of the arms and can observe its teammates' actions and the resulting payoffs. In addition, the ad hoc agent knows the true stochastic behavior of its teammates, i.e. the ad hoc agent knows both *act* and *comm*. Therefore, the ad hoc agent has a full model of the problem and must plan to optimally select the best actions and messages. Arms other than arm_* are considered because their observations affect the messages that the ad hoc agent can send to affect its teammates' actions.

We model this problem as a Markov Decision Process (MDP). An MDP is a 4-tuple $M = (S, A, P, R)$ where S is a set of states, $A(s)$ is the set of actions available from state $s \in S$, $P(s, a, s') = \Pr(s_{t+1} = s' | s_t = s, a_t = a)$ is the transition function specifying the probability of reaching state s' after taking action a in state s, and $R(s, a, s')$ is the resulting immediate reward.

In the bandit setting, part of the state corresponds to the pulls and observations of the ad hoc agent's teammates. The other factor of the state comes from the communications of the agents. Specifically, the state is given by the vector $(p_0, s_0, p_1, s_1, t, \text{sugg}, p_0^a, s_0^a, p_1^a, s_1^a, p_0^c, s_0^c, p_1^c, s_1^c, \text{phase})$, where p_i and s_i are the number of pulls and successes of arm_i, p_i^a and s_i^a are the number of pulls and success of the ad hoc agent, p_i^c and s_i^c are the number of communicated pulls and successes, and sugg is the ad hoc agent's most recent suggestion. We split the pulls and successes communicated by the ad hoc agent from the team's totals to model how its communications will affect the team. To better illustrate the communication, if the ad hoc agent has previously communicated its last observation, communicating its observed number of pulls and successes of the same arm will replace

its teammates' memory of this observation. Given that there are T rounds and n teammates, p_i and s_i are each bounded by nT, and p_i^a, s_i^a, p_i^c, and s_i^c are each bounded by T. The round t is bounded by T, and the most recent suggestion sugg takes on one of 3 values (arm_0, arm_1, or no suggestion). Finally, there are 2 possible phases of a round. Therefore, the state space has at most $(nT)^4 \cdot T \cdot T^4 \cdot T^4 \cdot 3 \cdot 2 = 6n^4T^{13}$ states.

The transition function P is composed of the *act* and *comm* functions, the probability distributions of the arms, and the ad hoc agent's actions. The reward function R is a combination of the rewards coming from the arms and the costs of communication. The actions of the MDP are selecting arms and messages. In the communication phase, there are $2 \cdot 3 \cdot 3 = 18$ possible actions, coming from optionally sending obs; sending mean_0, mean_1, or no mean; and suggesting one of the two arms or sending no suggestion.

Using Dynamic Programming (DP), it is possible to find the optimal solution to an MDP in polynomial time in terms of the number of states and actions [22]. The number of states is polynomial in terms of T and n, and the number of actions is 18. Let b be the time that it takes to calculate the behavior of the entire team. Calculating the transition function takes time proportional on b for any given state. Therefore, Theorem 1 directly follows.

THEOREM 1. *An ad hoc agent that knows the arm distributions and its teammates' behaviors can calculate its optimal behavior for maximizing the team's shared payoffs in poly(T, n, b) time.*

3.2 Teammates from a Finite Set

In this section, we relax the constraint on knowing the teammates' behaviors. Instead, the ad hoc agent knows that the behaviors are drawn from a known, finite set of deterministic behaviors. In addition, it still knows the true distributions of the arms. This case is of interest because a finite set of behaviors can often cover the space of likely behaviors. For example, analysis of bandit problems [17], ad hoc teamwork [3], and using machine learning with psychological models [18] suggests that a finite number of behaviors may be representative of the spread of possible behaviors that teammates may exhibit.

In general, this finite set of behaviors can vary, but, we consider two types of teammates: 1) greedy agents and 2) ones that choose arms using confidence bounds, specifically using UCB1 [1]. The teammates are assumed use the ad hoc agent's communicated pulls in selecting actions and to share information, but the ad hoc agent cannot determine their types from their messages. The ad hoc agent is given a prior probability over encountering teams following either of these behaviors.

To tackle this problem, we add the ad hoc agent's beliefs about its teammates into the state space. As the teammates are deterministic, there are three possibilities for the belief space: either both models are possible, only the greedy is possible, or only the UCB1 one is possible. Therefore, the combined belief and world state space is three times larger than the world state space, and the resulting MDP has state space of size $18n^4T^{13}$. The transition function can be modified to simultaneously update the ad hoc agent's beliefs as well as the world state based on whether a teammate model predicts the observed actions. Therefore, the MDP can again be solved using DP in polynomial time. Theorem 2 follows directly from this reasoning.

THEOREM 2. *An ad hoc agent that knows the arm distributions and that its teammates' behaviors are drawn from a known set of two deterministic behaviors can calculate its optimal behavior for maximizing the team's shared payoffs in poly(T, n, b) time.*

3.3 Teammates from a Continuous Set

In this section, we further relax the constraints on the teammates' behaviors, considering a continuous set of stochastic behaviors. We still consider a small number of possible behaviors, specifically ε-greedy and UCB(c). For these behaviors, ε is the probability of taking a random action, and c is a scaling factor of the confidence bound. Therefore, the ad hoc agent must maintain a belief distribution over values of ε, values of c, and the behavior types. The ad hoc agent knows that ε, c are uniformly distributed over $[0, 1]$, and it knows the prior probabilities of the two models. Note that while we only use two models for simplicity, this analysis can be extended for any fixed number of models.

To tackle this problem, we model the problem as a partially observable Markov decision process (POMDP). In our case, the belief space has three partially observed values: ε, c, and the probability of the teammates being ε-greedy versus UCB(c). The transition function for the fully observable state variables remains the same as the original MDP. The probabilities of the two models are updated given the probability that each of the models would have predicted the observed actions, and the updates to the probability distributions of ε and c are described in Lemma 3. The remainder of the POMDP remains as defined above.

Note that this problem is closely related to the problem of determining which MDP an agent is in, as studied in [5]. However, no formal bounds on the computational complexity of selecting an MDP have been found, though empirical results show that existing POMDP solvers can perform well on these problems.

Specifically, in Lemma 3 and Theorem 4, we show that in this expansion of the problem, the ad hoc agent can perform within η of the optimal behavior with calculations performed in polynomial time. This result comes from reasoning about the δ-covering of the belief space. For a metric space A, a set B is a *δ-covering* if $\forall a \in A \ \exists b \in B$ such that $|a - b| < \delta$. Intuitively, a δ-covering can be thought of as a set of multi-dimensional balls filling a space.

LEMMA 3. *The belief space of the resulting POMDP has a δ-covering with size poly$(T, n, 1/\delta)$.*

PROOF. Using Proposition 1 of [12], we know that the fully observed state variables result in a multiplicative factor that is polynomial in T and n. The probability between the two models is a single real value in [0,1], resulting in a factor of $1/\delta$. The parameter ε has a uniform prior, so the posterior is a beta distribution, relying on two parameters, α and β. These parameters correspond to the (fully observed) number of observed greedy and random pulls; thus, each are integers bounded by nT. Therefore, the probability distribution over ε results in a factor of size $(nT)^2$.

The parameter c has a uniform prior, and UCB agents choose based on comparing $\frac{s_i+s_i^c}{p_i+p_i^c} + c\sqrt{\frac{ln(p_0+p_0^c+p_1+p_1^c)}{p_i+p_i^c}}$ for $i = 1, 2$. Using linear programming, pieces of the range of c

can be eliminated, but the posterior remains uniform. Given the nature of c, the eliminated pieces of the range must be at the top or bottom of the current range of c. Therefore, the probability distribution over c can be represented using two real values in $[0, 1]$ that are the top and bottom of the uniform range of c, resulting in a factor of $1/\delta^2$ Combining these all of these factors results in a δ-covering of size $\text{poly}(T, n, 1/\delta)$. $\square$

From Theorem 1 in [15], it is known that a POMDP can be approximately solved in time polynomial in terms of the size of its covering number. While this theorem shows this case for the infinite horizon, discounted rewards case, these results extend to the simpler finite horizon setting. Given this result and Lemma 3, Theorem 4 follows directly.

THEOREM 4. *If an ad hoc agent can observe its teammates' actions, knows the arm distributions, and knows that its teammates are drawn from a known, continuous set of ε-greedy and UCB teammates, it can calculate an η-optimal behavior in poly$(n, T, b, 1/\eta)$ time.*

3.4 Unknown Arms

The previous sections assumed that the ad hoc agent already knew the underlying distributions of the arms, but in many cases the ad hoc agent may not have prior knowledge of the arms. Therefore, it is desirable for the ad hoc agent to be able to reason about trading off between exploring the domain, exploring its teammates, and exploiting its current knowledge. In this section, we prove that the ad hoc agent can optimally handle this tradeoff while planning in polynomial time. However, we still assume that the ad hoc agent knows the pulls and payoffs of its teammates, either by observing them or through their communications.

The belief space of the POMDP is increased to track two additional values, one for the Bernoulli success probability for each arm. The probabilities of each of these values can be tracked using a beta distribution similar to ε in Lemma 3, resulting in a multiplicative factor of $(nT)^2$. Therefore, the covering number has size $\text{poly}(T, n, 1/\delta)$. Theorem 5 follows naturally from this result and the reasoning in Theorem 4.

THEOREM 5. *If an ad hoc agent that does not know the arm distributions, but has a uniform prior over their success probability and does know that its teammates' behaviors are drawn from a known, continuous set of stochastic behaviors and can observe the results of their actions, it can calculate an η-optimal behavior in poly$(n, T, b, 1/\eta)$ time.*

4. EMPIRICAL EVALUATION

While the previous section focused on proving that our multi-armed bandit problems can be solved in polynomial time, the existing techniques for calculating exact solutions are impractical for solving problems with more than a couple of rounds and more than two arms. Therefore, in the empirical setting, we use Partially Observable Monte-Carlo Planning (POMCP) [19]. POMCP has been shown to be effective on a number of large POMDPs, and similar planning methods have been effective for ad hoc teamwork [3]. While POMCP will find an optimal solution given unlimited computation, no guarantees exist with limited computation. However, our results show that POMCP can be effective for cooperating with a variety of possible teammates in the multi-armed bandit setting.

4.1 Methods

POMCP is a Monte Carlo Tree Search (MCTS) algorithm that is based on the Upper Confidence bounds for Trees (UCT) algorithm [14]. Specifically, POMCP starts from the current state of the problem and performs a number of simulations until reaching the end of the problem. These simulations involve selecting the ad hoc agent's actions using upper confidence bounds and randomly sampling the outcomes from the arms. For its teammates, the ad hoc agent plans as if they are selecting actions using either the ε-greedy or the UCB algorithms. To model the effects of suggestions, agents are given some probability of following the suggestion rather than taking their regular action, with the probability being uniformly drawn from [0,1]. In all of the evaluations, we assume that the ad hoc agent can observe its teammates' actions and payoffs. The ad hoc agent knows the true distributions of the arms except where otherwise noted (Figure 3).

While we evaluate the ad hoc agent when it encounters teammates that are using the ε-greedy and the UCB algorithms, we also consider a number of agents that were not created by the authors, denoted *externally-created teammates*. These agents were designed by undergraduate and graduate students as part of an assignment on agent design. To prevent any bias in the creation of the agents, the students designed the entire team and were unaware of the ad hoc teamwork problem. These agents were given the same types of messages available to the ad hoc agent.

Section 2 specifies that the teammates are assumed to be tightly coordinated and know each other's actions and payoffs. However, the externally-created agents cannot freely share this information, instead they can communicate to their teammates given the existing message types. Therefore, modeling them as completely sharing their knowledge is imperfect, but this simplifies planning and our results indicate that this model is sufficient for achieving good performance. Note that the ad hoc agent can still observe the teams' actions and payoffs, but they do not observe the ad hoc agent's actions. In addition, the externally-created agents are not purely ε-greedy or UCB; they follow a variety of possible behaviors, with varying effectiveness, and may react differently to incoming messages. Rather than being optimal, these agents represent a diverse set of imperfect agents that may be created by a variety of designers attempting to solve a real problem.

4.2 Results

All evaluations use 100 runs with randomly selected teams, random either in the parameters for the ε-greedy and UCB teams, or in the selection of which externally-created team to use. The probability of a success on each arm is randomly selected as well. In this analysis, the ad hoc agent initially samples a number of ε-greedy and UCB teams with random parameter values. When the ad hoc agent knows the correct behavior type, the results are similar to knowing that either ε-greedy or UCB teams are possible. The results are the average team rewards normalized by the average reward if every agent continuously pulled the best arm. Statistical significance is tested with a paired Student-T test with $p < 0.05$ and is denoted with a "+" in the figures when comparing POMCP to all other methods.

We compare four behaviors of the ad hoc agent:

- **Match** - Plays as if it were another agent of the team's type, but can observe all agents' results

- **NoComm** - Always pulls the best arm and does not communicate
- **Obs** - Always pulls the best arm and communicates its last observation
- **POMCP** - Plans using POMCP which arm to pull and what to communicate

Match, NoComm, and Obs serve as baselines. Pulling the best arm and communicating other messages were tested, but generally produced worse results than either NoComm or Obs. Match is only used as a baseline when the arms' payoffs are unknown, as there was no way to provide the externally-created teammate serving as the ad hoc agent with the true knowledge of the arms' payoffs. Unless otherwise specified, there are 3 arms, 10 rounds, and 7 externally-created teammates to test how our approach scales to bigger problems than are theoretically proven. Furthermore, the costs for sending messages are randomly selected for each run, and all agents are informed of the costs. To model the size of different messages and allow for varied communication scenarios, the cost of sending the last observation is selected from $[0, 2m]$ (arm and payoff), the cost of sending the mean of an arm is in $[0, 3m]$ (arm, pulls, and successes), and the cost of suggesting an arm is in $[0, m]$ (arm), where $m = 0.75$ unless specified.

Figure 1 presents the results when the ad hoc agent encounters the problem discussed in Section 3.3, cooperating with teams that are ε-greedy or UCB, with varied message costs. Note that NoComm is unaffected by the message costs as it does not communicate. The results indicate that the agent can effectively plan its actions, significantly outperforming the baselines.

On the other hand, Figure 2 shows the results with externally-created agents, a problem not covered by any theoretical guarantees, as the models do not match the true teammates. If all agents start with no observations of the arms, all the methods perform similarly because the teammates usually quickly converge to the best arm. Therefore, for these results, we consider the case where in the first 5 rounds, the teammates' pulls of the best arm are biased to have a lower chance of success. Then, we evaluate how well the ad hoc agents help correct their teammates' biases. In these evaluations, we test the sensitivity of the agent to various problem parameters. Note that the message costs are also applied to the externally-created teammates, which are informed of the current message costs, so the performance of NoComm is also affected by message costs.

As the cost of communicating increases, NoComm becomes closer to the optimal behavior. As the number of rounds increases, communicating is more helpful because there is more time to reap the benefits of better informing the teammates. With more arms, it is harder to get the teammates to select the best arm, so communicating is less helpful. With more teammates, communicating is more likely to be outweighed by other agents' messages, but there is more benefit if the team can be convinced, hence the improvement of Obs. Overall, the results in all of these scenarios tell a similar story, specifically that reasoning about communication helps an ad hoc agent effectively cooperate with various teammates, even when its models of these teammates are incomplete or incorrect.

Finally, in Figure 3, we investigate how well the ad hoc agent can do when it does not know the true payoffs of the arms, but can still observe the payoffs of its teammates' actions. In the POMCP setting, the ad hoc agent samples its starting states by randomly selecting the payoff value of each arm. In the NoComm and Obs settings, the ad hoc agent acts ε-greedy, with $\varepsilon = 0.1$, because it does not know the true best arm. To encourage more sharing when the knowledge is limited, the base message cost is set to $m = 0.04$. The results show that even when the ad hoc agent is unsure of the arms' payoffs, it performs better by using communication in order to cooperate with its teammates.

5. RELATED WORK

Multiagent teamwork is a well studied topic, with most work tackling the problem of creating standards for coordinating and communicating. These efforts include the SharedPlans framework [10] which reasons about agents that share common recipes for coordination based on modeling agents' beliefs and intentions. One popular approach is the STEAM framework [23] which provides a set of generalized teamwork rules. In the TAEMS framework [11], agents use a set of hierarchical rules to model coordination relationships for the agents and tasks. While these algorithms have been shown to be effective, they require that the teammates share this coordination algorithm.

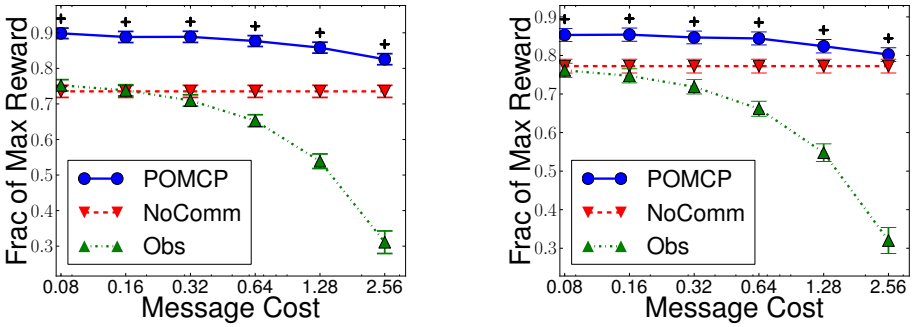
On the other hand, ad hoc teamwork focuses on the case where the agents do not share a coordination algorithm. Bowling and McCracken [4] consider robots playing soccer in which the ad hoc agent has a playbook that differs from its teammates'. In [16], Liemhetcharat and Veloso reason about selecting agents to form ad hoc teams. Barrett et al. [3] empirically evaluate an MCTS-based ad hoc team agent in the pursuit domain, and Barrett and Stone [2] analyze existing research on ad hoc teams and propose one way to categorize ad hoc teamwork problems. Other approaches include Jones et al.'s work [13] on ad hoc teams in a treasure hunt domain. A more theoretical approach is Wu et al.'s work [24] into ad hoc teams using stage games and biased adaptive play.

Ad hoc teamwork is also closely related to the area of opponent modeling, differing in whether one models teammates or opponents. Interacting with opponents often requires reasoning about worst case scenarios. One promising approach for opponent modeling is the AWESOME algorithm [7], which tackles repeated games and guarantees convergence and rationality. Further work investigates agents that explicitly model and reason about their opponent's beliefs in the form of interactive POMDPs [9] and interactive dynamic influence diagrams (I-DIDs) [8].

6. CONCLUSION

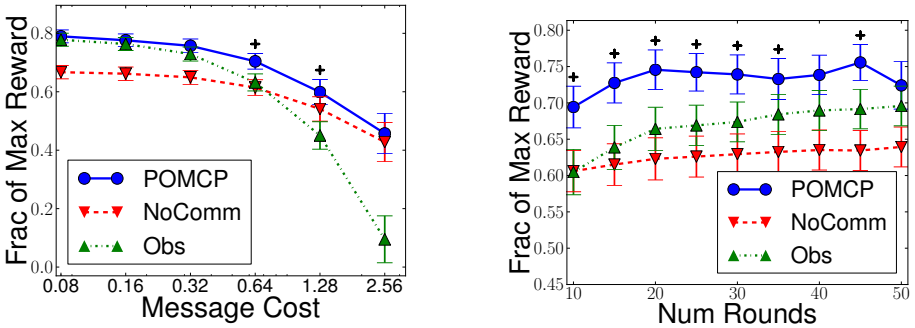
Past work into ad hoc teamwork has largely focused on scenarios in which the ad hoc agent cannot directly communicate with its teammates. This work addresses this gap by introducing a minimal domain with communication and proving that ad hoc team agents can optimally cooperate in some scenarios using only polynomial computation. Note that while the ad hoc agent controls what messages to send to its teammates, it cannot control their reactions to the messages. Furthermore, this paper evaluates an empirical algorithm for planning in ad hoc teamwork problems. This algorithm is shown to be effective when cooperating with teammates created by a variety of developers.

One interesting avenue for future research is investigating ad hoc teamwork in more complicated domains. Another

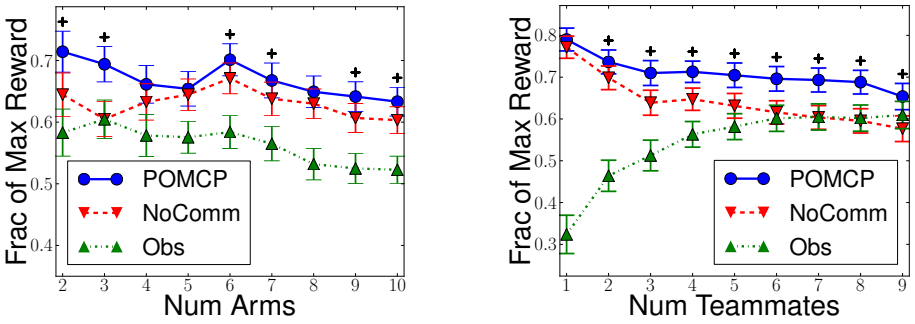


(a) ε-greedy teammates (b) UCB teammates

Figure 1: Normalized rewards with varied message costs with a logarithmic x-axis. Significance is denoted by "+"

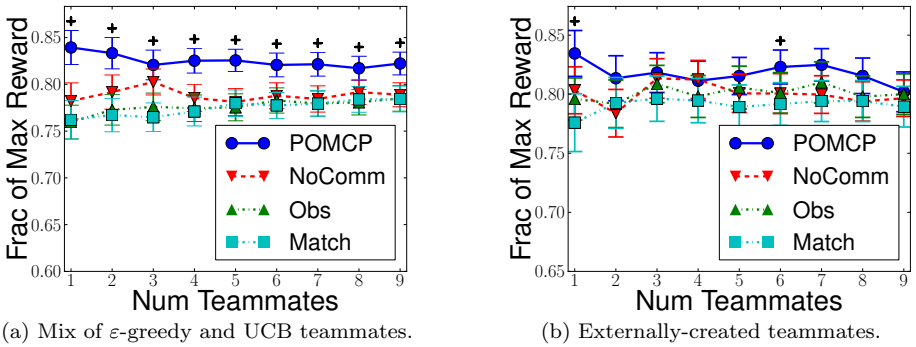


(a) Message costs with logarithmic x-axis. (b) Numbers of rounds.



(c) Numbers of arms. (d) Numbers of teammates.

Figure 2: Normalized rewards with varied parameters when cooperating with externally-created teammates.



(a) Mix of ε-greedy and UCB teammates. (b) Externally-created teammates.

Figure 3: Normalized rewards when dealing with unknown arms and varying numbers of teammates.

problem to consider is an ad hoc agent learning about agents that are in turn learning about it in a recursive modeling setting [6]. Another area for future work is applying ad hoc teamwork to enable agents to interact with humans.

7. REFERENCES

[1] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47:235–256, May 2002.

[2] S. Barrett and P. Stone. An analysis framework for ad hoc teamwork tasks. In *AAMAS '12*, June 2012.

[3] S. Barrett, P. Stone, and S. Kraus. Empirical evaluation of ad hoc teamwork in the pursuit domain. In *AAMAS '11*, May 2011.

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

[5] E. Brunskill. Bayes-optimal reinforcement learning for discrete uncertainty domains. In *AAMAS '12*, 2012.

[6] D. Carmel and S. Markovitch. Incorporating opponent models into adversary search. In *Proc. of AAAI*, pages 120–125, 1996.

[7] V. Conitzer and T. Sandholm. AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. *Machine Learning*, 67, May 2007.

[8] P. Doshi and Y. Zeng. Improved approximation of interactive dynamic influence diagrams using discriminative model updates. In *AAMAS '09*, 2009.

[9] P. J. Gmytrasiewicz and P. Doshi. A framework for sequential planning in multi-agent settings. *JAIR*, 24(1):49–79, July 2005.

[10] B. Grosz and S. Kraus. The evolution of sharedplans. In A. Rao and M. Woolridge, editors, *Foundations and Theories of Rational Agency*, pages 227–262, 1999.

[11] B. Horling, V. Lesser, R. Vincent, T. Wagner, A. Raja, S. Zhang, K. Decker, and A. Garvey. The TAEMS White Paper, January 1999.

[12] D. Hsu, W. S. Lee, and N. Rong. What makes some pomdp problems easy to approximate? In *Advances in Neural Information Processing System*. 2007.

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

[14] L. Kocsis and C. Szepesvari. Bandit based Monte-Carlo planning. In *ECML '06*, 2006.

[15] H. Kurniawati, D. Hsu, and W. S. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *In Proc. Robotics: Science and Systems*, 2008.

[16] S. Liemhetcharat and M. Veloso. Modeling mutual capabilities in heterogeneous teams for role assignment. In *IROS '11*, pages 3638 –3644, 2011.

[17] C. Mayo-Wilson, K. Zollman, and D. Danks. Wisdom of crowds versus groupthink: learning in groups and in isolation. *International Journal of Game Theory*, pages 1–29, 2012.

[18] A. Rosenfeld, I. Zuckerman, A. Azaria, and S. Kraus. Combining psychological models with machine learning to better predict peopleâĂŹs decisions. *Synthese*, 189:81–93, 2012.

[19] D. Silver and J. Veness. Monte-Carlo planning in large pomdps. In *NIPS '10*. 2010.

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

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

[22] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 1998.

[23] M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.

[24] F. Wu, S. Zilberstein, and X. Chen. Online planning for ad hoc autonomous agent teams. In *IJCAI*, 2011.