To appear in *Proceedings of the Twenty-First European Conference on Artificial Intelligence* (ECAI), Prague, Czech Republic, August 2014.

Communicating with Unknown Teammates

Samuel Barrett[1] and **Noa Agmon**[2] and **Noam Hazon**[3] and **Sarit Kraus**[2,4] and **Peter Stone**[1]

Abstract.
Past research has investigated a number of methods for coordinating teams of agents, but with the growing number of sources of agents, it is likely that agents will encounter teammates that do not share their coordination methods. Therefore, it is desirable for agents to adapt to these teammates, forming an effective *ad hoc team*. Past ad hoc teamwork research has focused on cases where the agents do not directly communicate. However when teammates *do* communicate, it can provide a valuable channel for coordination. Therefore, this paper tackles the problem of communication in ad hoc teams, introducing a minimal version of the multiagent, multi-armed bandit problem with limited 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 following unknown behaviors even when its models of these teammates are imperfect.

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, e.g. for bidding in ad auctions. With this increase in agents comes an increase in their interactions and the number of companies and laboratories creating these agents. Therefore, there is a growing need for agents to be able to cooperate with a variety of different teammates. This need motivates the area of *ad hoc teamwork*, where agents are evaluated based on their ability to cooperate with a variety of teammates. Stone et al. [16] define ad hoc teamwork problems as problems in which a team cannot pre-coordinate its actions and introduce an algorithm for evaluating ad hoc team agents.

Past work on ad hoc teamwork has focused on the case where the ad hoc agent cannot (or does not) directly communicate to its teammates and can only coordinate by observing its teammates' actions. However, in an increasingly interconnected world, this lack of reasoning about communication is a missed opportunity. Therefore, the focus of this work is to show that when there is some form of limited communication using a common language, an agent can influence its teammates to improve the performance of the team. It is important to consider that while the ad hoc agent can choose what messages to send, it cannot control how they will be interpreted. Thus, the goal is to find the optimal messages to send and actions to select in order to influence the team to achieve the best total reward.

This paper makes three main contributions. First, it introduces a minimal domain for investigating teammate communication based on a multi-armed bandit scenario. Second, it proves that when its teammates fulfill some assumptions, optimal behaviors can be found in polynomial time for several scenarios (with two Bernoulli actions and three types of messages). Third, the paper evaluates an empirical planning algorithm based on Upper Confidence bounds for Trees (UCT) which extends to problems not covered in the theoretical analysis. Thus, this paper shows that ad hoc agents can optimally learn about their environment and their teammates while both acting in the world and communicating with their teammates. This learning is tractable and can be performed in polynomial time in terms of the problem parameters. In addition, even when it has imperfect assumptions about its teammates, an ad hoc agent can still learn and adapt so as to enable its team to perform effectively.

[1] University of Texas at Austin, {sbarrett,pstone}@cs.utexas.edu
[2] Bar-Ilan University, {agmon,sarit}@cs.biu.ac.il
[3] Ariel University, noamh@ariel.ac.il
[4] University of Maryland

2 Background and Problem Description

This paper introduces a multiagent, multi-armed bandit problem that allows limited communication. The multi-armed bandit setting is a fundamental problem in single agent reinforcement learning [18], and a bandit setting without communication has been used to study ad hoc teamwork in the past [17]. It is chosen here to serve as a minimal decision making domain that exhibits the necessary properties for investigating communication with unknown teammates.

The multi-armed bandit setting is a useful abstraction for many decision making scenarios. For example, consider a scenario in which a number of robots are deployed to transport supplies following a disaster. These robots must repeatedly carry supplies along one of a few possible routes which vary in their speed and safety. In this setting, selecting a route corresponds to pulling an arm. It is desirable for these robots to share their knowledge about the routes, but this communication takes time and is limited to whatever messages their teammates understand. A robot that is adept at reasoning about ad hoc teamwork should adapt to its teammates' usage of these routes and help the team select the best routes. This research moves towards this goal in the bandit setting.

2.1 Ad Hoc Teamwork

While general multiagent research focuses on creating a coordinated team to perform tasks, in ad hoc teamwork the goal is to create agents that can cooperate with a variety of possible teammates [16]. Specifically, we assume that there are several existing teams of agents that can accomplish the task, and we want to create an agent that can fit into any of these teams. Compared to general teamwork research, the difference is that these teams cannot be altered by us; we can only design a single agent that should adapt to any of these teams. One might assume that the best behavior of the agent is to match the behavior of its teammates. However, matching their behavior may be undesirable when the agent has access to additional knowledge or better algorithms.

2.2 Models

The Markov Decision Process (MDP) is a useful model for repeated decision making tasks. An MDP is a 4-tuple $M = (S, A, P, R)$ where S is a set of states, A is the set of actions, $P(s, a, s') = \Pr(s_{r+1} = s' | s_r = s, a_r = a)$ 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 function. In an MDP, the goal is to find an optimal policy $\pi^*(s)$ that selects actions that maximize the long term expected reward. 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 [13].

An extended version of this model known as the Partially Observable Markov Decision Process (POMDP) is also used in our analysis. In this model, the agent cannot directly observe its true state s. Instead, it receives imperfect observations of the underlying state, $\Omega(s) = o \in O$, where O is the set of possible observations. The underlying states and transitions remain unchanged from the original MDP, as does the agent's goal of maximizing the reward. However, the agent's task is harder because it must reason about the true state.

The difficulty of solving a POMDP is bounded by the size of the δ-covering of its belief space. A belief state is the probability distribution over states that the agent may be in. The belief space is a combination of what the agent can directly observe about the world and its beliefs about the hidden state of the world. 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 with radius δ filling the space. The *covering number* is the size of the smallest δ-covering. From Theorem 5 in [10], it is known that a policy that performs within ϵ of the optimal policy for a POMDP can be found in polynomial time in terms of the size of a given δ-cover set B where $\delta = \text{poly}(\epsilon)$. This theorem shows this result for the infinite horizon, discounted rewards case, chosen because the discount factor bounds the expected total reward. However, these results extend to our finite horizon setting given that expected total reward is bounded by the number of rounds and agents $(n + 1)R$.

2.3 Bandit Setting

We formally define the bandit problem in this paper as the tuple $G = (\mathbb{A}, \mathbb{C}, \mathbb{P}, R)$ where $\mathbb{A}$ is a set of two arms $\{arm_0, arm_1\}$ with Bernoulli payoff distributions, returning either 0 or 1, $\mathbb{C} = \{(c_i, cost(c_i))\}$ is a finite set of possible communications and their costs, $\mathbb{P}$ denotes the players in the problem with $|\mathbb{P}| = n + 1$ with n of the agents being a pre-designed team, and R 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:

- **obs** – Send the agent's last selected arm and payoff
- **mean**$_i$ – Send the agent's observed mean and number of pulls for arm_i
- **suggest**$_i$ – Suggest that the teammates pull arm_i

These message types are understood by all of the agents. In the action phase, each agent chooses an arm and receives a payoff. The team's goal is to maximize the sum of payoffs minus the communication costs. We use arm_* to denote the arm with the highest payoff. Note that the results in this paper can be generalized to any number of fixed arms, other discrete distributions, and other message types.

2.4 Teammate Behavior

If the teammates have different knowledge from each other, this problem can be exponentially hard. However, we simplify the problem by assuming that the ad hoc agent's teammates form an existing team, and therefore are tightly coordinated. Therefore, this team's behavior can be described as a function of the team's total number of pulls and successes of each arm as they pool this knowledge using the message types provided above. The team's actions also rely on the ad hoc agent's pulls and successes that it has communicated, combining all of the team's pulls and successes as well as the ad hoc agent's into a single estimate of the quality of each arm. While the assumption that all of the knowledge is shared via communication may not always hold, it may hold in many scenarios. Section 5 considers agents that do not satisfy this assumption, although the ad hoc agent still uses this assumption to simplify planning. Each teammate's behavior consists of an action function, *act*, and a communication function, *comm*. These functions specify the probability of the agent selecting arms or sending messages.

3 Applying the Models

When the ad hoc agent knows its teammates' behaviors, it can model the bandit problem as an MDP. The MDP's state is composed of the pulls and observations of the ad hoc agent's teammates as well as the messages it has sent. Let $K = (p_0, s_0, p_1, s_1)$ be the knowledge about the arms where p_i and s_i are the number of pulls and successes of arm_i. Then, the state is given by the vector $(K_t, K_a, K_c, r, phase, sugg)$, where K_t is the team's knowledge from their pulls, K_a is the ad hoc agent's knowledge from its pulls, K_c is the knowledge that the ad hoc agent has communicated, r is the current round number, *phase* is the phase of the round, and *sugg* is the ad hoc agent's most recent suggestion. As the n agents on the team are coordinated, their actions depend on K_t and K_c and *not* directly on K_a. We split K_c from K_t to model how the ad hoc agent's messages will affect the team. For example, if the ad hoc agent already communicated an observation, communicating its observations of the same arm will replace its teammates' memory of this observation.

Next, we reason about the number of states and actions of the resulting MDP. Given that there are R rounds and n teammates, p_i and s_i in K_t are each bounded by nR, p_i and s_i in both K_a and K_c are each bounded by R. The round r is bounded by R, and there are 2 possible phases of a round. Finally, the most recent suggestion *sugg* takes on one of 3 values (arm_0, arm_1, or none). Therefore, the state space has at most $(nR)^4 \cdot R \cdot R^4 \cdot R^4 \cdot 2 \cdot 3 = 6n^4R^{13}$ states. While this sounds large, a polynomial bound means that the problem is tractable and existing algorithms can be applied.

The actions of the MDP are the possible arms and the available 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. Let ϵ represents no message, $o \in \{\epsilon, \text{obs}\}$, $m \in \{\epsilon, \text{mean}_0, \text{mean}_1\}$, and $s \in \{\epsilon, arm_0, arm_1\}$. In the communication phase, the ad hoc agent can send one message of each type, resulting in an action of the form (o, m, s). Therefore, there are $2 \cdot 3 \cdot 3 = 18$ actions in the communication phase, and 2 in the action phase.

The transition function P is composed of the *act* and *comm* functions, the arms' payoff distributions, and the effects of the ad hoc agent's messages. Specifically, *act* and the ad hoc agent's chosen arms affect the p_i values in K_t and K_a respectively, while the arm distributions specify how these actions affect the s_i values in K_t and K_a. The ad hoc agent's messages and K_a define the changes to K_c and *sugg*. The reward function R is a combination of the rewards coming from the arms and the costs of communication.

4 Theoretical Analysis

To solve the general problem of ad hoc teamwork in the bandit domain, we first tackle the simplest version of the problem and then progressively relax our assumptions. Specifically, Sections 4.1–4.4 show that a number of ad hoc team problems in the bandit setting are provably tractable, as summarized in Table 1. Specifically, these results prove that ad hoc team agents can plan approximately optimal behaviors involving communication without taking more than polynomial time.

Knowledge of Teammates	Teammate Type	Knowledge of Environment	Solution Type	Section
Known	Stochastic	Known	Exact	4.1
Finite Set	Deterministic	Known	Exact	4.2
Parameterized Set	Stochastic	Known	Approx.	4.3
Parameterized Set	Stochastic	Unknown	Approx.	4.4

Table 1: Problems that are solvable in polynomial time.

4.1 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, it knows the true stochastic behavior (*act* and *comm*) of its teammates. Therefore, the ad hoc agent has a full model of the problem described in Section 3. It is possible to find the optimal solution to an MDP using DP in time polynomial in the MDP's size, which is polynomial in the number of rounds R and teammates n. Therefore, Proposition 1 directly follows.

Proposition 1. *An ad hoc agent that knows the true arm distributions and its teammates' behaviors can calculate its optimal behavior for maximizing the team's shared payoffs in poly(R,n) time.*

4.2 Teammates from a Finite Set

In this section, we relax the constraint on knowing the teammates' behaviors. Rather than knowing the specific behavior of its teammates, the ad hoc agent instead 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 ad hoc teamwork [3] and using machine learning with psychological models [14] suggests that a small number of behaviors can represent the spread of possible behaviors.

In general, this finite set of behaviors can vary, but in this analysis, we consider two types of teammates: 1) greedy agents and 2) ones that choose arms using confidence bounds in the form of UCB1 [1]. The UCB1 agents select actions using

$$arm = \operatorname*{argmax}_i \frac{s_i}{p_i} + c\sqrt{\frac{\ln(p_0+p_1)}{p_i}} \tag{1}$$

where $c = 1$. The ad hoc agent is given a prior probability distribution over teams following either of these behaviors. The teammates are assumed to use the ad hoc agent's communicated pulls when selecting their actions. Additionally, we assume that these teammates share all information with each other and send messages that the ad hoc agent can hear, but these messages do not reveal the teammates' behaviors.

To analyze this problem, we add the ad hoc agent's beliefs about its teammates into the state space that the agent plans over. As the teammates are deterministic, there are three possibilities for the belief space: both models are still possible, only the greedy model is possible, or only the UCB1 model 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 $18n^4R^{13}$. In general, the increase in size is $2^k - 1$ where k is the number of models, but we assume that k is fixed and not a problem parameter. 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. Proposition 2 follows directly from this reasoning.

Proposition 2. *An ad hoc agent that knows the true 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(R,n) time.*

4.3 Teammates from a Continuous Set

In this section, we further relax the constraints on the teammates' behaviors, considering a continuous set of stochastic behaviors rather than the discrete set of deterministic behaviors used in the last section. 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 the scaling factor of the confidence bound in Eq. 1. Therefore, the ad hoc agent must maintain a belief distribution over values of ε, values of c, and p the probability of the teammates being ε-greedy. The ad hoc agent is given the prior knowledge that ε, c are uniformly distributed over $[0, 1]$, and it starts with an initial estimate of p. While we use two models for simplicity, this analysis can be extended for any fixed number of parameterized models.

To analyze this problem, we model the problem as a POMDP as discussed in Section 2.2. The transition function for the fully observable state variables remains the same as in the original MDP. In this setting, the belief space has three partially observed values: ε, c, and p the probability of the teammates being ε-greedy versus UCB(c). The value of p is updated using Bayes' rule given the probability of the models predicting the observed actions, and the updates to the probability distributions of ε and c are described in Lemma 1. The remainder of the POMDP remains as defined above.

In Lemma 1 and Theorem 1, 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, which defines the difficulty of solving the POMDP as discussed in Section 2.2.

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

Proof. The resulting size of the δ-covering is a product of the contributing factors. These factors come from the underlying MDP state s, ε, c, and p. Using Proposition 1 of [10], we know that the fully observed state variables result in a multiplicative factor that is polynomial in R and n. Therefore, since the ad hoc agent directly observes s, it only results is a factor of poly(R, n). The probability of the two models p 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 nR. Therefore, the probability distribution over ε can be represented using a factor of size $(nR)^2$.

The parameter c has a uniform prior, and UCB agents select arms using Eq. 1, combining the communicated and team's pulls by setting $p_j = p_j^t + p_j^c$ and $s_j = s_j^t + s_j^c$. The teammates will only select the lower arm when c is above a certain value and the higher arm when c is below a certain value. Therefore, the top and bottom ranges of c can be updated using linear programming from observing their actions. Note that the posterior remains uniform; only the range changes. 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 $1/\delta^2$. Multiplying all of of these factors results in a δ-covering of size poly$(R, n, 1/\delta)$. □

As discussed in Section 2.2, a POMDP can be solved approximately in polynomial time given a covering set. Given this result and Lemma 1, Theorem 1 follows directly.

Theorem 1. *Consider an ad hoc agent that can observe its teammates' actions, knows the true arm distributions, and knows that its teammates are drawn from a known, continuous set of ε-greedy and UCB teammates. This agent can calculate an η-optimal behavior in poly$(n, R, 1/\eta)$ time.*

4.4 Unknown Arms

The previous sections assumed that the ad hoc agent already knew the underlying distributions of the arms (i.e. the POMDP's transition function), but in many cases the ad hoc agent may not have this information. Therefore, it is desirable for the ad hoc agent 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. We again assume that the ad hoc agent knows its teammates' pulls and results, either by observing them directly or by listening to its teammates' messages.

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

Theorem 2. *Consider an ad hoc agent that does not know the true arm distributions, but has a uniform prior over their success probability, knows that its teammates' behaviors are drawn from a continuous set of ε-greedy and UCB teammates, and can observe the results of their actions. This agent can calculate an η-optimal behavior in poly$(n, R, 1/\eta)$ time.*

5 Empirical Evaluation

This section investigates whether the problem is empirically tractable in addition to being theoretically tractable. The results show that modeling the problem as a (PO)MDP and planning using this model significantly improves the performance of the team compared to several intuitive baseline behaviors in several scenarios. In this setting, calculating the exact optimal behavior becomes impractical as the problem size grows. Therefore, in the empirical setting, we use Partially Observable Monte-Carlo Planning (POMCP) [15]. 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 is not guaranteed to find an optimal solution given our limited computation, our results show that it plans an effective behavior in our setting.

5.1 Methods

POMCP is a Monte Carlo Tree Search (MCTS) algorithm that is based on the Upper Confidence bounds for Trees (UCT) algorithm [11]. Specifically, POMCP starts from the current state and performs a number of simulations until reaching the end of the problem. In the simulations, the agent selects its actions using upper confidence bounds on its current estimates of the available actions. The results of pulling arms are randomly sampled given the arms' distributions. For its teammates, the ad hoc agent plans as if they use either the ε-greedy or the UCB algorithms. To model the effects of sending suggestions, agents are given a probability of following the most recent suggestion, 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).

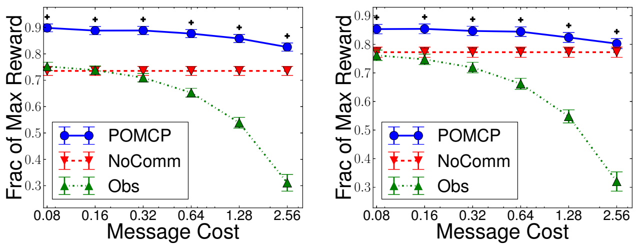
5.2 Results

The evaluations use 100 trials with teams where ε, c, and the arms' success probabilities are selected randomly uniformly between 0 and 1. This randomness is fixed across the different ad hoc agent behaviors to allow for paired statistical tests. As the ad hoc agent does not know its teammates' behaviors, it initializes its beliefs by sampling both behavior types with random parameter values. The results are average team rewards normalized by the average reward if all agents repeatedly pull the best arm. Statistical significance is tested using a Wilcoxon signed-rank test with $p < 0.05$, denoted by "+" 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** - Pulls the best arm and does not communicate
- **Obs** - Pulls the best arm and sends its last observation
- **POMCP** - Plans using POMCP

Match, NoComm, and Obs serve as baselines. Pulling the best arm and sending 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. Unless otherwise specified, there are 10 rounds and 7 teammates and we use 3 arms to test how our approach scales to bigger problems than are theoretically proven. Furthermore, the costs for sending messages are known by all agents and randomly selected for each run. These costs are sampled from $[0, m|c|]$, where $|c|$ is size of the message (3 for mean, 2 for obs, and 1 for sugg) and $m = 0.75$ unless otherwise specified.



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

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

Figure 1 presents the results when the ad hoc agent encounters the problem discussed in Section 4.3, 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. The performance of POMCP diminishes as the cost of messages rises because affecting the teammates becomes more costly. However, the POMCP approach will plan not to communicate when the message costs get too high. The results are similar when the ad hoc agent knows its teammates' true behavior, rather than assuming that both types are possible.

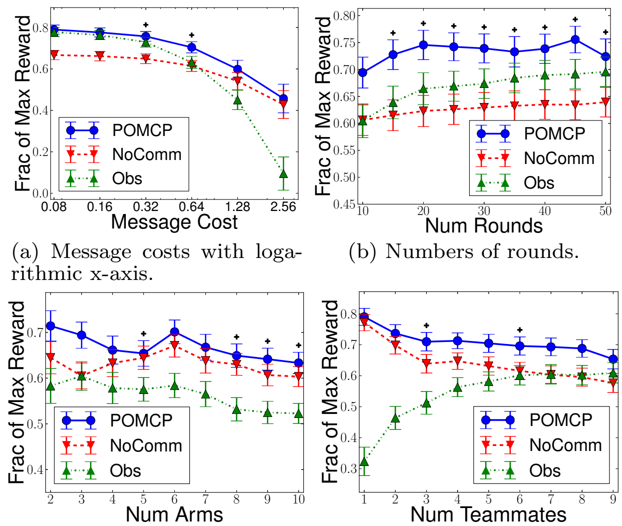
5.3 Externally-created Teammates

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 serve as a sample of the variety of teammates an ad hoc agent might encounter in real scenarios. 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 without considering ad hoc teamwork. These agents use the same three 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 via communication. However, the externally-created agents do not always choose to share this information, breaking this assumption. In addition, the externally-created agents follow a variety of behaviors, serving as a diverse set of imperfect agents that may be created by different designers attempting to solve real problems. We specifically did not analyze their behaviors to prevent biasing the design of our ad hoc agent. In our planning, we still assume that the teammates form a coordinated team of ε-greedy and UCB agents for ease of planning, and our results show that this approach is effective despite its inaccuracies.

Given that the externally-created teams quickly converge to the best arm, all approaches perform similarly with these teammates. Therefore, we investigate the worst case scenario for the team: the best arm performs poorly early in the scenario, possibly misleading the team into not pulling the arm later. To create this setting, 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. In this setting, both the teammates and the ad hoc agent are unaware of the initial bias of the arm. Therefore, this test evaluates how well the ad hoc agent can use its prior knowledge to correct the misinformation its teammates have observed.

Figure 2 shows the results with externally-created agents. In these evaluations, we test the sensitivity of the agent to various problem parameters, investigating under which conditions POMCP outperforms the baselines. Note that the message costs are also applied to the externally-created teammates, which know the current message costs, so the performance of NoComm is now affected by message costs.



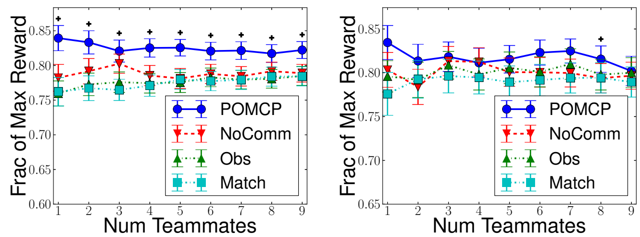
(a) Message costs with logarithmic x-axis. (b) Numbers of rounds. (c) Numbers of arms. (d) Numbers of teammates.

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

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 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.

5.4 Unknown Arms

While the previous sections investigated how an ad hoc agent can cooperate with a variety of teammates, the ad hoc agents were provided with prior knowledge about the underlying distributions of the arms. This section investigates a scenario in which the ad hoc agent is also uncertain about the true payoffs of the arms and must simultaneously learn about the world and its teammates, as discussed in Section 4.4. We still assume that the ad hoc agent can observe the payoffs of its teammates' actions, for example by listening to their messages. Figure 3 shows the results for this scenario. When using the POMCP behavior, 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 chooses arms ε-greedily, with $\varepsilon = 0.1$, because it does not know the true best arm. To encourage more sharing, 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 cooperating using communication.



(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.

6 Related Work

Multiagent teams have been well studied, with previous research mainly focusing on creating standardized methods for coordination and communication. The SharedPlans framework assumes common recipes exist across teammates [8]. In STEAM [19], team members build a partial hierarchy of joint actions. The TAEMS framework [9] consists of a hierarchy of rules, where agents coordinate through common groups, tasks, and methods. While these algorithms are effective in many settings, they assume that all teammates are using the same teamwork mechanism.

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 [12], 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 and propose one way to categorize ad hoc teamwork problems. A more theoretical approach is Wu et al.'s work [20] into ad hoc teams using stage games and biased adaptive play.

Goldman et al. [7] investigate learning to communicate. However, they assume very little about the meaning of messages and therefore learn over a long period of time, as opposed to the faster adaption enabled by our assumptions of the messages' meanings. Other work investigates agents that explicitly model and reason about their opponent's beliefs in the form of interactive POMDPs [6] and interactive dynamic influence diagrams (I-DIDs) [5].

7 Conclusion

Past research on ad hoc teamwork has largely focused on scenarios in which the ad hoc agent cannot (or does not) directly communicate with its teammates. This work addresses this gap by introducing an agent that reasons about communicating in ad hoc teams. In order to theoretically analyze this problem, we introduce a minimal domain that allows for communication. Then, we prove that an agent can optimally plan how to cooperate with a its teammates using only polynomial computation, even when it may encounter an infinite variety of teammates in the form of parameterized behavior models. Furthermore, we empirically evaluate an algorithm for planning in these problems and show that it allows an agent to adapt to teammates that were created by a variety of developers, even when the agent has only imperfect models of its teammates. Finally, we also empirically show that the ad hoc agent can effectively learn about its environment and its teammates simultaneously.

ACKNOWLEDGEMENTS

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). This research is supported in part by the U.S. Army Research Laboratory and the U.S. Army Research Office under grant number W911NF-08-1-0144 and by ERC grant #267523.

REFERENCES

[1] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer, 'Finite-time analysis of the multiarmed bandit problem', *Machine Learning*, **47**, 235–256, (May 2002).

[2] Samuel Barrett and Peter Stone, 'An analysis framework for ad hoc teamwork tasks', in *AAMAS '12*, (June 2012).

[3] Samuel Barrett, Peter Stone, Sarit Kraus, and Avi Rosenfeld, 'Teamwork with limited knowledge of teammates', in *AAAI*, (July 2013).

[4] Michael Bowling and Peter McCracken, 'Coordination and adaptation in impromptu teams', in *AAAI*, (2005).

[5] Prashant Doshi and Yifeng Zeng, 'Improved approximation of interactive dynamic influence diagrams using discriminative model updates', in *AAMAS '09*, (2009).

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

[7] Claudia V. Goldman, Martin Allen, and Shlomo Zilberstein, 'Learning to communicate in a decentralized environment', *Autonomous Agents and Multi-Agent Systems*, **15**(1), (2007).

[8] B. Grosz and S. Kraus, 'The evolution of SharedPlans', in *Foundations and Theories of Rational Agency*, (1999).

[9] Bryan Horling, Victor Lesser, Regis Vincent, Tom Wagner, Anita Raja, Shelley Zhang, Keith Decker, and Alan Garvey. The TAEMS White Paper, January 1999.

[10] David Hsu, Wee Sun Lee, and Nan Rong, 'What makes some POMDP problems easy to approximate?', in *NIPS*, (2007).

[11] Levente Kocsis and Csaba Szepesvari, 'Bandit based Monte-Carlo planning', in *ECML '06*, (2006).

[12] Somchaya Liemhetcharat and Manuela Veloso, 'Modeling mutual capabilities in heterogeneous teams for role assignment', in *IROS '11*, pp. 3638 –3644, (2011).

[13] Martin L Puterman and Moon Chirl Shin, 'Modified policy iteration algorithms for discounted Markov decision problems', *Management Science*, **24**(11), 1127–1137, (1978).

[14] Avi Rosenfeld, Inon Zuckerman, Amos Azaria, and Sarit Kraus, 'Combining psychological models with machine learning to better predict people's decisions', *Synthese*, **189**, 81–93, (2012).

[15] David Silver and Joel Veness, 'Monte-Carlo planning in large POMDPs', in *NIPS '10*, (2010).

[16] Peter Stone, Gal A. Kaminka, Sarit Kraus, and Jeffrey S. Rosenschein, 'Ad hoc autonomous agent teams: Collaboration without pre-coordination', in *AAAI '10*, (July 2010).

[17] Peter Stone and Sarit Kraus, 'To teach or not to teach? Decision making under uncertainty in ad hoc teams', in *AAMAS '10*, (May 2010).

[18] Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, USA, 1998.

[19] M. Tambe, 'Towards flexible teamwork', *Journal of Artificial Intelligence Research*, **7**, 83–124, (1997).

[20] Feng Wu, Shlomo Zilberstein, and Xiaoping Chen, 'Online planning for ad hoc autonomous agent teams', in *IJCAI*, (2011).