In *The Ninth International Conference on Autonomous Agents and Multi-Agent Systems* (AAMAS), May 2010.

To Teach or not to Teach? Decision Making Under Uncertainty in Ad Hoc Teams

Peter Stone
Department of Computer Science
The University of Texas at Austin
Austin, TX, 78712-0233 USA
pstone@cs.utexas.edu

Sarit Kraus
Department of CS
Bar-Ilan University
Ramat Gan, 52900 Israel
Inst. for Advanced CS
University of Maryland
College Park, MD 20742
sarit@cs.biu.ac.il

ABSTRACT

In typical multiagent *teamwork* settings, the teammates are either programmed together, or are otherwise provided with standard communication languages and coordination protocols. In contrast, this paper presents an *ad hoc team* setting in which the teammates are not pre-coordinated, yet still must work together in order to achieve their common goal(s). We represent a specific instance of this scenario, in which a teammate has limited action capabilities and a fixed and known behavior, as a finite-horizon, cooperative k-armed bandit. In addition to motivating and studying this novel ad hoc teamwork scenario, the paper contributes to the k-armed bandits literature by characterizing the conditions under which certain actions are potentially optimal, and by presenting a polynomial dynamic programming algorithm that solves for the optimal action when the arm payoffs come from a discrete distribution.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Coherence and coordination, Multiagent systems*

General Terms

Algorithms, Performance, Theory

Keywords

Autonomous agents, multiagent systems, coordination

1. INTRODUCTION

The year is 2009 and a robot is built to collect discarded aluminum cans from an ocean beach for recycling. There are two beaches in its neighborhood and when it is low tide, the robot can work at one of the two beaches. After collecting the cans, it returns to its base where it deposits the cans into a machine that reports back the number that were collected, and the robot recharges its batteries. The distance to the base and recharge time is such that it needs to skip the next tide cycle. Thus it is able to go to a beach roughly once a day. Meanwhile, the cans on the beach that it does not go to (and on both beaches when it is recharging) are either swept out by the high tide or collected by scavengers: the yield from each trip does not depend on which beach was most recently visited.

Because the robot developers do not know ahead of time how popular the beaches will be in any given month, they program the robot to be adaptive. Under the assumption that popularity changes on roughly a month-to-month basis, but holds constant (with some noise) within the month, they use the following algorithm:

- The robot starts each month by trying each beach once.
- Thereafter, it goes to the beach that has yielded the most cans per trip on average that month.

This strategy is not optimal: getting a low return from a beach the first time could lead to that beach never being tried again that month. But it is reasonable, and works well enough given that the beaches tend to be roughly equal in popularity and the value of cans is relatively low. That is, the cost of being slightly suboptimal is low.

Fast forward now to the year 2015. The robot described above has been doing its job consistently and robustly for six years. The ability to reprogram the robot has long been lost either due to changes in programming languages (think of the legacy COBOL programs at the time of the Y2K bug) or a malfunctioning connector on the robot. But the robot is useful and nobody would think of discarding it. In fact, the cost of aluminum has gone up significantly, and the community is no longer willing to let a whole tide cycle go by without collecting cans from one of the beaches. Furthermore, they are now more interested in having optimal decisions made so that every possible piece of aluminum is collected.

You are therefore hired to build a new robot (call it *Robot B*) to help out the existing one (*Robot A*). So that they can share the recharging station, *Robot B* will collect cans while *Robot A* is recharging and will recharge while *Robot A* is collecting. Each robot can observe which beach the other went to and how many cans it collected. However, since *Robot A* was built without any knowledge that there would be a *Robot B*, they cannot communicate beyond that.

Furthermore, the world has changed in two additional ways. First, there is now a third beach in the area that is more popular than the original two. *Robot B* is able to go to any of the three beaches, however *Robot A* is not aware of the new one (or is unable to navigate the more rugged terrain to get there). Second, there is now a municipal website that predicts the average popularity of the three beaches for a given month. You can equip *Robot B* with the ability to check this website, thus informing it as to which beaches are

Cite as: To Teach or not to Teach? Decision Making Under Uncertainty in Ad Hoc Teams, Peter Stone and Sarit Kraus, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. XXX-XXX.

likely to yield the most cans. However, as already indicated, you cannot reprogram *Robot A* to use this website.

If *Robot B* were acting alone, you could program it always to go to the beach that the website reports will be busiest (in this case the new beach). However it could instead provide more data to *Robot A* by going to one of the other beaches so that *Robot A* can observe how many cans it brings back. As will be shown below, doing so will sometimes be *Robot B*'s optimal action: it will be better off *teaching Robot A* rather than *exploiting* its knowledge of the most popular beach.

The above fictional setting can be formalized as a finite-horizon (because the problem resets every month) cooperative k-armed bandit [15] in a way that, to the best of our knowledge, has never been considered before in the literature. The formalism can be applied to any multiagent decision-making setting that shares the essential characteristics of the scenario described above, and can also be generalized to a much broader class of scenarios which we refer to as *ad hoc teamwork* settings.

In this paper, we characterize the conditions under which certain actions are potentially optimal in such a finite-horizon, cooperative k-armed bandit, and we present a dynamic programming algorithm that solves for the optimal action when the payoffs come from a discrete distribution. For Gaussian distributions we present some theoretical and experimental results and identify an open problem. While k-armed bandits are often used to study the exploration versus exploitation challenge, nobody has previously considered a multiagent cooperative setting in which the agents have different knowledge states and action capabilities. Thus our formalization is simultaneously a practical method for multiagent team decision-making, and a novel contribution to the literature on k-armed bandits.

Additionally, this paper takes an initial step towards the long-term goal of creating a fully robust ad hoc team player. We define an ad hoc team setting as one in which multiple agents with different knowledge and capabilities find themselves in a situation such that their goals and utilities are perfectly aligned, yet they have had no prior opportunity to coordinate. In addition to the can-collecting example above, ad hoc teams may arise in disaster rescue settings where people bring different robots together and they need to coordinate quickly, or indeed among any robots or software agents that have been programmed by different groups and/or at different times. An agent that is to succeed in an ad hoc team setting must be prepared to cooperate with many types of teammates: those that are more mobile and those that are less mobile; those with better sensing capabilities and those with more limited capabilities; those with known behaviors and those without. In the long-run, a good team player will need strategies for dealing with all these teammate types. In this paper, we use our novel version of the k-armed bandit formalism to focus on the ability to cooperate with a class of teammates that are less capable, have less knowledge, and have fixed and known behavior.

The remainder of the paper is organized as follows. First, in Section 2, we formalize the situation of study as an instance of a 3-armed bandit problem and specify all of our assumptions. Then, in Sections 3–5, we present our detailed analysis, algorithms, and results. Section 6 considers a generalization to more than 3 arms, Section 7 presents related work and Section 8 concludes.

2. FORMALISM

The k-armed bandit problem [15] is a much-studied problem in sequential decision making. The basic setting is as follows. At each time step, a learning agent selects one of the k arms to pull. The arm returns a payoff according to a fixed, but generally unknown, distribution. The agent's goal is to maximize the sum of the payoffs it receives over time. The k-armed bandit is a classic setting for studying the exploration vs. exploitation problem: at any given time, the agent could greedily select the arm that has paid off the best so far, or it could select a different arm in order to gather more information about its distribution. It is also the basis for reinforcement learning theory, representing the stateless action selection problem [19].

In order to study the ad hoc team problem laid out in Section 1, we extend the standard setting to include two distinct agents, known as the *teacher* (*Robot B*) and the *learner* (*Robot A*), who select arms (choose beaches) alternately, starting with the teacher. We initially consider a bandit with just three arms such that the teacher is able to select from any of the three arms, while the learner is only able to select from among the two arms with the lower expected payoffs. We consider the fully cooperative case such that the teacher's goal is to maximize the expected sum of the payoffs received by the two agents over time (the teacher is risk neutral). Specifically, we make the following assumptions, all of which align with our can-collecting example:

- The payoff distributions of all arms are fully known to the teacher, but unknown to the learner.
- The learner can only select from among the two arms with the lower expected payoffs.
- The results of all actions are fully observable (to both agents).
- The number of rounds (actions per agent) remaining is finite and known to the teacher.
- The learner's behavior is fixed and known: it acts greedily, always selecting the arm with the highest observed sample average so far. Any arm that has never been pulled is assumed to have a sample average of ∞. Thus, the learner always prefers selecting an arm that has not been selected previously. If there is more than one such arm, it selects randomly from among them. This assumption reflects optimism in the face of uncertainty on the part of the learner. (optimistic initialization).

The teacher must then decide whether to do what is best in the short term, namely pull the arm with the highest expected payoff; or whether to increase the information available to its teammate, the learner, by pulling a different arm. Note that if the teacher were acting alone, trivially its optimal action would be to always pull the arm with highest expected payoff.

By these assumptions, the learner is both less capable and less knowledgeable than the teacher, and it does not understand direct communication from the teacher. It is tempting to think that we should begin by improving the learner. But in the ad hoc team setting, that is not an option. The learner "is what it is" either because it is a legacy agent, or because it has been programmed by others. Our task is to determine the teacher's best actions *given* such learner behavior.

We use the following notation for the three arms. The learner selects between Arm_1 and Arm_2, while the teacher can additionally choose Arm_*. While we consider two different forms of distributions for the payoffs, throughout the paper we use the following notation:

- μ_i is the expected payoff of Arm_i ($i \in \{1, 2, *\}$).
- n_i is the number of times Arm_i has been pulled (observed) in the past.
- m_i is the cumulative payoff from all the past pulls of Arm_i.
- $\bar{x}_i = \frac{m_i}{n_i}$ is the observed sample average so far.
- r is the number of rounds left.

Throughout the paper we assume that $\mu_* > \mu_1 > \mu_2$. If μ_* is not the largest, then the teacher's choice is trivially to always select the arm with the largest expected payoff. The ordering of Arm_1 and Arm_2 is without loss of generality. In this setting, the question we ask is, which arm should the teacher pull, as a function of r and all the n_i, $\bar{x}_i$, and Arm_i payoff distributions (including μ_i)?

3. ARBITRARY DISTRIBUTION ARMS

In this section, we present theoretical results that apply regardless of the forms of the distributions of the payoffs from the three arms.

3.1 The teacher should consider pulling Arm_1

First, to understand that the problem specified in Section 2 is not trivial, we show that there are situations in which the teacher should not greedily optimize its short-term payoff by pulling Arm_*, but rather should increase the amount of information available to the learner by pulling Arm_1.

In fact, even with just one round remaining ($r = 1$), it is not difficult to construct such a case. For example, suppose that $\mu_* = 10, \mu_1 = 9, \mu_2 = 5, \bar{x}_1 = 6, \bar{x}_2 = 7, n_1 = n_2 = 1$. Suppose further that the distribution of payoffs from Arm_1 is such that the probability of obtaining a value greater than 8 is $\eta > \frac{1}{2}$. Thus with probability η, after an agent selects Arm_1, its sample average will be greater than $\bar{x}_2$.

Should the teacher select Arm_*, then the learner will select Arm_2 (because $\bar{x}_1 < \bar{x}_2$), yielding an expected total payoff during the round of $\mu_* + \mu_2 = 15$. On the other hand, should the teacher select Arm_1, there is a greater than 50% chance that the learner will select Arm_1 as well. The expected payoff is then $\mu_1 + \eta\mu_1 + (1 - \eta)\mu_2 > 9 + \frac{9}{2} + \frac{5}{2} = 16$.

Therefore there are situations in which it is better for the teacher to pull Arm_1 than Arm_*. This paper is devoted to characterizing exactly what those situations are.

3.2 The teacher should never pull Arm_2

Second, we argue that the teacher should only consider pulling Arm_* or Arm_1. On the surface, this result appears obvious: why should the teacher pull Arm_2 just to prevent the learner from doing the same? In fact, there is a relatively straightforward proof that applies when $\bar{x}_1 < \bar{x}_2$ (similar to our proof of Theorem 3.2 below). However the proof of the fully general result that includes the seemingly simpler case that $\bar{x}_1 > \bar{x}_2$ is surprisingly subtle. Due to space constraints, we only sketch the proof below. The full proof is available in an online appendix.[1]

THEOREM 3.1. *It is never optimal for the teacher to pull Arm_2.*

Proof sketch. The proof uses induction on r.

Base case: $r = 1$. If the teacher starts by pulling Arm_2, the best expected value the team can achieve is $\mu_2 + \mu_1$. Meanwhile, if it starts with Arm_*, the worst the team expects is $\mu_* + \mu_2$. This expectation is higher since $\mu* > \mu_1$.

Inductive step: Assume that the teacher should never pull Arm_2 with $r - 1$ rounds left. Let π^* be the optimal teacher action policy that maps the states of the arms (their μ_i, n_i, and $\bar{x}_i$) and the number of rounds left to the optimal action: the policy that leads to the highest long-term expected value. Consider the sequence, S, that begins with Arm_2 and subsequently results from the teacher following π^*. To show: there exists a teacher action policy π' starting with Arm_* (or Arm_1) that leads to a sequence T with expected value greater than that of S. That is, the initial pull of Arm_2 in S does not follow π^*.

The underlying idea is that the sequence T should start with the teacher pulling Arm_* repeatedly, and tracking the values obtained by the learner to see if it can ever discern what the sequence S would have looked like after some number of rounds (it *simulates* sequence S). This may not be possible, for example if sequence S begins with a pull of Arm_1, whereas after the initial pull of Arm_2 in T, the values are such that Arm_1 is never pulled.

If the teacher ever *does* get to the point that all of the learner's pulls of Arm_1 and Arm_2 in T can be used in simulating S, then the teacher can mimic S from that point until it runs out of rounds (we can prove that the simulation necessarily ends with fewer rounds executed in S than in T). Then nothing that would have happened after the mimicking ended (that is that *will* happen in S) could have higher expected value than all the extra pulls of Arm_* that came before the mimicking started in T.

If, on there other hand, there is never a point that all the pulls of Arm_1 and Arm_2 can be used in the simulation, then sequence T must have more pulls of Arm_* and fewer pulls of Arm_2 than sequence S (which itself requires some care to prove rigorously).

Either way, the sequence T has higher expected value than sequence S, so the initial pull of Arm_2 in S was suboptimal. $\square$

Thus, when the teacher decides to teach the learner, it does so by pulling Arm_1. Pulling Arm_* can be thought of as exploiting, or maximizing short-term payoff. In the remainder of this paper, we sometimes refer to the teacher pulling Arm_1 as "teaching," and pulling Arm_* as "not teaching."

3.3 Never teach when $\bar{x}_1 > \bar{x}_2$

Third, we show that the teacher's choice is clear whenever $\bar{x}_1 > \bar{x}_2$. That is, if the current sample average of Arm_1 is greater than that of Arm_2 such that the learner will choose Arm_1 next, then the teacher should always choose Arm_*: it should not teach.

THEOREM 3.2. *When $\bar{x}_1 > \bar{x}_2$, it is always optimal for the teacher not to teach (to pull Arm_*).*

Proof sketch. The proof can be seen as a much simplified version of the proof of Theorem 3.1 in which simulating sequence S is always possible after two rounds. Specifically, if the teacher selects Arm_1 first, let the learner response (which depends on the result from Arm_1) be a. On the other hand, if the teacher begins by selecting Arm_* twice, then the learner will pull Arm_1 and a on its first two turns. Following that, the teacher could mimic the optimal sequence as if there were one additional round remaining (pretend r is

[1]See `http://www.cs.utexas.edu/~pstone/Papers/bib2html/b2hd-AAMAS2010-adhoc.html`

$r+1$). Nothing that could happen in the omitted last round has higher expected value than the additional two pulls of Arm_* at the beginning. $\square$

3.4 Do not teach when $n_1 = 0$ and/or $n_2 = 0$

When starting a new task such that the learner has no experience with any of its arms, the teacher should pull Arm_*: it should not teach. The proof proceeds similarly to the proof of Theorem 3.2. In fact, the proof generalizes to the statement that the teacher should never do what the student is about to do anyway.

4. DISCRETE DISTRIBUTION ARMS

In Section 3, we presented theoretical results that do not depend in any way on the form of the distributions governing the payoffs from the various arms: the teacher should never pull Arm_2, and it should only consider Arm_1 when $\bar{x}_1 < \bar{x}_2$. In this section and the next, we analyze when exactly the teacher should select Arm_1, which depends on the exact distributions of the payoffs. We first restrict our attention to *binary* distributions such that each Arm_i returns a 1 with probability p_i, and a 0 otherwise. Thus $\mu_i = p_i$, and m_i is the number of times the arm has yielded a payoff of 1 thus far. In this setting we can solve for the optimal teacher action using finite horizon dynamic programming. The algorithm generalizes to any discrete distribution. Note that in our can-collecting example from Section 1, the distribution is necessarily discrete since the number of cans collected is always a finite integer.

4.1 $\bar{x}_1 < \bar{x}_2, r = 1$

To develop intuition, we begin by considering what the teacher should do when $r = 1$ (one action remaining for each agent). As shown in Section 3, the teacher should never teach when $\bar{x}_1 > \bar{x}_2$.

When $\bar{x}_1 < \bar{x}_2$ (i.e., $\frac{m_1}{n_1} < \frac{m_2}{n_2}$), there are two conditions that must hold for it to be worthwhile for the teacher to teach. First, it must be the case that pulling Arm_1 could change the learner's action from Arm_2 to Arm_1; and second, it must be the case that the expected cost of teaching is less than the expected benefit of teaching. Specifically, we need the following to hold:

1. $\frac{m_1+1}{n_1+1} > \frac{m_2}{n_2}$
2. $p_* - p_1 < p_1(p_1 - p_2)$

The right hand side of the second inequality is the probability that Arm_1 will yield a 1 multiplied by the difference in expected values between Arm_1 and Arm_2.

Note that we can also explicitly calculate the expected values of both not teaching (EV_{nt}) and teaching (EV_t). $\text{EV}_{nt} = p_* + p_2$ and $\text{EV}_t = p_1 + p_1^2 + (1 - p_1)p_2$.

4.2 Algorithm and analysis

Building on the intuition from Section 4.1, this section sketches our fully-implemented polynomial memory and time dynamic programming algorithm for determining the teacher's optimal action with any number of rounds left. The full algorithm is available in the online appendix.[1] It takes as input initial values for m_1, n_1, m_2, n_2, and r, which we denote as M_1, N_1, M_2, N_2, and R respectively, and it outputs whether the teacher's expected value is higher if it teaches by pulling Arm_1 or if it exploits by pulling Arm_*.

The dynamic programming algorithm works backwards from smaller to bigger values of r, computing the expected value of the optimal action from any possible values of m_1, n_1, m_2, and n_2 that could be reached from the initial values.

First, consider the values that m_1, n_1, m_2, and n_2 can take on when there are r rounds left.

- Because both agents can pull Arm_1 any number of times, with r rounds left (after $R - r$ rounds have passed), n_1 can range from N_1 (if Arm_1 was never selected) to $N_1 + 2(R - r)$.
- Any number of the $n_1 - N_1$ times that Arm_1 was pulled, m_1 could have increased by 1. Thus m_1 can range from M_1 to $M_1 + (n_1 - N_1)$.
- Because only the learner pulls Arm_2, it will be pulled at most once per round. But the range of n_2 depends on the value n_1, because the learner only pulls Arm_2 when it does not pull Arm_1. Thus n_2 can range from $N_2 + \max(0, R - r - (n_1 - N_1))$ to $N_2 + (R - r) - \max(0, n_1 - N_1 - (R - r))$.
- Similarly to m_1, m_2 can range from M_2 to $M_2 + (n_2 - N_2)$.

The algorithm is structured as nested `for` loops using these ranges. For each reachable combination of values, the algorithm computes the teacher's optimal action (Arm_1 or Arm_*), and the expected long-term value of taking that action: the expected sum of payoffs for the optimal action and all future actions by both the teacher and the learner.

Both the memory and runtime complexities of this algorithm for computing the optimal teacher action with R rounds remaining for any starting values of the other variables are $O(R^5)$.

Although the algorithm runs iteratively, in principle we can convert the stored data structure into closed form computations of both teaching and not teaching. This conversion is based on the probabilities of the various possible outcomes of the pulls of the arms. However the closed form equations will be dependent upon m_1, n_1, m_2, and n_2.

4.3 Other Discrete Distributions

The algorithm and analysis to this point in this section all deal with the *binary* case in which each arm returns either 1 or 0 on each pull: 1 for a success and 0 for a failure. However, the algorithm and analysis extend trivially to distributions in which the success and failure payoffs from each arm differ from 1 and 0 and differ across the arms. The key property is that each arm has a success payoff that is realized with probability p_i and a (lower) failure payoff that is realized otherwise. Either or both of the payoffs can even be negative, representing an action penalty.

The results can also be generalized from binary distributions to any discrete distribution. In this case the algorithm includes extra nested `for` loops for each possible outcome of pulling an arm (not just two per arm). The exponent of the space and runtime complexities of the algorithm is increased accordingly, but the algorithm remains polynomial.

4.4 Numerical Results and Experiments

With the aid of the algorithm presented in Section 4.2, we tested several conjectures experimentally. In this section we consider the following questions:

1. Are there any patterns in the optimal action as a function of r when all other parameters are held constant?
2. How sensitive is the expected value computation to the relationship between $m_1, n_1, m_2, n_2, p_1, p_2$, and p_*?
3. When the algorithm is run, how many of the states tend to have Arm_1 (teaching) as the optimal action?

First, consider the effect of increasing the number of rounds remaining to be played, r. Intuitively, as r increases, the more time there is to benefit from teaching. However, there are even cases such that increasing r from 1 to 2 leads to a change in optimal action from teaching to not teaching. This situation arises when, with just one round remaining, there is a small enough cost to teaching that the teacher ought to try to get the learner to forgo Arm_2 even though the chances of succeeding are small; but with two rounds remaining, the learner's initial selection of Arm_2 will almost surely be sufficient for it to "teach itself" that it should select Arm_1 on the next round. This scenario is exemplified by the following parameters: $p_* = .076075, p_1 = .076, p_2 = .075, m_1 = 3020, n_1 = 40000, m_2 = 910, n_2 = 12052$.[2] In this case, both constraints from Section 4.1 are satisfied, thus the optimal action when $r = 1$ is Arm_1 (teach). However when $r = 2$, $\text{EV}_t = .302228 < \text{EV}_{nt} = .303075$: the optimal teacher action is Arm_*.

Second, note that the optimal action is very sensitive to the exact values of all the parameters. For example, when $p_* = .5, p_1 = .4, p_2 = .16, r = 4, m_2 = 2$, and $n_2 = 5$, the teacher's optimal action can differ even for identical values of $\bar{x}_1$. When $m_1 = 1$ and $n_1 = 3$, the optimal action is not to teach (Arm_*), but when $m_1 = 2$ and $n_1 = 6$, the optimal action is to teach (Arm_1) — even though $\bar{x}_1$ is $\frac{1}{3}$ in both cases. Similarly small changes in any of the other parameter values can change the teacher's optimal action.

Third, we consider how many of the states tend to have Arm_1 (teaching) as the optimal action when running the algorithm. For example, when $p_* = .5, p_1 = .4, p_2 = .16, m_1 = n_1 = m_2 = n_2 = 1$, solving for the optimal action with 15 rounds to go (r=15) leads to 81600 optimal actions computed (iterations through the `for` loops), 80300 of which are not to teach (Arm_*). In general, it seems that at least 90% of the optimal actions are Arm_*, even when the ultimate correct action is to teach, and usually significantly more than that. This observation perhaps suggests that in the Gaussian case below, when the optimal action cannot be solved for so easily, the default heuristic should be not to teach. We examine this hypothesis in Section 5.3

5. NORMAL DISTRIBUTION ARMS

In Section 4, we focused on arms with discrete payoff distributions. However in general ad hoc team settings, action payoffs may come from continuous distributions. In this section we turn to the case in which the distributions are Gaussian. Now, in addition to the expected value μ_i, which is the mean of the distribution, arms are characterized by a standard deviation, σ_i.

There are two main reasons that this case is more complicated than the discrete case. First, rather than a discrete set of possible future states, there are infinitely many possible outcomes from each pull. Second, in contrast to the constraints laid out in Section 4.1 for when it is worthwhile to teach, in the Gaussian case the μ's and the $\bar{x}$'s (which correspond to the p's and the m's and n's in the binary case) interact in the same inequality, rather than constituting independent constraints.

Both of these complications are readily illustrated even with $r = 1$. We thus begin by analyzing that case in Section 5.1. Recall that all the results from Section 3 still apply in this case. For example, it is only worth considering teaching when $\bar{x}_1 < \bar{x}_2$. We then consider the case when $r = 2$ in Section 5.2 and present some empirical data in Section 5.3. In contrast to the discrete case, we do not have an algorithm for exactly computing the optimal action when $r > 1$. In principle it can be estimated numerically, though with increasing inefficiency as r increases.

5.1 $\bar{x}_1 < \bar{x}_2, r = 1$

In order to analyze this case, we make use of the cumulative distribution function (CDF) of the normal distribution, denoted as $\Phi_{\mu,\sigma}(v)$. Exactly as in the binary case, with one round left, the teacher should teach when the expected cost of teaching, $\mu_* - \mu_1$, is less than the probability that teaching will successfully cause the learner to switch its choice from Arm_2 to Arm_1, $\Phi_{\mu_1,\sigma_1}(y)$, multiplied by the benefit of successful teaching, $\mu_1 - \mu_2$. Here y is the minimum return from Arm_1 that would cause the sample average of Arm_1 to surpass that of Arm_2: $\frac{m_1+y}{n_1+1} = \bar{x}_2$.

Therefore, the teacher should pull Arm_1 if and only if

$$1 - \Phi_{\mu_1,\sigma_1}(\bar{x}_2(n_1 + 1) - \bar{x}_1 n_1) > \frac{\mu_* - \mu_1}{\mu 1 - \mu_2} \quad (1)$$

(Recall that $\bar{x}_1 = \frac{m_1}{n_1}$ by definition). Otherwise, the teacher should pull Arm_*. We can then compute the expected value of the optimal action as:

- If $\bar{x}_1 > \bar{x}_2$, $\text{EV}_{nt} = \mu_* + \mu_1$
- Else, if the optimal action is to teach, $\text{EV}_t = \mu_1 + \mu_2 \Phi_{\mu_1,\sigma_1}(\bar{x}_2(n_1+1) - \bar{x}_1 n_1)) + \mu_1(1 - \Phi_{\mu_1,\sigma_1}(\bar{x}_2(n_1 + 1) - \bar{x}_1 n_1))$
- Else $\text{EV}_{nt} = \mu_* + \mu_2$.

Since there are readily available packages, for example in java, for computing $\Phi_{\mu_1,\sigma_1}(y)$, this result can be considered a closed form solution for finding the optimal teacher action and its expected value when $r = 1$.

5.2 $\bar{x}_1 < \bar{x}_2, r \geq 2$

In contrast, when $r > 1$, there is no such closed form method for finding the optimal action. Rather, integrals over functions need to be estimated numerically. As r increases, the inefficiency of this process compounds: for each sample, and at each round, it is necessary to estimate the values of both EV_{nt} and EV_t so that the optimal action from that point can be determined. In a sense, the value of a nested integral, with a total of r levels of depth, needs to be computed. Alternatively, the continuous distribution can be approximated with a discrete distribution and then solved as in Section 4. To date, we have not been able to characterize anything more formal or concrete about this case. Instead we discuss some conjectures and heuristics in the following section.

5.3 Numerical Results and Experiments

Even if we cannot practically determine in general what the teacher's optimal action is, it may be possible to find some reasonable heuristics. To this end, in this section we consider the following questions, the first of which is parallel to the first question considered in Section 4.4:

1. Are there any rules or patterns in the optimal action as a function of r?
2. How do various teacher heuristics compare to one another in performance?

[2] Note that this scenario is not particularly unlikely: $\frac{m_1}{n_1} \approx p_1, \frac{m_2}{n_2} \approx p_2$.

First, just as in the binary case, intuition suggests that increasing r should make it more beneficial to teach since there is more time for the added information to be used by the learner. However again, we can find a counterexample even with $r = 1$ and 2.

Consider the case in which $(\mu_*, \sigma_*) = (10, 0), (\mu_1, \sigma_1) = (9, 2)$, and $(\mu_2, \sigma_2) = (7, 2)$. Suppose that the learner has observed Arm_1 being pulled once when it got a payoff of 6.99 ($\bar{x}_1 = 6.99, n_1 = 1$), and it observed Arm_2 once for a payoff of 8 ($\bar{x}_2 = 8, n_2 = 1$).

With these values it is barely *not* worth it for the teacher to teach with $r = 1$. That is, with these values, Inequality 1 is not satisfied, but if $\bar{x}_1$ were 7.01, then it would be satisfied. Thus we know with certainty that the teacher's optimal action is Arm_*.

When $r = 2$, we can determine experimentally what the teacher's optimal action is by averaging the results of multiple trials when the teacher starts by teaching vs. not teaching and then acting optimally in the last round. In this case, when averaging over 2000 samples, the teacher reliably does better teaching (34.4 average return over the last 2 rounds) than when not teaching (34.2). Though the numbers are close and have high variance within a set of 2000 samples, the result is robust across multiple sets of 2000 samples.

When doing these experiments, we can gain a deeper understanding by considering the average situation after the teacher and learner have each taken one action, such that there is one more round remaining. First, consider the case in which the teacher does not teach with two rounds remaining. Thus it selects Arm_* and the learner selects Arm_2. Though the teacher's action has no impact on the relationship between $\bar{x}_1$ and $\bar{x}_2$ for the final round, the learner's action does. In one set of 2000 samples, the status after the first round was as follows:

- $\bar{x}_1 > \bar{x}_2$: 29.5%
- $\bar{x}_1 < \bar{x}_2$, Inequality 1 true (worth teaching): 39.2%
- $\bar{x}_1 < \bar{x}_2$, Inequality 1 false (not worth teaching): 31.4%

Weighting all three cases by their frequency, the total average expected value during the last round was 17.737.

On the other hand, when the teacher selects Arm_1 with two rounds remaining, we see the following breakdown after the first round:

- $\bar{x}_1 > \bar{x}_2$: 64.0%
- $\bar{x}_1 < \bar{x}_2$, Inequality 1 true (worth teaching): 14.1%
- $\bar{x}_1 < \bar{x}_2$, Inequality 1 false (not worth teaching): 22.0%

Again weighting the three cases by their frequency, the total average expected value during the last round was 18.322.

So in this case, after teaching in the second last round, the expected value of the last round is higher than when not teaching in the second last round. Most of this advantage comes because it is more likely that $\bar{x}_1 > \bar{x}_2$ prior to the final round. This advantage makes up for the slight cost of teaching in the initial round.

Though perhaps typical, it is not always the case that increasing r increases the benefit of teaching. Just as we found in the binary case in Section 4.4, in the Gaussian case it is also possible that increasing r from 1 to 2 and holding all other parameters constant could cause a switch from teaching being optimal to not teaching being optimal.

For example, consider the case in which $(\mu_*, \sigma_*) = (2.025, 0)$, $(\mu_1, \sigma_1) = (2, 1)$, and $(\mu_2, \sigma_2) = (1, .0001)$. Suppose that $\bar{x}_1 = 3, n_1 = 1$, and $\bar{x}_2 = 3.4, n_2 = 1$. Inequality 1 holds because the cost of teaching, $\mu_* - \mu_1 = .025$, is less than the potential benefit, $\mu_1 - \mu_2 = 1$, times the probability that teaching will succeed, $1 - \Phi_{\mu,\sigma}(.38) = .036$. Thus the optimal action when $r = 1$ is Arm_1.

However with two rounds remaining, the optimal action is Arm_*. Again considering sets of 2000 samples, the expected value of teaching is reliably 8.85 (4.025 of which comes from the last round), while that of not teaching is 8.70 (3.750 from the last round). Intuitively in this case, teaching is generally unlikely to help, and is also generally unnecessary: the learner will "teach itself" that Arm_1 is better than Arm_2 when it selects Arm_2 the first time. However with just one round remaining, it is worth it for the teacher to take a chance that teaching will help because even though the odds are low, so is the cost.[3]

Second, in addition to being of theoretical interest, the phenomenon that increasing r can cause teaching to be less worthwhile also has practical import, in particular in the context of considering possible heuristics for the teacher when $r > 1$. Specifically, we tested the following three heuristic teacher strategies under a variety of conditions:

1. Never teach;
2. Teach iff $\bar{x}_1 < \bar{x}_2$;
3. Teach iff it would be optimal to teach if $r = 1$ and all other parameters were unchanged.

Heuristic 3 would be particularly appealing were it the case that increasing r always made teaching more worthwhile. As it is, we found that none of these heuristics consistently outperforms the others.

Specifically, we compared the three heuristics under the six possible relationships of $\mu_1, \mu_2, \bar{x}_1$, and $\bar{x}_2$ subject to the constraint that $\bar{x}_1 < \bar{x}_2$ (e.g. $\bar{x}_1 < \bar{x}_2 < \mu_1 < \mu_2$, or $\mu_1 < \bar{x}_1 < \mu_2 < \bar{x}_2$). For each comparison, we sampled μ_1 and μ_2 uniformly at random from $[0, 10]$, setting the lower of the two draws to be μ_2; sampled σ_1 and σ_2 uniformly at random from $[0, 1]$; set $n_1 = n_2 = 1$; and drew m_1 and m_2 from their respective distributions until the required relationship between $\mu_1, \mu_2, \bar{x}_1$, and $\bar{x}_2$ was satisfied. Holding all of these values constant, we then tested all three heuristics for 9 different values of r ranging from 2 to 500.[4] Each test consisted of 10 trials, with the results being averaged. We then repeated the entire process with new draws of $\mu_1, \mu_2, \bar{x}_1$, and $\bar{x}_2$ five times for each of the six relationships.

An analysis of these results revealed that each heuristic outperforms the other two under some circumstances. Finding more sophisticated heuristic and/or principled teacher strategies that perform consistently well is one of the main open directions of future work in the context of this research.

6. MORE THAN THREE ARMS

To this point, we have assumed that the learner has only two arms available and the teacher has only one additional arm. In this section we generalize to the case in which there are more than three arms total.

Observe that adding additional arms that are only available to the teacher does not change anything. Only the best such arm (the one with the greatest expected value) should ever be considered by the teacher. We continue to call that arm Arm_*; the others can be ignored entirely.

Thus, we focus on the case in which there are additional arms available to both the teacher and the learner: Arm_1,

[3]Thanks to Daniel Stronger for this example.

[4]2,3,4,5,10,20,50,100, and 500.

Arm$_2$, ..., Arm$_z$ such that $\mu_1 > \mu_2 > \ldots > \mu_z$. In brief, the results we presented in Sections 3–5 all extend naturally to this more general case.We generalize the notation from Section 2 in the obvious ways.

- It can be beneficial for the teacher to pull Arm$_1$–Arm$_{z-1}$.
- The teacher should never pull Arm$_z$.
- Never teach with Arm$_i$ when $\bar{x}_i > \bar{x}_j, \forall j \neq i$.
- Do not teach when $n_1 = 0, n_2 = 0, \ldots$, and/or $n_z = 0$.
- There *are* situations in which the teacher should teach with Arm$_j$ even when $\exists i < j$ s.t. $\bar{x}_i > \bar{x}_j$. That is, pulling Arm$_2$ may be optimal, even when $\bar{x}_1 > \bar{x}_2$.

Together, these statements comprise everything that can be said theoretically about which arm to pull, independent of the distributions of the arms. The last result is perhaps somewhat surprising. It arises when $r \geq 2$ and $\bar{x}_j$ is an underestimate of the true value ($\bar{x}_j < \mu_j$) and $\exists k > j$ s.t. $\bar{x}_k$ is a large overestimate.[5] Then it can be better to ensure that Arm$_j$ is pulled as many times as possible, to minimize the chance that Arm$_k$ is ever pulled. We have found examples of such scenarios in both the discrete and Gaussian cases.

Additionally, the algorithm for the discrete case generalizes naturally. Expected values and optimal actions must now be calculated for all reachable values of m_1–m_z and n_1–n_z. Since the teacher could teach with any arm other than Arm$_z$, the ranges of the variables m_1–m_{z-1} and n_1–n_{z-1} match those of m_1 and n_1 in Section 4.2. The range of m_z matches that of m_2 in Section 4.2, and n_z is similar to n_2, except that the two occurrences of $n_1 - N_1$ (both inside "max" operators) need to be changed to $\sum_{i=1}^{z-1} n_i - N_i$.

Both the memory and runtime bounds of the extended algorithm generalize naturally to $O(R^{2z+1})$. The extended algorithm generalizes to arbitrary discrete distributions exactly as in Section 4.3.

Similarly, with normal distributions and one round left, the results from Section 5.1 generalize by considering the expected value of any Arm$_j$, $j < z, j \neq i$; and as in Section 5.2 we have no closed form solution if $r > 2$.

7. RELATED WORK

The broad context for this research is ad hoc teams in which teammates need to work together without any prior coordination. This perspective is at odds with most prior treatments of agent teamwork, which define explicit coordination protocols, languages, and/or shared assumptions about which the agents are mutually aware [8, 17].

The concept of ad hoc *human* teams has arisen recently in military and industrial settings, especially with the rise of outsourcing. There have also been autonomous agents developed to help support human ad hoc team formation [10, 12, 7]. This work relies on an analysis of the sources of team variability, including member characteristics, team characteristics, and task characteristics [12]. Our can-collecting example in Section 1 was inspired in part by an experimental bottle collecting task done by human ad hoc teams [7]. In addition, software agents have been used to support the *operation* of human teams [4], and for distributed information gathering from distinct, otherwise independent information sources [20]. But we are not aware of any work that enables an autonomous agent to itself act as an ad hoc teammate with previously unknown teammates.

The only prior work that we are aware of that takes a perspective similar to the ad hoc team perspective is that of Brafman and Tennenholtz [3] in which they consider a teacher agent and a learner agent repeatedly engaging in a joint activity. While the learner has no prior knowledge of this activity, the teacher understands its dynamics. As in our model, the teacher's goal is also to lead the learner to adopt a particular behavior. However the outcomes of actions in their setting are deterministic, and they mainly consider a situation in which teaching is not costly: the goal of their teacher is to maximize the number of times that the learner chooses the "right" action. Thus in some sense, the teacher is not "embedded" in the environment.

Also somewhat related is the recent work of Zhang et al. [21] on "environment design." Here, the controlling agent can alter aspects of the environment for a learning agent in an MDP so as to influence its behavior towards a particular goal. Once again, the controlling agent is not itself embedded in the environment and taking actions itself.

In the narrower context of our k-armed bandit instantiation of ad hoc teams, our research is characterized by cooperative agents with asymmetric information and asymmetric capabilities which are acting in an uncertain environment in which both agents are embedded in the environment (their actions affect the team's payoff) but the agents cannot communicate directly. To the best of our knowledge, no prior research meets all of the above characteristics. For example, research on multiagent reinforcement learning either does not consider the teacher as being embedded in the environment [9], considers the agents as having symmetric information and capabilities [5] and/or consider more competitive agents [16]. There are many other approaches for cooperative multiagent learning (see surveys at [14, 18]). But to the best of our knowledge, none covers our setting.

k-armed bandits have been extensively studied (see a survey at [1]), but also in this literature we are not familiar with any work that considers a teacher and a student with asymmetric capabilities and information who aim to maximize joint reward. Most of the works consider the case where each player tries to maximize its own reward (e.g., [2, 6]). Situations in which the agents do not have symmetric roles are studied in the context of the principal-agent problem [11]. This setting is much different than ours because of the conflicting utilities of the principal and the agent.

Multi-player multi-armed bandit problems have been also used to model the challenges facing users of collaborative decision-making systems such as reputation systems in e-commerce, collaborative filtering systems, and resource location systems for peer-to-peer networks. Here the main challenge is deciding which player to trust [13]. We assume that the learner sees the actual outcomes of the teacher and no issues of trust arise.

8. DISCUSSION AND CONCLUSION

This paper has presented a multiagent cooperative k-armed bandit in which the agents have different knowledge states and different action capabilities, as an example of ad hoc teamwork. We have studied in detail the task of a teacher that knows the payoff distributions of all of the arms as it interacts with a learner that does not know the distributions, and that can only pull a subset of the arms. The teacher's goal is to maximize the expected sum of payoffs as the two agents alternate actions. At any point, it can either exploit its best available action or increase the learner's knowledge

[5]The published version mistakenly wrote this as $\exists k < j$.

by demonstrating one of the learner's actions.

Within the specific scenario examined in this paper, we proved several theorems regarding situations in which we know which actions are or cannot be optimal for the teacher. We then narrowed our focus to two different types of probability distributions for the arms. For discrete distributions, we presented a polynomial memory and time algorithm for finding the teacher's optimal action. When the arms have Gaussian distributions, we can only find the optimal action efficiently when there is one round left. In both cases we augment the theoretical results with some experimental analyses using our fully-implemented algorithms.

This research opens up several exciting directions for future research. Aside from the points mentioned throughout the paper, many other generalizations are possible. For example, we have verified that at least some of our results can be extended to the discounted, infinite horizon case. One could consider arms with additional types of distributions, or types of distributions that differ among the arms (e.g. some discrete and some Gaussian). Additionally, we have considered a scenario in which the teacher knows the set of distributions of the arms, but not which arm has which distribution. Thus the teacher needs to itself explore, while still trading off between exploitation and teaching. Under restrictive assumptions, our dynamic programming algorithm from Section 4 extends to this case.

It is also important to note that although we assumed throughout the paper that the learner is completely greedy, our underlying problem remains interesting even when the learner uses a policy that yields some exploration. For example, suppose the learner follows an ϵ-greedy policy, i.e., it selects the arm with the highest observed sample average so far only with probability $(1-\epsilon)$ and with probability ϵ chooses an arm randomly. If we revisit the example in Section 3.1, it can still be beneficial for the teacher to pull Arm_1. Specifically, when $\epsilon = 0.01$ and everything else is the same as in the example, then it is still beneficial to teach: the expected values of teaching and not teaching are still close to 16 and 15 respectively.

In the broader context, this research is just one step towards the long-term goal of creating a fully capable ad hoc team player. In order to achieve this goal, many more studies of this magnitude will be needed that consider situations in which, for example, there are more than two teammates, the teammates can communicate directly, the teammates' behaviors are not fully known, or some teammates have *more* knowledge and/or capabilities than our agent. We intend to follow up on these challenges in our future research and hope that this research will inspire others to also work towards the eventual creation of fully general ad hoc team players.

Acknowledgements

Thanks to Yonatan Aumann, Vincent Conitzer, Reshef Meir, Daniel Stronger and members of the UT Austin Learning Agents Research Group (LARG) for helpful comments and suggestions. The research is supported in part by grants from NSF (IIS-0917122, 0705587), ONR (N00014-09-1-0658), DARPA (FA8650-08-C-7812), U.S. Army Lab (W911NF-08-1-0144), ISF (#1357/07), the FHWA (DTFH61-07-H-00030), and the Fulbright and Guggenheim Foundations.

9. REFERENCES

[1] D. Bergemann and J. Valimaki. Bandit problems. Technical report, Cowles Foundation Discussion Paper, 2006.

[2] P. Bolton and C. Harris. Strategic experimentation. *Econometrica*, 67:349–374, 1999.

[3] R. I. Brafman and M. Tennenholtz. On partially controlled multi-agent systems. *JAIR*, 4:477–507, 1996.

[4] H. Chalupsky, Y. Gil, C. Knoblock, K. Lerman, J. Oh, D. Pynadath, T. Russ, and M. Tambe. Electric elves: Applying agent technology to support human organizations. In *IAAI*, 2001.

[5] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *AAAI*, pages 746–752, 1998.

[6] M. Cripps, G. Keller, and S. Rady. Strategic experimentation with exponential bandits. *ECONOMETRICA*, 73:39–68, 2005.

[7] J. A. Giampapa, K. Sycara, and G. Sukthankar. Toward identifying process models in ad hoc and distributed teams. In K. V. Hindriks and W.-P. Brinkman, editors, *HuCom*, pages 55–62, December 2008.

[8] B. J. Grosz and S. Kraus. Collaborative plans for complex group actions. *AIJ*, 86:269–358, 1996.

[9] L. ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *MLJ*, 8(3/4):293–321, 1992.

[10] J. Just, M. Cornwell, and M. Huhns. Agents for establishing ad hoc cross-organizational teams. In *International Conference on Intelligent Agent Technology*, pages 526–30, September 2004.

[11] A. Kayay. When does it pay to get informed? *International Economic Review*, 2009. forthcoming.

[12] R. Kildare. Ad-hoc online teams as complex systems: agents that cater for team interaction rules. In *Proceedings of the 7th Asia-Pacific Conference on Complex Systems*, December 2004.

[13] R. D. Kleinberg. *Online Decision Problems*. PhD thesis, Department of Mathematics, MIT 2005.

[14] L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *JAAMAS*, 11:387–434, 2005.

[15] H. Robbins. Some aspects of the sequential design of experiments. *Bulletin American Mathematical Society*, 55:527–535, 1952.

[16] A. Schaerf, Y. Shoham, and M. Tennenholtz. Adaptive load balancing: A study in multi-agent learning. *JAIR*, 2:475–500, 1995.

[17] P. Stone and M. Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *AIJ*, 110(2):241–273, June 1999.

[18] P. Stone and M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, July 2000.

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

[20] K. Sycara, K. Decker, A. Pannu, M. Williamson, and D. Zeng. Distributed intelligent agents. *IEEE Expert*, 11(6), December 1996.

[21] H. Zhang, Y. Chen, and D. Parkes. A general approach to environment design with one agent. In *IJCAI*, 2009.