To appear in *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence* (AAAI), Bellevue, Washington, USA, July 2013.

Teamwork with Limited Knowledge of Teammates

Samuel Barrett and **Peter Stone**
Dept. of Computer Science
The Univ. of Texas at Austin
Austin, TX 78712 USA
{sbarrett,pstone}@cs.utexas.edu

Sarit Kraus[1,2]
[1]Dept. of Computer Science
Bar-Ilan University
Ramat Gan, 5290002 Israel
[2]Inst. for Adv. Computer Studies
University of Maryland
College Park MD 20742
sarit@cs.biu.ac.il

Avi Rosenfeld
Dept. of Industrial Engineering
Jerusalem College of Technology
Jerusalem, 9116001 Israel
rosenfa@jct.ac.il

Abstract

While great strides have been made in multiagent teamwork, existing approaches typically assume extensive information exists about teammates and how to coordinate actions. This paper addresses how robust teamwork can still be created even if limited or no information exists about a specific group of teammates, as in the ad hoc teamwork scenario. The main contribution of this paper is the first empirical evaluation of an agent cooperating with teammates not created by the authors, where the agent is not provided expert knowledge of its teammates. For this purpose, we develop a general-purpose teammate modeling method and test the resulting ad hoc team agent's ability to collaborate with more than 40 unknown teams of agents to accomplish a benchmark task. These agents were designed by *people other than the authors* without these designers planning for the ad hoc teamwork setting. A secondary contribution of the paper is a new transfer learning algorithm, TwoStageTransfer, that can improve results when the ad hoc team agent does have *some* limited observations of its current teammates.

1 Introduction

Creating effective teamwork is central to many tasks. Due to the importance of this problem, a variety of teamwork frameworks and formalizations have been proposed by the multiagent research community (Grosz and Kraus 1996; Tambe 1997; Horling et al. 1999). In implementing teamwork, one of these existing approaches may be used, but they all require agreeing in advance upon a shared coordination protocol. However, many real-world domains exist where agents may be developed by a variety of sources, making it difficult to ensure that all the agents share the same protocols. Therefore, it is important that agents be capable of adapting to previously unseen teammates to cooperate in accomplishing their joint tasks. Researchers have begun studying this problem under the name *ad hoc teamwork* (Stone et al. 2010), but most previous work assumes that the ad hoc team agent knows its teammates' behaviors or has expert-provided knowledge.

Consider a scenario in which several robots are tasked with capturing intruders. These robots are designed to coordinate as a team to capture the intruder as quickly as possible. However, if a robot breaks, it may be necessary to immediately replace the robot with a new robot that may not know the team's strategy for capturing intruders. Therefore, it is advantageous for the replacement robot to observe its teammates and quickly adapt to them in order for the team to capture intruders in minimal time. If the replacement robot has observations of past teammates, it can use this knowledge to adapt to the new teammates more quickly. To address this problem, this paper uses a simple, but representative version of this team domain with teammates not developed by the authors. In this problem, we assume that all teammates have experience in the domain, but the replacement agent may not necessarily have experience with its *current* teammates.

The main contribution of this paper is the introduction and evaluation of an ad hoc team agent that explicitly builds a library of models of past teammates and selects actions using a sample-based planning algorithm. The ad hoc team agent is evaluated by its ability to cooperate with more than 40 previously unobserved teams of agents not created by the authors, denoted *externally-created* teams. The externally-created teammates were designed to perform the shared task, but not adapt to new teammates. The use of externally-created teammates prevents biasing the teammates to be prepared for ad hoc teamwork and prevents inadvertent expert knowledge being given to the ad hoc agent. The second contribution is a new transfer learning algorithm, TwoStage-Transfer, that can improve the ad hoc team agent's performance when it has some limited observations of its current teammates.

2 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 (Grosz and Kraus 1999). In STEAM (Tambe 1997), team members build a partial hierarchy of joint actions. The TAEMS framework (Horling et al. 1999) consists of a hierarchy of rules, where agents coordinate through common groups, tasks, and methods.

While these algorithms are effective in many settings, they all assume that all teammates are using the same teamwork mechanism. Even when this assumption is true, in many classes of teamwork problems, including the problems we study, selecting optimal actions is computationally intractable (Pynadath and Tambe 2002). Thus, even if a joint coordination algorithm could be assumed, designing agents to effectively learn how to behave using "standard" approaches such as Bayesian learning or planners for partially observable Markov Decision Processes (POMDPs) are not realistic. In order to effectively learn this task and avoid this inherent complexity, we use generalized modeling and transfer learning combined with a sample-based planner.

Additionally, we consider a challenging yet practical set of problems where teammates cannot be assumed to have a common teamwork algorithm, referred to as ad hoc teamwork. This paper is particularly motivated by Barrett et al.'s work (2011), in which the ad hoc agents are provided expert knowledge about potential teammates. An overview to current research on ad hoc teamwork is presented by Barrett and Stone (2012). One line of early research on ad hoc teams involves an agent teaching a novice agent while performing a repeated joint task (Brafman and Tennenholtz 1996). Other research includes Jones et al.'s (2006) research on pickup teams cooperating to accomplish a treasure hunt. Liemhetcharat and Veloso (2011) reason about selecting agents for ad hoc teams, and in robot soccer, Bowling and McCracken (2005) consider the case where the ad hoc agent is given a different playbook from its teammates. Further work into ad hoc teams using stage games and biased adaptive play was performed by Wu et al. (2011). However, the majority of these works use author-created teammates or provide the agent with expert knowledge of its teammates.

Whereas ad hoc teamwork focuses on cooperating with teammates and makes assumptions about their intentions, the related problem of opponent modeling considers reasoning about opponents and often focuses on the worst case scenarios. One interesting approach is the AWESOME algorithm (Conitzer and Sandholm 2007) which achieves convergence and rationality in repeated games. Another approach is to explicitly model and reason about other agents' beliefs such as the work on I-POMDPs (Gmytrasiewicz and Doshi 2005), I-DIDs (Doshi and Zeng 2009), and NIDs (Gal and Pfeffer 2008). However, modeling other agents' beliefs greatly expands the planning space, and these approaches do not currently scale to larger problems. On the other hand, our research demonstrates that general teamwork modeling can be applied to learn how to behave in teams, even without **any** a-priori knowledge of the current teammates, though our approach does not make formal performance guarantees.

3 Problem Description

This paper considers the case where an ad hoc agent is trying to cooperate with a set of teammates it has never seen before. If the ad hoc agent and its teammates share methods for communication or coordination, it can directly cooperate with them. However, if these protocols are not available, the ad hoc agent should observe its teammates and try to adapt to their behaviors. If the ad hoc agent has previously observed agents similar to its current teammates, it should try to leverage its prior experiences in order to cooperate with them more effectively. As motivated in the introduction, an example domain is robots capturing intruders, for which we adopt a simpler version of the problem taken from the multiagent systems literature that still retains the interesting problem of requiring coordinated teamwork.

3.1 Pursuit Domain

The pursuit domain is a popular problem in the multiagent systems literature because it requires all of the teammates to cooperate to capture the prey (Stone and Veloso 2000). The details vary, but the pursuit domain revolves around a set of agents called *predators* trying to capture an agent called the *prey* in minimal time.

In our version of the pursuit domain, the world is a rectangular, toroidal grid of size 20x20, where moving off one side of the grid brings the agent back on the opposite side. Four predators attempt to capture the randomly moving prey by surrounding it on all sides in as few time steps as possible. At each time step, each agent can select to move in any of the four cardinal directions or to remain in its current position. All agents pick their actions simultaneously with collisions being decided randomly. In addition, each agent is able to observe the positions of all other agents.

3.2 Ad Hoc Teamwork

Ad hoc team agents must be able to cooperate with a variety of previously unseen teammates to accomplish a task. In this paper, we adopt the setting where a team of n agents are pre-designed to cooperate to accomplish a shared task. Then, one of these agents is replaced by an ad hoc team agent that shares the team's goals, but does not know its teammates' behaviors. Therefore, the ad hoc agent's goal is to improve the team's performance given its current teammates, but the ad hoc agent cannot directly control its teammates' behaviors, only its own actions. In this work, the shared task is the pursuit domain, and the teams have 4 agents that are externally created as discussed in depth in Section 5.1. This problem fits into the evaluation framework proposed by Stone et al. (2010).

4 Modeling Teammates

In order to cooperate effectively with its teammates, the ad hoc agent chooses its actions by planning about their long term effects. To this end, this paper assumes that the ad hoc agent knows the model of the domain and the prey but must learn to predict its teammates' actions. Learning teammate models is a departure from previous work where teammate models were fixed (Tambe 1997) or a set of possible models was provided by an expert (Barrett, Stone, and Kraus 2011).

4.1 Planning

Even if the ad hoc agent has a perfect model of its teammates, the planning problem is still difficult. With four predators and a single prey, the pursuit domain has a branching factor of $5^5 = 3,125$ actions and there are approximately $(20 * 20)^5 \approx 10^{13}$ different states. When the ad hoc agent is uncertain of its teammates' behaviors, the problem is partially observable, but the problem is too large for standard, optimal POMDP planning algorithms. To plan efficiently, our ad hoc agent uses Upper Confidence bounds for Trees (UCT) (Kocsis and Szepesvari 2006), a Monte Carlo Tree Search (MCTS) algorithm that employs upper confidence bounds for controlling the tradeoff between exploration and exploitation. Previous work has shown that UCT performs well on domains with high branching factors, such as Go (Gelly and Wang 2006) and large POMDPs (Silver and Veness 2010) as well as pursuit problems (Barrett, Stone, and Kraus 2011).

In UCT, the ad hoc agent plans more each time it must select an action. To plan, the ad hoc agent performs a number of simulations from the current world state until the prey is captured. The ad hoc agent tracks the number of times it has seen each state-action and estimates the value of the state-action, i.e. the expected time to capture the prey from the given state. During these simulations, the ad hoc agent selects its actions by choosing the one with the highest upper confidence bound, causing it to explore when it is unsure of the best action and exploit its knowledge when it is confident of the results.

4.2 Model Selection

Performing the simulations for the UCT rollouts requires that the ad hoc agent has a model its teammates' behavior. If the agent starts with a correct prior belief distribution over a set of possible behavior models, it can update the model probabilities using Bayes' theorem. However, if the correct model is not in the set, using Bayes' theorem may drop the posterior probability of good models to 0 for a single wrong prediction, unduly punishing good models that make a single mistake. Therefore, it may be advantageous to update the probabilities more conservatively, such as using the polynomial weights algorithm that has shown promise in regret minimization (Blum and Mansour 2007):

$$\begin{aligned} \text{loss} &= 1 - P(\text{actions}|\text{model}) \\ P(\text{model}|\text{actions}) &\propto (1 - \eta * \text{loss}) * P(\text{model}) \end{aligned}$$

where η is empirically chosen to be 0.1.

Given the current belief distribution over the models, the ad hoc agent can sample teammate models for planning, choosing one model for each rollout similar to the approach adopted by Silver and Veness (2010). Sampling the model once per rollout is desirable compared to sampling a model at each time step because resampling each step can lead to states in a rollout that no model predicts should be reached.

4.3 Learning Models

The previous sections described how the ad hoc agent can select the correct model and use it for planning, but they did not specify the source of these models. Past work has typically assumed that the ad hoc agent has expert-provided models, but a more general solution is for the ad hoc agent to learn the models. Learning allows the agent to gain a good set of diverse models over its lifespan, allowing better performance with arbitrary new teammates. The ad hoc agent builds models of past teammates' behaviors offline and then selects from these learned models online while cooperating with new teammates. It is expected that the past teammates are representative of the distribution of future teammates, though the future teammates have not yet been seen.

We treat building teammate models as a supervised learning problem, where the goal is to predict the teammates' actions using the features in Table 1 with all positions being relative to the modeled teammate. The model predicts the next action of each teammate; when combined with a model of the domain, the ad hoc agent can plan far into the future. With its observations of past teammates, the ad hoc agent learns a decision tree, implemented in the Weka toolbox (Hall et al. 2009). Several other classifiers were tried including SVMs, naive Bayes, decision lists, and nearest neighbor approaches as well as boosted versions of these classifiers. However, decision trees outperformed these methods in initial tests in a combination of prediction accuracy and training time. All model learning is performed offline, reflecting past experience in the domain, but the ad hoc agent updates its belief over the models online.

Description	Num. Features	Values
Predator Number	1	$\{0, 1, 2, 3\}$
Prey x position	1	$\{-10, \ldots, 10\}$
Prey y position	1	$\{-10, \ldots, 10\}$
Predator$_i$ x position	3	$\{-10, \ldots, 10\}$
Predator$_i$ y position	3	$\{-10, \ldots, 10\}$
Neighboring prey	1	{true,false}
Cell neighboring prey is occupied	4	{true,false}
Previous two actions	2	$\{\leftarrow, \rightarrow, \uparrow, \downarrow, \bullet\}$

Table 1: Features for predicting a teammate's actions. Positions are relative to the teammate.

To capture the notion that the ad hoc agent is expected to have extensive prior general domain expertise (as is assumed in the ad hoc teamwork setting), though not with the specific teammates at hand, we pre-train the ad hoc agent with observations of a pool of past teammates. Specifically, it watches teams of four predators for 50,000 steps for each past teammate type, and builds a separate model for each type of teammate. Preliminary tests show that less data can still be effective, but the focus of this research is about minimizing observations of the current teammates, not the previous ones. We treat the observations of previous teammates as experience prior to deploying the ad hoc agent. If some observations of the current teammates are available, we can improve our results using transfer learning in the form of TwoStageTransfer as discussed in Section 6.

5 Results with Unknown Teammates

This section evaluates a number of ad hoc agents that vary in the amount of information they have about their teammates. If the ad hoc agent has access to the true model, it can plan to cooperate with its teammates optimally or use the same behavior as the missing teammate. However, in the fully general ad hoc teamwork scenario, such a model is not available. Thus, this paper instead focuses on the case in which the ad hoc agent learns models of its past teammates. This section evaluates ad hoc agents that learn models of past teammates to several baselines.

5.1 Evaluation

To properly evaluate an ad hoc team agent, it is important to test it with a variety of possible externally-created teammates. Therefore, we collected two sets of teammates created by undergraduate and graduate computer science students in two offerings of a workshop taught by co-author Kraus at Bar-Ilan University. Importantly, these agents were created with no discussion of ad hoc teams; instead, the students were asked to create a team of predators that captured the prey as quickly as possible. The agents produced varied wildly in their approaches as well as their effectiveness and are broken into two sets based on the workshop for which they were produced. The first set of agents, $\text{Student}_{\text{Broad}}$, has 29 agents unfiltered for performance. The second set of agents, denoted $\text{Student}_{\text{Selected}}$, has 12 well performing student-created agents selected in Barrett et al. (2011). These agents are selected for being capable of capturing the prey in a 5x5 world in less than 15 steps on average, where the selection was used to eliminate agents that were too ineffective for any ad hoc agent to help. $\text{Student}_{\text{Broad}}$ contains a wider range of performance than $\text{Student}_{\text{Selected}}$ as it is filtered less heavily. These two sets provide a total of 41 different teammate behaviors for testing the ad hoc agent. Throughout this paper, we refer to a teammate type as its behavior function, meaning that agents coming from different students have different types.

For these evaluations, results are averaged over 1,000 episodes. In each episode, a random team is selected, a single agent is replaced by the ad hoc agent, and the team is given 500 steps in which to capture the prey as many times as possible. When the prey is captured, it is teleported to a random unoccupied location. The randomness is fixed across evaluations of different ad hoc agents to permit paired statistical analysis, and all statistical tests are performed as paired Student-T tests with $p = 0.01$. Results are normalized compared to the maximum number of prey captured if the ad hoc agent was planning with the true teammate models.

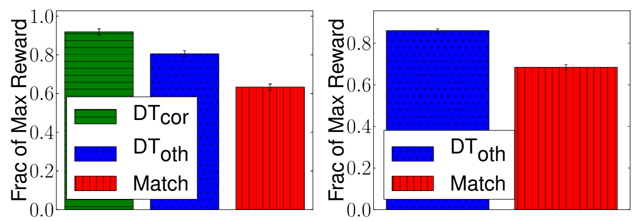
The behaviors tested are listed below, varying the information the ad hoc agent is given. All models are learned on past teammates in $\text{Student}_{\text{Broad}}$ using the methods in Section 4.3, where one model was learned per past teammate type. The behaviors are ordered from least to most general ad hoc team settings:

- **Match**: Match teammates' behavior. The ad hoc agent knows the true model of its teammates, and behaves like the agent it is replacing.
- **DT_{cor}**: Plan with the correct decision tree, the best the ad hoc agent can do with the learned models.
- **DT_{oth}**: Plan with the other decision trees. The ad hoc agent plans with its models learned from past teammates, **not** including the current type.

The ad hoc agent builds a separate decision tree for each type of teammate it has observed. The Match behavior represents the performance of the original, unified team of externally-created agents. The DT_{cor} behavior is the upper bound of how the ad hoc agent can perform if it knows its teammates' type and has previously encountered this type of teammate. The DT_{oth} behavior is representative of the most general ad hoc team scenario, where the ad hoc agent must cooperate with teammates that it has not previously observed.

5.2 Results

This section compares the performance of ad hoc agents that have learned models of their teammates to agents that have access to true models of their teammates. In the DT_{oth} setting, the ad hoc agent has only observed the teammates in $\text{Student}_{\text{Broad}}$ excluding the teammate it plays in evaluation, resulting in 28 past teammates in Figure 1a and 29 past teammates in Figure 1b. The DT_{cor} results are unattainable in the general ad hoc team setting because they require knowing the type of the current teammates prior to cooperating with them, and Match additionally requires the true model of the teammates.



(a) $\text{Student}_{\text{Broad}}$ (b) $\text{Student}_{\text{Selected}}$

Figure 1: Using varied amounts of prior knowledge to cooperate with agents.

Figure 1a shows the performance of these methods. As the teammates are created by students for a class project, the teammates may be arbitrarily far from optimal, causing Match to perform fairly poorly. Planning using UCT with the model learned from the current teammate type (DT_{cor}) performs well, but is unreachable as it requires that the ad hoc agent has encountered this type of teammate before and knows the type of these teammates. If the ad hoc agent has not encountered this type of teammate before, the DT_{oth} setting still performs very well, significantly outperforming the originally designed team shown by the Match baseline. The performance in the DT_{oth} setting suggests that the ad hoc agent can cooperate very effectively with teammates it has never seen before.

In case the agents created by one class are biased to be similar, we also evaluate the ad hoc agent on cooperating with the $\text{Student}_{\text{Selected}}$ agents. In the DT_{oth} setting, the ad hoc agent is given the 29 models of the previous teammates from $\text{Student}_{\text{Broad}}$. Figure 1b shows the results for cooperating with teammates drawn from $\text{Student}_{\text{Selected}}$. Despite the fact that the teammates drawn from $\text{Student}_{\text{Selected}}$ show better performance than the teammates from $\text{Student}_{\text{Broad}}$, the Match setting still performs the worst. Once again, planning with incomplete models still performs effectively, with DT_{oth} significantly outperforming Match. These results confirm those for the $\text{Student}_{\text{Broad}}$ tests, and the rest of the paper focuses on the $\text{Student}_{\text{Broad}}$ agents.

6 Generalizing Prior Experiences

Section 5 discusses how an ad hoc agent should cooperate with teammates it has interacted with before as well as how the agent should cooperate with completely new teammates. However, in many cases, an ad hoc agent may have a limited amount of time to observe its current teammates before it interacts with them. In addition, it has extensive observations from past interactions with other teammates. For example, in pickup soccer, this scenario corresponds to having past experience in pickup soccer, showing up to a new game, and watching a couple minutes before joining in. This scenario fits the *transfer learning* (TL) paradigm, but requires the ability to leverage multiple sources of related data. In this section, we introduce a new transfer learning algorithm to leverage such information to speed up learning about new teammates.

6.1 Transfer Learning Background

In this section, we discuss three state of the art transfer learning algorithms. In TL, the goal is to reuse information learned on a *source* data set to improve results on a *target* data set. For TL, only the performance on the target data matters; the source data is only used for training. Following this terminology, we consider the current teammates to be the *target* set, and the previously observed teammates are the *source* set.

TrAdaBoost (Dai et al. 2007) is a boosting-based algorithm, in which the source and target data sets are lumped together and then a model is learned via boosting. TwoStageTrAdaBoost (Pardoe and Stone 2010) was designed in response to problems of TrAdaBoost overfitting the training data. Therefore, TwoStageTrAdaBoost first searches over a set of possible weightings of the source data points, and determines which weighting is best using cross-validation. While the other transfer learning algorithms described here focus on using boosting, bagging approaches have also shown promise, specifically in the form of TrBagg (Kamishima, Hamasaki, and Akaho 2009). The TrBagg algorithm uses bootstrap sampling to create a number of data sets taken from the combined source and target data sets. Then, a model is learned on each data set, and these models then undergo a filtering phase, using cross validation to determine which models are most helpful.

6.2 TwoStageTransfer Algorithm

While the transfer learning algorithms discussed in Section 6.1 are effective on some problems, they do not directly address the problem of transferring knowledge from multiple sources. In general, they lump all source data into a single data set and expect the learning algorithms to handle this data. TwoStageTransfer is inspired by the TwoStageTrAdaBoost algorithm (Pardoe and Stone 2010), and it is designed to explicitly leverage multiple source data sets. Specifically in this domain, the ad hoc agent has observed many other agents, some of which are more similar to the target teammate than others. Therefore, tracking the source of the data may be important as it allows the ad hoc agent to discount data from agents that are differ greatly from it. Recent research into transfer learning has shown this information may improve results. Yao and Doretto (2010) had success using a boosting approach that allows each data source to propose a weak classifier at each iteration, but do not combine the data sets to learn a single classifier. In (2012), Huang et al. propose the SharedBoost algorithm to select the best features for prediction from a small number of source data sets for text classification.

TwoStageTransfer's goal is to find the best possible weighting for *each* set of source data and create a classifier using these weights, as described in Algorithm 1. TwoStageTransfer takes in the target data set T, the set of source data sets $\mathbb{S} = \{S_1, \ldots, S_n\}$, a number of boosting iterations m, a number of folds k for cross validation, and a maximum number of source data sets to include b. We use the annotation S^w to mean the data set S taken with weight w spread over the instances. The base model learner used in this case is a decision tree learner that handles weighted instances.

Ideally, TwoStageTransfer would try every combination of weightings, but having n source data sets and m different weightings leads to m^n possible combinations (in our case 10^{28}). Rather than try all of them, TwoStageTransfer first evaluates each data source independently, and calculates the ideal weight of that data source using cross validation. Then, it adds the data sources in decreasing order of the calculated weights. As it adds each data set, it finds the optimal weighting of that set with the data that has already been added. Finally, it adds the data with the optimal weight and repeats the procedure with the next data set. This algorithm requires only $nm + nm = 2nm$ combinations to be evaluated (in our case 560), with nm for the initial evaluations and then m when adding each n data sets. To achieve this efficiency, this approach does not guarantee optimality.

TwoStageTransfer is a general transfer learning algorithm that can be used in a variety of settings for learning from multiple source domains. For example, when classifying the subject of text documents, you may have labeled data from a variety of sources including newspapers, personal letters, and books. When trying to build a classifier for a new magazine, it is useful to transfer information about these other sources, bearing in mind that some sources such as newspapers may be more similar to the magazine than letters.

Algorithm 1 Transfer learning with multiple sources

```
TwoStageTransfer (T, 𝕊, m, k, b)
  for all S_i in 𝕊: do
    w_i ← CalculateOptimalWeight(T, ∅, S_i, m, k)
  Sort 𝕊 in decreasing order of w_i's
  F ← ∅
  for i from 1 to b do
    w ← CalculateOptimalWeight(T, F, S_i, m, k)
    F ← F ∪ S_i^w
  Train classifier c on T ∪ F
  return c

CalculateOptimalWeight(T, F, S, m, k):
  for i from 1 to m do
    w_i = |T|/(|T|+|S|) (1 − i/(m−1))
    Calculate err_i from k-fold cross validation on T using F
    and S^{w_i} as additional training data
  return w_j such that j = argmax_i(err_i)
```

6.3 Results

While the tests in Section 5 do not focus on the speed of learning, speed is crucial when learning about the current teammates. As in Section 5, the ad hoc agent has previously observed 50,000 training steps of each of the past 28 teammate types and this data is considered prior background knowledge. In addition, it has seen only 100 training steps of the current type of the teammate. Note that this is significantly less than the testing time of 500 steps, but the ad hoc agent is not learning online other than adapting its belief distribution over the possible models. Both the past and current teammates in this test are taken from $\text{Student}_{\text{Broad}}$.

All of the transfer learning algorithms use decision trees as their base learning algorithm. Each algorithm has some set of parameters that can be tuned, and their values were chosen in some preliminary tests and limited by computational power. For TwoStageTransfer and TwoStageTrAdaBoost, 10 different weightings were used. In TrAdaBoost and TwoStageTrAdaBoost, 10 boosting iterations were used. For TrBagg, a total of 1,000 sets were used for training classifiers, and a Naive Bayes classifier served as the fallback model. As in Section 5, all statistical tests are performed as paired Student-T tests with $p = 0.01$.

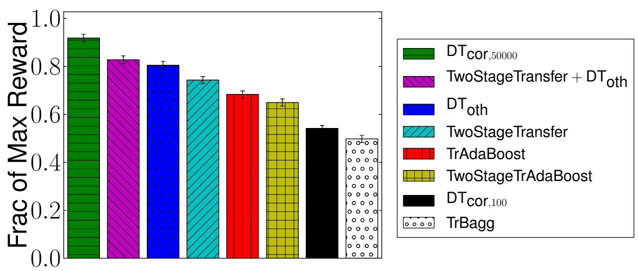


Figure 2: Comparing transfer learning algorithms' abilities to improve ad hoc agents that have limited observations of their current teammates.

Figure 2 shows the results of the four transfer learning algorithms, where the ad hoc agent plans with UCT using one of the learned models. All learning is done offline with only model selection happening online during the evaluation. One baseline for comparison is ignoring the previously observed teammates and learning a new model from just the observed 100 steps of the current teammates, shown as $\text{DT}_{\text{cor},100}$. As an upper baseline, we compare to the unattainable performance of using a model learned from 50,000 steps of the current teammate, shown as $\text{DT}_{\text{cor},50000}$, which represents the best performance attainable using models learned given large amounts of data.

In these results, TwoStageTransfer outperforms the other transfer learning algorithms, and the difference is statistically significant with $p < 0.01$. In addition, combining the models learned with TwoStageTransfer with the models learned from representative teammates in the TwoStageTransfer + DT_{oth} setting helps, reaching results that are statistically significantly better than DT_{oth} with $p < 0.01$. TrBagg performed poorly in this setting, mis-transferring information, possibly due to the fallback model used, though several were tested.

In addition, it is important to see how well TwoStageTransfer scales with different amounts of target data. Figure 3 shows results with varying amounts of target data, but constant amounts of source data. The difference between the results with 1,000 steps of target data and 100 is statistically significant, but the differences between 10,000 and 1,000 or 100 and 10 are not. The results show that the performance of TwoStageTransfer does improve with more target data, but the improvement is not smooth.

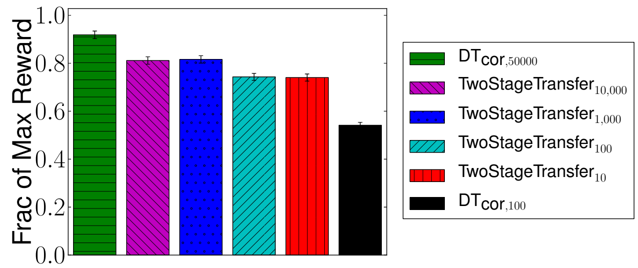


Figure 3: Various amounts of target data for creating models with TwoStageTransfer.

7 Conclusion

Existing research focuses on cases where teamwork frameworks are set in advance or an agent has access to the correct model of its teammates. Therefore, the core contribution of this paper is the introduction of an ad hoc agent that autonomously learns a library of models of past teammates and the evaluation this agent when cooperating with over 40 externally-created teammates that it has not previously encountered. Another contribution is the development of a new transfer learning algorithm, TwoStageTransfer, that can significantly improve results when the ad hoc agent has a limited number of observations of its teammates and observations of several previous teammates. Both learning a library of models and using transfer learning to improve models are expected to be applicable to other domains in which an agent must adapt to new teammates.

While this paper answers several questions about ad hoc team agents, it also raises new questions to be explored in future research. Future research should test whether similar algorithms apply to other ad hoc team domains. One possible research avenue is into what to communicate to teammates when communication protocols are available. In addition, this work focuses on agents that follow mostly fixed behaviors, with little adaptation to the ad hoc agent's behaviors; handling adaptive teammates is a complicated, but exciting, area for future research.

Acknowledgments

A portion of this research took place in the Learning Agents Research Group (LARG) at The University of Texas at Austin. LARG research is supported in part by grants from NSF (IIS-0917122), ONR (N00014-09-1-0658), and the FHWA (DTFH61-07-H-00030). This work was supported in part by ERC grant #267523, the Google Inter-university center for Electronic Markets and Auctions, and MURI grant number W911NF-08-1-0144

References

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

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

Blum, A., and Mansour, Y. 2007. Learning, regret minimization, and equilibria.

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

Brafman, R. I., and Tennenholtz, M. 1996. On partially controlled multi-agent systems. *Journal of Artificial Intelligence Research* 4:477–507.

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

Dai, W.; Yang, Q.; Xue, G.-R.; and Yu, Y. 2007. Boosting for transfer learning. In *ICML '07*, 193–200.

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

Gal, Y., and Pfeffer, A. 2008. Network of influence diagrams: Reasoning about agents' beliefs and decision-making processes. *Journal of Artificial Intelligence Research* 33:109–147.

Gelly, S., and Wang, Y. 2006. Exploration exploitation in Go: UCT for Monte-Carlo Go. In *NIPS '06*.

Gmytrasiewicz, P. J., and Doshi, P. 2005. A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research* 24(1):49–79.

Grosz, B. J., and Kraus, S. 1996. Collaborative plans for complex group action. *Artificial Intelligence* 86(2):269–357.

Grosz, B., and Kraus, S. 1999. The evolution of shared-plans. In Rao, A., and Woolridge, M., eds., *Foundations and Theories of Rational Agency*, 227–262.

Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; and Witten, I. H. 2009. The WEKA data mining software: an update. *SIGKDD Explor. Newsl.* 11:10–18.

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

Huang, P.; Wang, G.; and Qin, S. 2012. Boosting for transfer learning from multiple data sources. *Pattern Recognition Letters* 33(5):568 – 579.

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

Kamishima, T.; Hamasaki, M.; and Akaho, S. 2009. TrBagg: A simple transfer learning method and its application to personalization in collaborative tagging. In *Ninth IEEE International Conference on Data Mining*, 219 –228.

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

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

Pardoe, D., and Stone, P. 2010. Boosting for regression transfer. In *ICML '10*.

Pynadath, D. V., and Tambe, M. 2002. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research* 16:389–423.

Silver, D., and Veness, J. 2010. Monte-Carlo planning in large POMDPs. In *NIPS '10*.

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

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

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

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

Yao, Y., and Doretto, G. 2010. Boosting for transfer learning with multiple sources. In *CVPR '10*.