Cooperating with Unknown Teammates in Robot Soccer

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

Abstract

Many scenarios require that robots work together as a team in order to effectively accomplish their tasks. However, precoordinating these teams may not always be possible given the growing number of companies and research labs creating these robots. Therefore, it is desirable for robots to be able to reason about *ad hoc teamwork* and adapt to new teammates on the fly. This paper adopts an approach of learning policies to cooperate with past teammates and reusing these policies to quickly adapt to the new teammates. This approach is applied to the complex domain of robot soccer in the form of half field offense in the RoboCup simulated 2D league. This paper represents a preliminary investigation into this domain and presents a promising approach for tackling this problem.

1 Introduction

As the presence of robots grows, so does their need to cooperate with one another. Usually, multiagent research focuses on the case where all of the robots have been given shared coordination protocols before they encounter each other (Grosz and Kraus 1996b; Tambe 1997; Horling et al. 1999). However, given the growing number of research laboratories and companies creating new robots, not all of these robots may share the same coordination protocols. Therefore, it is desirable for these robots to be capable of learning and adapting to new teammates. This area of research is called *ad hoc teamwork* and focuses on the scenario in which the developers only design a single agent or small subset of agents that need to adapt to a variety of teammates (Stone et al. 2010).

Previous research into ad hoc teamwork has established a number of theoretical and empirical methods for handling unknown teammates, but these analyses largely focus on simple scenarios that may not be representative of the real problems that robots may encounter in ad hoc teams. This paper moves towards these complex problems by investigating ad hoc teamwork in the RoboCup 2D simulation domain, a complex, multiagent domain in which teams of agents coordinate their actions in order to compete against other intelligent teams. This domain requires effective cooperation between the agents while acting in a world with continuous states and continuous actions, all while facing complex opponents. The main contribution of this paper is the introduction of an algorithm for selecting effective actions for cooperating with a variety of teammates in this domain. This paper also presents a preliminary empirical evaluation of this approach. Overall, this paper serves as a preliminary exploration of ad hoc teamwork in a complex domain and discusses a promising approach for creating intelligent ad hoc team agents for this problem.

To quickly adapt to new teammates, it is imperative to use knowledge learned about previous teammates. This paper adopts the method of learning a policy of how to cooperate with previous teammates and then reuses this knowledge to cooperate with new teammates. Learning these policies is treated as a reinforcement learning problem, where the inputs are features derived from the world state and the actions are soccer moves such as passing or dribbling. Then, the agent can select from these policies on the fly to adapt to various teammates.

The remainder of this paper is organized as follows. Section 2 situates this research in the literature, and then Section 3 describes the problem in more depth. Section 4 describes the methods used to tackle this problem, and Section 5 presents preliminary results. Finally, Section 6 concludes.

2 Related Work

Multiagent teamwork is a well studied area of research. Previous research into multiagent teams has largely focused on creating shared protocols for coordination and communication which will enable the team to cooperate effectively. For example, the STEAM (Tambe 1997) framework has agents build a partial hierarchy of joint actions. In this framework, agents monitor the progress of their plans and adapt their plans as conditions change while selectively using communication in order to stay coordinated. Alternatively, the Generalized Partial Global Planning (GPGP) (Decker and Lesser 1995) framework considers a library of coordination mechanisms from which to choose. Decker and Lesser argue that different coordination mechanisms are better suited to certain tasks and that a library of coordination mechanisms is likely to perform better than a single monolithic mechanism. In SharedPlans (Grosz and Kraus 1996a), agents communicate their intents and beliefs in order to reason about coordinating joint actions, while revising these intents as conditions change.

While pre-coordinated multiagent teams are well studied, there has been less research into teams in which this pre-coordination is not available. This work builds on the ideas presented by Barrett et al. (2013), specifically learning models about past teammates and using these models to quickly adapt to new teammates. However, that work focused on a simpler domain in the form of a grid world, while this work investigates a complex, simulated robotics domain. One early exploration of ad hoc teamwork is that of Brafman and Tennenholtz (1996) in which a more experienced agent must teach its novice teammate in order to accomplish a repeated joint task. Another line of research, specifically in the RoboCup domain, was performed by Bowling and McCracken (2005). In their scenario, an agent has a different playbook from that of its teammates and must learn to adapt. Their agent evaluates each play over a large number of games and predicts the roles that its teammates will adopt.

Other research into ad hoc teams includes Jones et al.'s (2006) work on pickup teams cooperating in a domain involving treasure hunts. Work into deciding which teammates should be selected to create the best ad hoc team was performed by Liemhetcharat and Veloso (2012). Additional work includes using stage games and biased adaptive play to cooperate with new teammates (Wu, Zilberstein, and Chen 2011). However, the majority of these works focus on simple domains and provide the agent with a great amount of expert knowledge.

A very closely related line of research is that of opponent modeling, in which agents reason about opponents rather than teammates. This area of research largely focuses on bounding the worst case scenario and often restricts its focus onto repeated matrix games. One interesting approach is the AWESOME algorithm (Conitzer and Sandholm 2007) which achieves convergence and rationality in repeated games. Valtazanos and Ramamoorthy explore modeling adversaries in the RoboCup domain in (2013). 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.

3 Problem Description

In this paper, we consider the scenario in which an ad hoc agent must cooperate with a team of agents that it has never seen before. If the agent is given coordination or communication protocols beforehand, it can easily cooperate with its teammates. However, in scenarios where this prior shared knowledge is not available, the agent should still be able to cooperate with a variety of its teammates. Specifically, an agent should be able to leverage its knowledge about previous teammates in order to help it when it encounters new teammates.

3.1 Ad Hoc Teamwork

Evaluating an ad hoc team agent is difficult because the evaluation relies on how well the agents can cooperate with various teammates; we only create a single agent rather than an entire team. Therefore, to estimate the performance of agents in this setting, we adopt the evaluation framework proposed by Stone et al. (2010). In this work, Stone et al. formulate the performance of an ad hoc team agent as explicitly depending on the teammates and domains that it may encounter. Therefore, we describe the domain and teammates that the ad hoc team agents will encounter in the following sections.

3.2 RoboCup 2D Simulation Domain

The 2D Simulation League is one of the oldest leagues in RoboCup and is therefore one of the best studied, both in competition and research. In this domain, teams of 11 autonomous agents play soccer on a simulated 2D field for two 5 minute halves. The game lasts for 6,000 simulation steps, each lasting 100 ms. At each of these steps, these agents receive noisy sensory information such as their location, the location of the ball, and the locations of nearby agents. After processing this information, agents select abstract actions that describe how they move in the world, such as dashing, kicking, and turning. This domain is used as it provides a testbed for teamwork in a complex domain without requiring focus on areas such as computer vision and legged locomotion.

3.3 Half-Field Offense

Rather than use full 10 minute 11 on 11 game, this work instead uses the quicker task of half field offense introduced by Shivaram et al. (2007). In Half Field Offense (HFO), a set of offensive agents attempt to score on a set of defensive agents, including a goalie, without letting the defense capture the ball. A view of this game is shown in Figure 1. This task is useful as it allows for much faster evaluation of team performance than running full games as well as providing a simpler domain in which to focus on ways to improve ball control. In this work, we focus on the setting in which two offensive agents attempt to score on a defender and a goalie.

If the ball leaves the offensive half of the field or the defense captures the ball, the offensive team loses. If the offensive team scores a goal, they win. In addition, if no goal is scored within 500 simulation steps (50 seconds), the defense wins.

At the beginning of each episode, the ball is moved to a random location within the 25% of the offensive half closest to the midline. Let *length* be the length of the soccer pitch. Offensive players start on randomly selected vertices forming a square around the ball with edge length $0.2 \cdot \textit{length}$ with an added offset uniformly randomly selected in $[0, 0.1 \cdot \textit{length}]$. The goalie begins in the center of the goal, and the remaining defensive players start randomly in the back half of their defensive half.

In order to run some existing teams used in the RoboCup competition, it is necessary to field the entire 11 player team for the agents to behave correctly. Therefore, it is necessary to create the entire team and then constrain the additional players to stay away from play, only using the agents needed for half field offense. This approach may affect the players used in the HFO, but empirical tests have shown

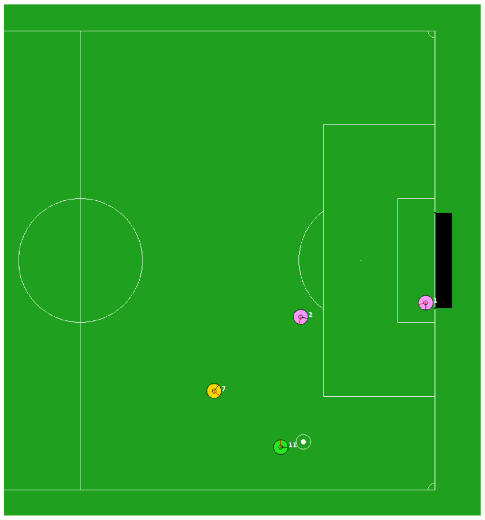
Figure 1: A screenshot of half field offense in the 2D soccer simulation league. The green agent is under our control, and the yellow player is its externally created teammate. These agents are trying to score against the two red defenders.

that the teams still perform well and that our ad hoc team agent can still adapt to these teams. We choose a fixed set of player numbers for the teammates, based on which player numbers tended to play offensive positions in observed play. The defensive players use the behavior created by Helios for the 2013 RoboCup competition, and the player numbers are similarly selected.

3.4 Teammates

In ad hoc teamwork research, it is important to use *externally-created* teammates to evaluate the various ad hoc team agents. Externally-created teammates are created by developers other than the authors and represent real agents that are created for the domain when developers do not plan for ad hoc teamwork scenarios. They are useful because their development is not biased to make them more likely to cooperate with ad hoc team agents. As part of the 2D simulation league competition, teams are required to release binary versions of their agents following the competition. Specifically, we use the binary releases from the 2013 competition. These agents provide an excellent source of *externally-created* teammates with which to test the possible ad hoc team agents. Specifically, we use the Helios and Yushan agents in this work, as well as the Base agent from the Helios code release.

4 Methods

Intelligent behaviors for cooperating with teammates can be difficult to design, even if the teammates' behaviors are known. In the 2D simulation league, there is the added difficulty of the domain being noisy and requiring many low level actions to accomplish any given task. To address this challenge, we propose that the agent employ a probabilistic planner that uses a stochastic model of the domain and learns models of its teammates.

4.1 MDP Formulation

Before specifying how to approach this ad hoc teamwork problem, it is helpful to describe how to model the problem. Specifically, we model the problem as a Markov Decision Process (MDP), which is a standard formalism in reinforcement learning (Sutton and Barto 1998) for describing an agent interacting with its environment. An MDP is 4-tuple (S, A, P, R), where S is a set of states, A is a set of actions, $P(s'|s, a)$ is the probability of transitioning from state s to s' when after taking action a, and $R(s, a)$ is a scalar reward given to the agent for taking action a in state s. In this framework, a policy π is a mapping from states to actions, which defines an agent's behavior for every state. The agent's goal is find the policy that maximizes its long term expected rewards. For every state-action pair, $Q^*(s, a)$ represents the maximum long term reward that can be obtained from (s, a) and is defined by solving the Bellman equation

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q^*(s',a')$$

where $0 < \gamma < 1$ is the discount factor representing how much more immediate rewards are worth compared to delayed rewards. The optimal policy π^* can then be derived by choosing the action a that maximizes $Q^*(s, a)$ for every $s \in S$. We now describe how to model the HFO domain as an MDP.

State A state $s \in S$ describes the current positions, orientations, and velocities of the agents as well as the position and velocity of the ball. Currently, the noiseless versions of these values are used, but the full noisy estimations provided by the simulator will be used in future work.

Actions In the 2D simulation league, agents act by selecting whether to dash, turn, or kick and specify values like the power and angle to kick at. Combining these actions to accomplish the desired results is a difficult problem. Therefore, this work builds on the code release by Helios (Akiyama 2010). This code release provides for a number of high level actions, such as passing, shooting, or moving to a specified point.

We use a set of 5 high level actions when the agent has the ball:

1. Shoot – shoot the ball at the goal, avoiding any opponents.
2. Short dribble – dribble the ball a short distance, maintaining control of the ball.
3. Long dribble – dribble the ball a longer distance, which will require chasing the ball down.
4. Cross – perform a crossing pass near the goal.
5. Pass – perform a direct, leading, or through pass to the teammate.

Each action considers a number of possible movements of the ball and evaluates their effectiveness given the locations of the agent's opponents and teammates. Each action therefore represents a number of possible actions that are reduced to discrete actions using the Helios evaluation function. While using these high level actions restricts the possibilities that the agent can take, it also enables it to learn more quickly and prune out ineffective actions, allowing it to select more intelligent actions with less samples.

Transition Function The transition function is defined by a combination of the simulated physics of the domain as well as the actions selected by the other agents. The agent does not directly model this function; instead it stores samples observed from played games as described in Section 4.2.

Reward Function The reward function is 1,000 when the offense wins, -1,000 when the defense wins, and -1 per each time step taken in the episode. The value of 1,000 is chosen to be greater than the effects of step rewards over the whole episode, but not completely outweigh these effects. Other values were tested with similar results.

4.2 Collecting Data

Rather than explicitly model the MDP's transition function, the agent directly uses samples taken from HFO games with its teammates. For each teammate, the agent plays in a number of games in which it explores the different available actions. From each action, the agent stores the tuple $\langle s, a, r, s' \rangle$, where s is the original state, a is the action, r is the reward, and s' is the resulting state.

We treat an action as going from when the agent begins the action until the ball is either once again held by the offense or the episode has ended due to the ball leaving play, the defense controlling the ball, or a goal occurring. Given that we only control a single agent, when its teammate has the ball, it will follow its own policy. We also store the results of the teammates actions in the tuple $\langle s, r, s' \rangle$ as we do not track their actions and our agent follows a fixed policy when away from the ball. If we could control all of the agents, we would instead consider the action from the teammate's perspective. If we treated this problem as a single agent problem, we would instead consider an action to end only when our agent once again controls the ball.

For each action, the agent observed 10,000 transitions. Similarly, the agent observed 10,000 transitions where its teammate had the ball, resulting in a total of 60,000 observations. This number comes from the 6 types of transitions, coming from the 5 actions and the transitions when the agent is away from the ball.

There are many ways to represent the state. Ideally, we want a compact representation that allows the agent to learn quickly by generalizing its knowledge about a state to similar states without over-constraining the policy. Therefore, we select 8 features:

- X position – the agent's x position on the field
- Y position – the agent's y position on the field
- Orientation – the direction that the agent is facing
- Goal opening angle – the size of the largest open angle of the agent to the goal
- Teammate's goal opening angle – the teammate's goal opening
- Distance to opponent – the distance to the closest opponent
- Distance from teammate to opponent – the distance from the teammate to the closest opponent
- Pass opening angle – the open angle available to pass to the teammate

4.3 Learning a Policy

Given the stored observations from the previous section, it is possible to learn a policy from these observations. Specifically, we use the Fitted Value Iteration (FVI) algorithm analyzed by Szepesvari and Munos (2005). Similar to Value Iteration (VI), FVI iteratively backs up rewards to improve its estimates of the values of states. Rather than looking at every state and every possible outcome from each state, FVI uses samples of these states and outcomes to approximate the values of state-action pairs. At each iteration, the agent updates the following equation for each tuple

$$Q(s, a) = r + \gamma * V(s')$$

where $V(s')$ is initialized to 0. Then, we update our estimates of V via

$$V(s') = max_{a'} Q(s', a')$$

As $Q(s', a')$ is not directly observed, we estimate its values using function approximation as discussed later in this section.

We also consider states in which the teammate has the ball, where our agent follows a fixed policy. During this time, we treat this as a single action that the agent must take.

To generalize its knowledge to states and actions it has not observed, the agent employs function approximation. While there are many forms of function approximation, preliminary tests showed that neural networks performed better than other methods in this setting. The effectiveness of neural networks combined with FVI is backed up by existing research (Riedmiller 2005). However, neural networks have a number of open parameters such as the number of hidden nodes, the number of layers, and activation functions. To explore these parameters and select effective values, we used a grid search over these parameters. For each parameter set, we ran FVI to learn a policy and then tested this policy in 1,000 games.

While this work is focused on learning policies for when the agent has the ball, considering its policy away from the ball is very important. In RoboCup, positioning of agents and assigning robots to roles is often crucial to the team's performance (MacAlpine et al. 2012). However, the base code release by Helios (Akiyama 2010) uses a static role assignment system, where our agent always attempts to play the "Center Forward" position. This role is not ideal for the HFO setting as it prevents passing in many scenarios. Therefore, we use a simple heuristic function to assign the agent to the role that is closest to a desired offset from its teammate. To find this desired offset, we similarly perform a grid search over possible x and y offsets.

5 Preliminary Results

This section presents the preliminary results of this work. Results are for 10,000 games of half field offense.

Team	Fraction Scored
Random	0.248
Match	0.292
Random and Positioning	0.338
FVI using Linear Regression with RBFs	0.341
FVI using Neural Network	0.411
FVI using Neural Network and Positioning	0.698

Table 1: Importance of different factors when cooperating with a Helios 2013 teammate against two Helios defenders.

		Policy Learned With		
True Type		Base	Yushan	Helios
	Base	0.386	0.309	0.340
	Yushan	0.427	0.474	0.421
	Helios	0.478	0.489	0.698

Table 2: Cooperating with different teammates against two Helios defenders.

5.1 Importance of Factors of the Approach

First, let us investigate the importance of different factors of the approach, such as different learning methods and positioning approaches. The baseline performance in this domain is given by acting randomly given the actions provided and matching the externally created teammates' behaviors. The results in Table 1 show that the learning a policy using FVI can improve greatly performance, going from a random performance of 0.248 to 0.411, an improvement of 65%. In addition, considering the positioning of the agent also improves performance a large amount, especially when using a learned policy. The total improvement is from 0.248 to 0.698, an improvement of over 180%. Computing the value function for a given set of parameters takes less than 2 minutes on average. Once the value function is computed, selecting actions can be done at real time for the simulator.

5.2 Cooperating with Different Teammates

If the agent encounters a teammate that is following an unknown behavior, the agent can reuse a policy it has learned from a past teammate. Table 2 show the results of using policies learned with past teammates with the current teammate. The three teammate types considered are the Helios 2013 agent, the Base agent released by Helios in 2013 that our agent is based on, and the Yushan 2013 agent. We do not use the WrightEagle agent due to its problems running in synchronized mode.

These results show that using an incorrect model of the teammates does hurt performance. Using the policy learned with the Helios agent does work well with all of the other teammates, but there is still room for improvement. Therefore, it is desirable for the agent to select which policy to follow on the fly.

One approach for this is to treat the problem as a multi-armed bandit where the different arms are the policies to follow. However, this approach requires many games to learn as each pull of an arm corresponds to playing a game of HFO. In addition, this approach requires many observations of each policy with the new teammate.

It is desirable to learn more quickly, which requires using information of the teammate's actions. To this end, it is possible to compare the teammate's movements to the stored $\langle s, a, r, s' \rangle$ tuples. Specifically, for the n closest states, s, we can compare the resulting state to the stored s' and calculate the distance of the teammate from its expected position. Assuming that the differences are distributed normally allows us to calculate the probability of the teammate ending up in the observed position. Then, the agent can update its beliefs about its teammate's type using a Bayesian update. However, this problem remains an open problem.

6 Conclusion

The majority of past research on ad hoc teamwork has focused on domains such as grid worlds or matrix games and in which only a few agents interact. However, the dream of applying ad hoc teamwork techniques to real world problems requires scaling these methods up to more complex and noisy domains that involve many agents interacting. This work presents a step towards this goal by investigating ad hoc teamwork in the area of simulated robot soccer. This research is a preliminary report focusing on how to approach this complex problem. This paper presents an approach for learning models that predict teammates' actions as an effective way of representing knowledge learned from past teammates. Then, we show how this knowledge can be used to learn intelligent behaviors using a sample-based learning algorithm to select high level actions.

This paper represents a move towards scaling up reasoning about ad hoc teamwork to a complex domain involving many agents. One area for future work is applying this approach to real robots; for example, in future drop-in player challenges run at the RoboCup competition, such as those held in 2013 in the 2D simulation league, the 3D simulation league, and the Standard Platform League (SPL). Another interesting avenue for future research is reasoning about agents that are learning about the ad hoc agent over time. In the long term, an interesting application of ad hoc teamwork is to allow agents to cooperate with either other agents or humans by considering them as additional agents in the domain.

References

Akiyama, H. 2010. Agent2d base code release. http://sourceforge.jp/projects/rctools.

Barrett, S.; Stone, P.; Kraus, S.; and Rosenfeld, A. 2013. Teamwork with limited knowledge of teammates. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*.

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.

Decker, K. S., and Lesser, V. R. 1995. Designing a family of coordination algorithms. In *ICMAS '95*, 73–80.

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.

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., and Kraus, S. 1996a. Collaborative plans for complex group actions. *Artificial Intelligence* 86:269–368.

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

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

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.

Kalyanakrishnan, S.; Liu, Y.; and Stone, P. 2007. Half field offense in RoboCup soccer: A multiagent reinforcement learning case study. In *RoboCup-2006: Robot Soccer World Cup X*, volume 4434 of *Lecture Notes in Artificial Intelligence*. Berlin: Springer Verlag. 72–85.

Liemhetcharat, S., and Veloso, M. 2012. Modeling and learning synergy for team formation with heterogeneous agents. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, 365–374. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

MacAlpine, P.; Urieli, D.; Barrett, S.; Kalyanakrishnan, S.; Barrera, F.; Lopez-Mobilia, A.; Ştiurcă, N.; Vu, V.; and Stone, P. 2012. UT Austin Villa 2011: A champion agent in the RoboCup 3D soccer simulation competition. In *Proc. of 11th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*.

Riedmiller, M. 2005. Neural fitted q iteration first experiences with a data efficient neural reinforcement learning method. In Gama, J.; Camacho, R.; Brazdil, P.; Jorge, A.; and Torgo, L., eds., *Machine Learning: ECML 2005*, volume 3720 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 317–328.

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

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

Szepesvári, C., and Munos, R. 2005. Finite time bounds for sampling based fitted value iteration. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, 880–887. New York, NY, USA: ACM.

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

Valtazanos, A., and Ramamoorthy, S. 2013. Bayesian interaction shaping: Learning to influence strategic interactions in mixed robotic domains. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, 63–70. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

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