Defining and Using Ideal Teammate and Opponent Agent Models

Peter Stone[1], Patrick Riley[2], and Manuela Veloso[2]

[1]AT&T Labs — Research
180 Park Ave., room A273
Florham Park, NJ 07932
pstone@research.att.com
www.research.att.com/~pstone

[2]Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
{pfr,veloso}@cs.cmu.edu
www.cs.cmu.edu/{~pfr,~mmv}

Submitted to the Fourth International Conference on MultiAgent Systems (ICMAS-2000), November 1999

Abstract

A common challenge for agents in multiagent systems is trying to predict what other agents are going to do in the future. Such knowledge can help an agent determine which of its current action options is most likely to help it achieve its goals. There is a long history in adversarial game playing of using a model of an opponent which assumes that it always acts optimally, as in minimax search. However, minimax is designed for turn-taking adversarial domains with complete information and discrete actions. In contrast, our research focuses on adversarial domains in which the agents have incomplete information, noisy sensors and actuators, and a continuous action space. We introduce "ideal-model-based behavior outcome prediction" (IMBBOP) which models the results of other agents' future actions in relation to their optimal actions based on an ideal world model. Our technique also includes a method for relaxing this optimality assumption. We define IMBBOP and illustrate its use within the simulated robotic soccer domain. We include empirical results demonstrating the effectiveness of IMBBOP.

Keywords: RoboCup, multiagent robotics, using agent models, teamwork and cooperation

1 Introduction

A common challenge for agents in multiagent systems is trying to predict what other agents are going to do in the future. Such knowledge can help an agent determine which of its current action options are most likely to help it achieve its goals.

There is a long history in adversarial game playing of using a model of an opponent which assumes that it always acts optimally. For example, in the minimax search algorithm, one enumerates all the possible actions that the other agent may take and then always assumes that the other agent takes the action that is the ideal action from its own viewpoint, which would be the worst action for us. This means that minimax acts safely, but not opportunistically: it maximizes worst-case performance. If the other agent does not perform its ideal action, then minimax's choice yields an outcome better than predicted.

Minimax is designed for turn-taking adversarial domains with complete information and discrete actions (such as chess). In contrast, our research focuses on adversarial domains in which the agents have incomplete information, noisy sensors and actuators, and a continuous action space (such as adversarial robotic control). Nonetheless, we are able to build on one key feature of minimax—the use of a model of the other agent's future actions in relation to its theoretical optimal actions—in our model-based technique, which we call "ideal-model-based behavior outcome prediction" (IMBBOP). Our technique also includes a method for relaxing this optimality assumption.

An alternative to minimax, in which other agents aren't necessarily assumed to act optimally, is the recursive modeling method (RMM) [3]. Using RMM, an agent models the internal state and action selection strategy of another agent in order to predict its actions. This method is recursive because the other agent might similarly be modeling the original agent, leading to an arbitrary depth of reasoning (techniques for limiting this depth have been studied [2, 10]).

A limitation of both minimax and RMM is that they rely on knowing the state of other agents and their action capabilities in order to construct payoff matrices. In contrast, our research is concerned with situations in which the agent's state and action capabilities may not be known.

Past approaches have examined methods for deducing agents' action capabilities through observation [11]; deducing agent's plans given their actions [4]; and deducing agents' actions given incomplete observations of their states [9]. All of these approaches address situations in which an agent does not have the information necessary to determine the optimal actions of other agents in the environment.

IMBBOP addresses similar situations, but differs from all of these approaches in that it does not directly deduce an agent's actions, plans, or capabilities. Rather, it uses an idealized world model and observable agent state information to estimate the agent's optimal action. It characterizes the *actual* capabilities of the agent in relation to this estimated optimal action.

The remainder of this paper is organized as follows. Section 2 presents our definition of ideal-model-based behavior outcome prediction. Section 3 describes our application of IMBBOP which was successfully implemented as a part of the CMUNITED-99 simulated robotic soccer team. Section 4 presents the results of CMUNITED-99 in an international tournament as well as controlled experiments highlighting the effectiveness of IMBBOP. Section 5 presents conclusions and future work.

2 Ideal-Model-Based Behavior Outcome Prediction

IMBBOP is designed for situations in which an agent X has a goal G to be achieved by time T. X must determine whether agent Y can prevent (if an "opponent") or achieve (if a "teammate")[1] G after X takes action A. In particular, X must determine which of its possible actions $A_1, \ldots, A_n$ is most likely to achieve G by time T.

IMBBOP makes the following assumptions:

- X must select an action from among $A_1, \ldots, A_n$ to be executed immediately. It then ceases to affect the achievement of G.
- Whether or not Y can achieve or prevent X's goal depends on T. That is, $\exists t$ s.t. Y could achieve G by, or prevent G from being achieved by, time t.
- X has a model of the world dynamics.
- X has incomplete information regarding Y's current state.
- X has an incomplete model of Y's capabilities (how it can affect the world). That is, X knows (through the world model) what actions Y can take, but has no model about how Y chooses its action. However, based on the world

[1] Here we consider an agent to be a teammate if it also has the goal G and to be an opponent if it has the goal of preventing G. We assume that X knows which agents are teammates and which are opponents.

model, X can deduce an upper bound on Y's capabilities in terms of the minimum time necessary to execute tasks. For example, the world model could specify a maximum possible agent speed.

Given these assumptions, IMBBOP works as follows.

1. Using the model of world dynamics and the resultant upper bounds on agent capabilities, determine analytically the minimum t such that Y could prevent or achieve G by time t after X takes action A.

2. Use a threshold on $T - t$ to predict whether not action A will succeed: the greater $T - t$, the more likely Y is to be able to prevent or achieve G by time T. Thus, $T - t$ is an indication of the likelihood that action A will result in goal G being achieved by time T.

In step 1, such an analysis is made possible under the simplifying assumption that the world dynamics and action capabilities of Y are known. In addition, X fills in missing information about Y with best-case values from Y's perspective (i.e., if Y could be in one of n states, X assumes that Y is in the state from which it could most quickly achieve or prevent G). Note that there is no guarantee that Y could *actually* achieve G by time t.

For example, if Y is currently located at location (x_1, y_1) and must get to location (x_2, y_2) in order to prevent G, then, using a theoretical maximum speed of s, X could compute analytically that Y cannot get to location (x_2, y_2) in time less than $\frac{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}{s}$. In actual fact, it may be unlikely that Y could actually arrive at (x_2, y_2) so quickly given the time necessary for it to figure out that it needs to get there and possibly accelerate to the maximum speed.

In practice, X will execute action A based on whether or not $T - t$ exceeds some threshold.

3 IMBBOP in Robotic Soccer

3.1 RoboCup

Our IMBBOP implementation is carried out in the simulated robotic soccer domain using the RoboCup soccer server [1, 5]. In this domain, there are 22 agents, each acting up to 10 times per second. Each agent gets local, incomplete perceptory information, making it impossible to determine another agent's impression of the world state based only upon the actual world state. Sensors, actuators, and world dynamics are all noisy. However, both the ball and players have maximum speeds enforced by the simulation.

Over the past several years, we have created teams of soccer-playing agents for use in the RoboCup simulator. The teams are all called "CMUnited-XX," where "XX" indicates the year in which they first participated in the RoboCup international simulator tournament. For example, the most recent incarnation, "CMUnited-99," was introduced at the RoboCup-99 tournament in Stockholm, Sweden which was held in August of 1999.

Although CMUnited-98 [8], the champion of RoboCup-98, out-scored its opponents by a combined score of 66–0, it failed to score on many opportunities in which it had the ball close to the opponent's goal, especially against the better opponents. In order to improve its performance, we introduced IMBBOP into the CMUnited-99 team. This improvement turned out to be essential, since several teams were able to beat CMUnited-98 at RoboCup-99, but CMUnited-99 won all of its games by a combined scored of 110–0 and was awarded the RoboCup-99 simulator league championship.

3.2 IMBBOP in CMUnited-99

IMBBOP is used in several ways in CMUnited-99. Most significantly in terms of performance, it is used to decide when to shoot and when to pass when an agent has the ball very near to the opponent's goal. It is also used by agents to determine when the opponents are likely to be able to steal the ball from them.

While IMBBOP is principally concerned with predicting the outcomes of other agents' behaviors, it also makes use of a model of the agent's own action outcomes. In general, it is possible to predict an agent's own action outcomes via empirical testing. For example, we determined empirically that an agent can generally position the ball and kick it with high power in a chosen direction in 4 or fewer simulator cycles. While 4 is not a hard upper bound on the number of cycles due to the noise in the simulator, it is an empirically reliable estimate and is used in our estimate of the number cycles it will take an opponent to steal the ball.

3.2.1 When to Shoot

One of the most important offensive decisions in robotic soccer is the decision of when to shoot the ball towards the goal. In CMUnited-98, decisions about when to shoot were made in one of three ways:

- Based on the distance to the goal
- Based on the number of opponents between the ball and the goal
- Based on a decision tree.

Each of these methods is imperfect. Distance to goal completely ignores how the opponents are positioned. The number of opponents between the ball and the goal does not accurately reflect in how *good* of a position the defenders are. Lastly, the decision tree was trained for passing[6], so its performance on the related but different behavior of shooting is questionable.

CMUNITED-99 makes this decision in a more principled way by using a model of an "optimal" opponent goalie. That is, we use a model of a goalie that reacts instantaneously to a kick, moves to exactly the right position to stop the ball, and catches with perfect accuracy.

When deciding whether to shoot, the agent first identifies its best shot target. It generally considers two spots, just inside each of the two sides of the goal. The agent then considers the lines from the ball to each of these possible shot targets. *shot-target* is the position whose line is further from the goalie's current position.

The agent then predicts, given a shot at *shot-target*, the ball's position and goalie's reaction using the optimal goalie model. We use the following predicates:

blocking-point The point on the ball's path for which an optimal goalie heads.

ball-to-goalie-cycles The number of cycles for the ball to get to the *blocking-point*

goalie-to-ball-cycles The number of cycles for the goalie to get to the *blocking-point*

shot-margin =*ball-to-goalie-cycles*−*goalie-to-ball-cycles*

better-shot(k) Whether teammate k has a better shot than the agent with the ball, as judged by *shot-margin*

The value of *ball-to-goalie-cycles* corresponds to T in our definition of IMB-BOP in Section 2, while *goalie-to-ball-cycles* corresponds to t. The value *shot-margin* is a measure of the quality of the shot. The smaller the value of *shot-margin*, the more difficult it will be for the goalie to stop the shot. For example, for a long shot, the ball may reach the *blocking-point* in 20 cycles (*ball-to-goalie-cycles*$= 20$), while the goalie can get there in 5 cycles (*goalie-to-ball-cycles*$= 5$). This gives a *shot-margin* of 15. In terms of IMBBOP as defined in Section 2, $T = 20$, $t = 5$, and $T - t = 15$. This is a much worse shot than if it takes the ball only 12 cycles (*ball-to-goalie-cycles*$= 12$) and the goalie 10 cycles to reach the *blocking-point* (*goalie-to-ball-cycles*$= 10$). The latter shot has a *shot-margin* of only 2 ($T = 12, t = 10, T - t = 2$). Further, if *shot-margin*< 0 ($T - t < 0$), then the "optimal" goalie could not reach the ball in time, and the shot should succeed.

3.2.2 When to Pass

Using a model of opponent behavior gives us a more reliable and adaptive way of making the shooting decision. We can also use it to make better passing decisions via a model of teammate behavior outcomes. When near the goal, the agent may often be faced with the decision about whether to pass or shoot the ball. The agent with the ball simulates the situation where its teammate is controlling the ball, using the goalie model to determine how good of a shot the teammate has. If the teammate has a much better shot, then the predicate *better-shot*(k) will be true. This indicates that the agents should pass the ball rather than shooting itself.

In this case, the agent is using an optimal model of both the teammate *and* the opponent goalie. The *shot-target* is computed from the teammate's perspective, and the speed at which the teammate will be able to propel the ball towards the goal is assumed to be as high as possible according to the world model. The *blocking-point*, *ball-to-goalie-cycles*, and *goalie-to-ball-cycles* are all computed from the teammate's and goalie's current positions. Now, since we are primarily predicting the teammate's performance, T is the *goalie-to-ball-cycles* and t is the *ball-to-goalie-cycles*. The greater the value of $T - t$, the more likely the teammate is to succeed in getting the ball past the goalie.

There is one complication here; it takes some time to pass the ball. In the time that elapses during a pass, the world will change, and the receiving agent may then decide the original agent has a better shot. This could potentially lead to passing loops where neither agent will shoot. CMUNITED-99 does two things to avoid this loop. First, the agent will only pass to a teammate with a better shot if, given the current state of the world, the goalie cannot stop the shot ($T - t > 0$). Secondly, the extra time difference between the passing and receiving agents must be greater than some threshold (5 in CMUNITED-99).

Note that this analysis of shooting ignores the presence of defenders. Just because the goalie can not stop the shot (as judged by the optimal goalie model)

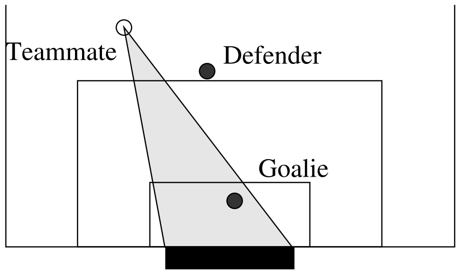

Figure 1: The Breakaway Cone

does not mean that a nearby defender can not run in to kick the ball away.

3.2.3 Breakaways

An important idea in many team ball sports like soccer is the idea of a *breakaway*. Intuitively, this is when some number of offensive players get the ball and themselves past the defenders, leaving only perhaps a goalie preventing them from scoring. Shooting and passing at the proper time is particularly important on breakaways. If the agent shoots too early, the goalie will have plenty of time to stop the ball. If the agent shoots too late, then the goalie may have time to get the ball before the kick is complete.

After looking at log-files from previous competitions, we saw many opportunities for breakaways which were not taken advantage of. We therefore incorporated the above technique for using a model of teammates and opponent goalies into a special-purpose breakaway behavior.

The first question which has to be answered is "What exactly is a breakaway?" This is built upon several predicates:

controlling-teammate Which teammate (if any) is currently controlling the ball. "Control" is judged by whether the ball is in the area defined by the simulator within which the player can physically kick the ball.

controlling-opponent Which opponent (if any) is currently controlling the ball

opponents-in-breakaway-cone The breakaway cone is shown in Figure 1. The cone has its vertex at the player with the ball and extends to the opponents goal posts.

teammates-in-breakaway-cone The same as the previous definition, but for the other side of the field. This is used when judging whether the opponents currently have a breakaway.

our-breakaway = (*controlling-teammate*$\neq$ None) $\wedge$ (*controlling-opponent*=None) $\wedge$ (*opponents-in-breakaway-cone*$\leq$1)

their-breakaway = (*controlling-opponent*$\neq$ None) $\wedge$ (*controlling-teammate*=None) $\wedge$ (*teammates-in-breakaway-cone*$\leq$1)

Once the agent determines that it is on a breakaway (when *our-breakaway* is true), it starts dribbling the ball roughly towards the opponent's goal (actually slightly across the front of the goal). It continues to do so until deciding to shoot, at which point it kicks the ball as hard as possible towards a corner of the goal (*shot-target*). The decision of when to stop dribbling in order to shoot is the key decision when on a breakaway.

We use the optimal model described above to help make this decision. During a breakaway, the agent shoots when either one of the following is true:

1. *shot-margin* (or $T-t$) gets below a certain threshold (1 cycle in CMUNITED-99)
2. The time that it would take for the goalie to proceed directly to the ball and steal it gets below a certain threshold (6 cycles in CMUNITED-99). This time is again determined analytically using an optimal model of the goalie's movement capabilities (See the description of "cycles to steal" in Section 3.2.4).

This skill was extremely effective in the competition, with the vast majority of our goals being scored using the specialized breakaway code.

3.2.4 Cycles to Steal

CMUNITED-99 makes use of teammate and opponent models most prominently in its special-purpose breakaway behavior. However, it also makes use of opponent models to determine whether or not an agent can keep control of the ball without an opponent stealing it. Such an ability is a prerequisite for an agent being able to safely *dribble* (interleave short kicks and dashes in a given direction so that the agent in effect moves with the ball).

In this case, it is necessary to determine whether the nearest opponent could get to the ball in less time than it would take to safely kick the ball away. Thus, T is the time it would take to kick the ball away. In CMUNITED-99, we use the constant $T = 4$ since an agent can generally position the ball and kick it with high power in a chosen direction in 4 simulator cycles.

The time (t) it would take the opponent to get to the ball's current position is computed based on the opponent's current position, the maximum speed of the opponent in the simulator, and an estimate of how long it would take the opponent to move around the agent to get to the ball (only if the agent is between the ball and the opponent).

The agent bases its decision of whether or not to dribble based on a threshold of on $T - t$. In CMUNITED-99, players only dribble when $T - t < -3$. That is, they need to be fairly certain that the opponent will not be able to steal the ball before they are willing to dribble.

4 Results

In this section we present empirical results demonstrating the effectiveness of IMBBOP in the simulated robotic soccer domain. First we evaluate the performance of an agent on a breakaway when using IMBBOP. Second, we present anecdotal evidence of IMBBOP's usefulness to the team as a whole.

4.1 Controlled Testing

In order to test the effectiveness of IMBBOP in simulated robotic soccer, we ran simulations involving only 2 players: a goalie and a striker. The striker and ball were repeatedly placed 30m away from the goal. The goalie was placed next to the goal. The task of the striker is to attempt to shoot the ball past the goalie into the goal, while the goalie aims to thwart the striker. This setup creates a breakaway situation as described in Section 3.2.3.

In all cases, the striker dribbles the ball roughly towards the goal until deciding to shoot, as described in Section 3.2.3. Meanwhile, the goalie must decide when to start moving towards the ball in an attempt to block it. At one extreme, it could wait until the player shoots, thereby ensuring that it will no longer be able to change the ball's direction. At the other extreme, it could immediately move towards the striker in an attempt to "cut down the angle," or reduce the amount of open goal from the striker's perspective.

In particular, the strategies we use for the goalie during testing are:

1. Wait for the striker to shoot the ball before trying to get it.

2. Once the striker gets to within 24m of the goal, run out to try and catch the ball.

3. Once the striker gets to within 35m of the goal (effectively immediately), run out to try and catch the ball.

Meanwhile, we tested several striker strategies with IMBBOP and without against each of the possible goalie strategies. As described in Section 3.2.3, the CMUNITED-99 strikers use two different types of opponent models when executing breakaways: one based on *shot-margin* (Condition 1 in Section 3.2.3) and one based on the predicted number of cycles it would take the goalie to steal the ball (Condition 2). Thus, the striker can use neither, either, or both models when deciding when to shoot. When using neither model, it shoots purely based on its distance to the goal.

Thus, the strategies for the striker are:

1. Use both models to determine when to shoot (conditions 1 and 2).

2. Use only the *shot-margin* model (condition 1).

3. Use only the stealing ball model (condition 2).

4. Shoot as soon as within 17m of the goal.

5. Shoot as soon as within 25m of the goal.

Each striker strategy was tested against each goalie strategy for 10,000 simulator cycles, which allows between 90 and 200 separate breakaway attempts. The percentage of breakaways that result in goals for each of these combinations of strategies is shown in Figure 2. The numbered goalie strategy indicated on the x-axis corresponds the goalie strategy as numbered above. When using both models, the striker performs better than when shooting based on distance in all cases. Note however, that when using only one of the models, the results are sometimes *worse* than shooting based just on distance. A goalie has two basic ways to stop a shot (as reflected in the two models): by getting in the way once the ball is kicked, or by stealing the ball before the shot can be taken. Using only one of the models only reflects one of these possibilities. On the other hand, a distance threshold takes into account both of these abilities *for a particular goalie strategy*.

For any particular breakaway, there is some distance from the goal at which the striker should shoot. This distance depends mostly on the strategy which the

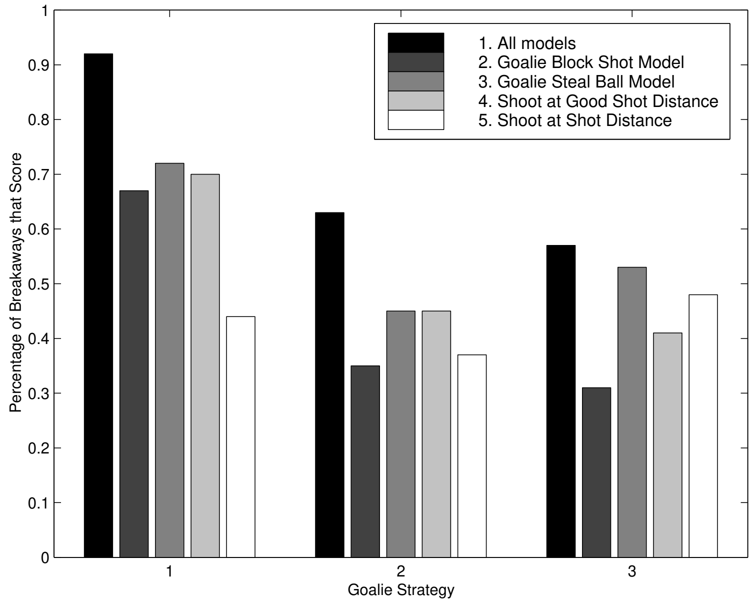

Figure 2: The effectiveness of different types of models. The percentage of breakaways which result in a goal is shown for various goalie strategies and uses of models. The numbered goalie strategy indicated on the x-axis corresponds the strategy as numbered in the text.

goalie is using. Therefore, for any particular goalie strategy, *some* distance threshold for shooting will probably perform quite well. The importance of using the goalie models is that with the models, the agent can perform well no matter what strategy the goalie uses. The players do not need to know the goalie strategy *a priori*, and if the goalie changes its strategy, the agents will adapt effectively.

4.2 RoboCup-99

IMBBOP has proven to be very useful to us in creating the CMUNITED-99 team of soccer-playing agents [7]. CMUnited-99 was built upon the CMUnited-98 team which won the RoboCup-98 simulator competition [8]. CMUnited-98 became publicly available after the 1998 competition so that other people could build upon our research. Thus, we expected there to be several teams at RoboCup-99 that could beat CMUnited-98, and indeed there were.

With this expectation, we knew that CMUnited-99 needed to improve upon CMUnited-98. With that in mind, we noticed that CMUNITED-98 often missed its breakaway opportunities. By incorporating both teammate and opponent models as described in Section 3 into CMUNITED-99(as well as several other improvements), we were able to create a new team that could score consistently (roughly one goal every minute) against CMUNITED-98.

Since there were several improvements over CMUNITED-98 incorporated into CMUNITED-99, and also since the RoboCup tournaments do not provide controlled testing environments, we cannot make any definite conclusions based on the competitions. However, watching the games, we noticed many goals scored as a result of well-timed shots and passes near the opponent goal. In the end, CMUnited-99 went on to win the RoboCup-99 championship, outscoring its opponents by a combined score of 110–0.

5 Conclusion and Future Work

Ideal-model-based behavior outcome prediction is potentially applicable and useful in any domain in which an agent does not know the states and action capabilities of other agents in the environment. By using a model of the world dynamics to determine an upper-bound on agent performance (the ideal), an agent's actual performance can be characterized in relation to this ideal.

When evaluating whether a potential action is likely to achieve an agent's goal, an agent using IMBBOP uses a time-based threshold ($T - t$) to represent the predicted maximum or minimum difference between another agent's actual performance and its ideal performance. Our work reported in this paper uses hard-wired thresholds. However, the opportunity exists for on-line learning of these threshold values. Learning models, or adaptively modeling other agents, is a part of our on-going and future research.

Meanwhile, our IMBBOP implementation has played a significant role in our successful development of a team of simulated robotic soccer-playing agents.

References

[1] Emiel Corten, Klaus Dorer, Fredrik Heintz, Kostas Kostiadis, Johan Kummeneje, Helmut Myritz, Itsuki Noda, Patrick Riley, Peter Stone, and Travlex Yeap. Soccer server manual, version 5.0. Technical Report RoboCup-1999-001, RoboCup, 1999. At URL http://ci.etl.go.jp/~noda/soccer/server/Documents.html.

[2] Edmund H. Durfee. Blissful ignorance: Knowing just enough to coordinate well. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 406–413, Menlo Park, California, June 1995. AAAI Press.

[3] P. J. Gmytrasiewicz, E. H. Durfee, and D. K. Wehe. A decision-theoretic approach to coordinating multiagent interactions. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 62–68, 1991.

[4] Marcus J. Huber and Edmund H. Durfee. Deciding when to commit to action during observation-based coordination. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 163–170, Menlo Park, California, June 1995. AAAI Press.

[5] Itsuki Noda, Hitoshi Matsubara, Kazuo Hiraki, and Ian Frank. Soccer server: A tool for research on multiagent systems. *Applied Artificial Intelligence*, 12:233–250, 1998.

[6] Peter Stone. *Layered Learning in Multi-Agent Systems*. PhD thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, December 1998. Available as technical report CMU-CS-98-187.

[7] Peter Stone, Patrick Riley, and Manuela Veloso. The CMUnited-99 champion simulator team. In Manuela Veloso, Enrico Pagello, and Hiroaki Kitano, editors, *RoboCup-99: Robot Soccer World Cup III*, Berlin, 2000. Springer Verlag.

[8] Peter Stone, Manuela Veloso, and Patrick Riley. The CMUnited-98 champion simulator team. In Minoru Asada and Hiroaki Kitano, editors, *RoboCup-98: Robot Soccer World Cup II*. Springer Verlag, Berlin, 1999.

[9] Milind Tambe. Recursive agent and agent-group tracking in a real-time , dynamic environment. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 368–375, Menlo Park, California, June 1995. AAAI Press.

[10] Jose M. Vidal and Edmund H. Durfee. Recursive agent modeling using limited rationality. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 376–383, Menlo Park, California, June 1995. AAAI Press.

[11] Xuemei Wang. Planning while learning operators. In *Proceedings of the Third International Conference on AI Planning Systems*, May 1996.