# Using Decision Tree Confidence Factors for Multiagent Control

Peter Stone and Manuela Veloso

Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
{pstone,veloso}@cs.cmu.edu
http://www.cs.cmu.edu/{~pstone,~mmv}

**Abstract.** Although Decision Trees are widely used for classification tasks, they are typically not used for agent control. This paper presents a novel technique for agent control in a complex multiagent domain based on the confidence factors provided by the C4.5 Decision Tree algorithm. Using Robotic Soccer as an example of such a domain, this paper incorporates a previously-trained Decision Tree into a full multiagent behavior that is capable of controlling agents throughout an entire game. Along with using Decision Trees for control, this behavior also makes use of the ability to reason about action-execution time to eliminate options that would not have adequate time to be executed successfully. This multiagent behavior represents a bridge between low-level and high-level learning in the Layered Learning paradigm. The newly created behavior is tested empirically in game situations.

## 1 Introduction

Multiagent Systems is the subfield of AI that aims to provide both principles for construction of complex systems involving multiple agents and mechanisms for coordination of independent agents' behaviors. As of yet, there has been little work with Multiagent Systems that require real-time control in noisy, adversarial environments. Because of the inherent complexity of this type of Multiagent System, Machine Learning is an interesting and promising area to merge with Multiagent Systems. Machine learning has the potential to provide robust mechanisms that leverage upon experience to equip agents with a large spectrum of behaviors, ranging from effective individual performance in a team, to collaborative achievement of independently and jointly set high-level goals. Especially in domains that include independently designed agents with conflicting goals (*adversaries*), learning may allow agents to adapt to unforeseen behaviors on the parts of other agents.

*Layered Learning* is an approach to complex multiagent domains that involves incorporating low-level learned behaviors into higher-level behaviors [13]. Using simulated Robotic Soccer (see Section 2) as an example of such a domain, a Neural Network (NN) was used to learn a low-level individual behavior (ballinterception), which was then incorporated into a basic collaborative behavior (passing). The collaborative behavior was learned via a Decision Tree (DT) [13].

This paper extends these basic learned behaviors into a full multiagent behavior that is capable of controlling agents throughout an entire game. This behavior makes control decisions based on the confidence factors associated with DT classifications—a novel approach. While operator success probabilities have previously been stored in tree structures [3], our work makes use of confidence measures originally derived from learning classification tasks. It also makes use of the ability to reason about action-execution time to eliminate options that would not have adequate time to be executed successfully. The newly created behavior is tested empirically in game situations.

The rest of the paper is organized as follows. Section 2 gives an overview of foundational work in the Robotic Soccer domain. The new behavior, along with explanations of how the DT is used for control and how the agents reason about action-execution time, is presented in Section 3. Extensive empirical results are reported in Section 4, and Section 5 concludes.

## 2 Foundational Work

This section presents brief overviews of Robotic Soccer research and of Layered Learning. Further details with regards to both topics can be found in [13].

### 2.1 Robotic Soccer

As described in [5], Robotic Soccer is an exciting AI domain for many reasons. The fast-paced nature of the domain necessitates real-time sensing coupled with quick behaving and decision making. Furthermore, the behaviors and decision-making processes can range from the most simple reactive behaviors, such as moving directly towards the ball, to arbitrarily complex reasoning procedures that take into account the actions and perceived strategies of teammates and opponents. Opportunities, and indeed demands, for innovative and novel techniques abound.

Robotic Soccer systems have been recently developed both in simulation [6, 9, 12, 14] and with real robots [1, 4, 10, 11, 15, 13]. While robotic systems are difficult, expensive, and time-consuming to use, they provide a certain degree of realism that is never possible in simulation. On the other hand, simulators allow researchers to isolate key issues, implement complex behaviors, and run many trials in a short amount of time. While much of the past research has used Machine Learning in constrained situations, nobody has yet developed a full behavior based on learning techniques that can be used successfully in a game situation.

The Soccer Server [7], which serves as the substrate system for the research reported in this paper, captures enough real-world complexities to be a very challenging domain. This simulator is realistic in many ways: (i) the players' vision is limited; (ii) the players can communicate by posting to a blackboard that is visible (but not necessarily intelligible) to all players; (iii) each player is controlled by a separate process; (iv) each team has 11 members; (v) players have limited stamina; (vi) actuators and sensors are noisy; (vii) dynamics and kinematics are modelled; and (viii) play occurs in *real time*: the agents must react to their sensory inputs at roughly the same speed as human or robotic soccer players. The simulator, acting as a server, provides a domain and supports users who wish to build their own agents (clients).

### 2.2 Layered Learning

Layered Learning is a Multiagent Learning paradigm designed to allow agents to learn to work together in a real-time, noisy environment in the presence of both teammates and adversaries. Layered Learning allows for a bottom-up definition of agent capabilities at different levels in a complete multiagent domain. Machine Learning opportunities are identified when hand-coding solutions are too complex to generate. Individual and collaborative behaviors in the presence of adversaries are organized, learned, and combined in a layered fashion (see Figure 1).

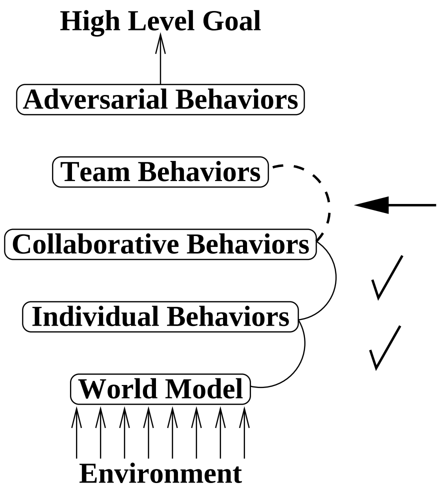


**Fig. 1.** An overview of the Layered Learning framework. It is designed for use in domains that are too complex to learn a mapping straight from sensors to actuators. We use a hierarchical, bottom-up approach. Two low-level behaviors have been previously learned. The work reported in this paper creates a team behavior that facilitates higher-level learning.

To date, two levels of learned behaviors have been implemented [13]. First, Soccer Server clients used a Neural Network (NN) to learn a low-level individual skill: how to intercept a moving ball. Then, using this learned skill, they learned a higher-level, more "social," skill: one that involves multiple players. The second skill, the ability to estimate the likelihood that a pass to a particular teammate will succeed, was learned using a Decision Tree (DT). The DT was trained using C4.5 [8] under the assumption that the player receiving the ball uses the trained NN when trying to receive the pass. This technique of incorporating one learned behavior as part of another is an important component of Layered Learning. As a further example, the output of the decision tree could be used as the input to a higher-level learning module, for instance a reinforcement learning module, to learn whether or not to pass, and to whom.

The successful combination of the learned NN and DT demonstrated the feasibility of the Layered Learning technique. However, the combined behavior was trained and tested in a limited, artificial situation which does not reflect the full range of game situations. In particular, a passer in a fixed position was trained to identify whether a particular teammate could successfully receive a pass. Both the teammate and several opponents were randomly placed within a restricted range. They then used the trained NN to try to receive the pass.

Although the trained DT was empirically successful in the limited situation, it was unclear whether it would generalize to the broader class of game situations. The work reported in this paper incorporates the same trained DT into a complete behavior using which players decide when to chase the ball, and after reaching the ball, what to do with it.

First, a player moves to the ball–using the NN–when it does not perceive any teammates who are likely to reach it more quickly. Then, using a pre-defined communication protocol, the player probes its teammates for possible pass receivers (collaborators). When a player is going to use the DT to estimate the likelihood of a pass succeeding, it alerts the teammate that the pass is coming, and the teammate, in turn, sends some data reflecting its view of the world back to the passer. The DT algorithm used is C4.5 [8], which automatically returns confidence factors along with classifications. These confidence factors are useful when incorporating the DT into a higher level behavior capable of controlling a client in game situations.

## 3 Using the Learned Behaviors

As described in Section 2, ML techniques have been studied in the Soccer Server in isolated situations. However, the resulting behaviors have not yet been tested in full game situations. In this paper, we examine the effectiveness in game situations of the DT learned in [13].

To our knowledge, this paper reports the first use of confidence factors from DTs for agent control. In particular, the confidence factors that are returned along with classifications can be used to differentiate precisely among several options.

### 3.1 Receiver Choice Functions

Recall that the DT estimates the likelihood that a pass to a specific player will succeed. Thus, for a client to use the DT in a game, several additional aspects of its behavior must be defined. First, the DT must be incorporated into a full *Receiver Choice Function* (RCF). We define the RCF to be the function that determines what the client should do *when it has possession of the ball*: when the ball is within kicking distance (2m). The input of an RCF is the client's perception of the current state of the world. This perceived state includes both the agent's latest sensory perception and remembered past positions of currently unseen objects [2]. The output of an RCF is an action from among the options *dribble, kick,* or *pass*, and a direction, either in terms of a player (i.e. towards teammate number 4) or in terms of a part of the field (i.e. towards the goal). Consequently, before using the DT, the RCF must choose a set of candidate receivers. Then, using the output of the DT for each of these receivers, the RCF can choose its receiver or else decide to dribble or kick the ball. Table 1 defines three RCFs, one of which uses the DT, and the others defined for the purposes of comparison.

---

1. Each player has a set of receivers that it considers, as indicated in Figure 2. The set of candidates is determined by the player's actual location on the field, rather than its assigned position.
2. Any potential receiver that is too close or too far away (arbitrarily chosen—but constant—bounds) is eliminated from consideration.
3. Any player that is out of position (because it was chasing the ball) is eliminated from consideration.
4. **IF** there is an opponent nearby (arbitrarily chosen—but constant—bound) **THEN** any potential receiver that cannot be passed to immediately (the passer would have to circle around the ball first) is eliminated from consideration.
5. **IF** one or more potential receivers remain **THEN** pass to the receiver as determined by the *Receiver Choice Function* (RCF):

   PRW (*Prefer Right Wing*): Use a fixed ordering on the options. Players in the center prefer passing to the right wing over the left.

   RAND (*Random*): Choose randomly among the options.

   DT (*Decision Tree*): Pass to the receiver to which the trained decision tree (see Section 2) assigns the highest **success confidence**. If no confidence is high enough, kick or dribble as indicated below.
6. **ELSE** (*No potential receivers remain*)
   - **IF** there is an opponent nearby, **THEN** kick the ball forward;
   - **ELSE** dribble the ball forward.

---

**Table 1.** Specification of the RCFs.

As indicated in Table 1, the set of candidate receivers is determined by the players' *positions*. Each player is assigned a particular position on the field, or an area to which it goes by default. The approximate locations of these positions are indicated by the locations of the players on the black team in Figure 2. The

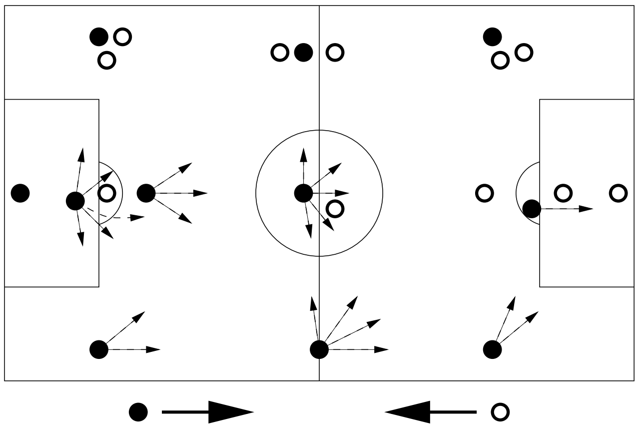

**Fig. 2.** Player positions used by the behaviors in this paper. The black team, moving from left to right, has a goalie, a sweeper, and one defender, midfielder, and forward on the left, center, and right of the field. The arrows emanating from the players indicate the positions to which each player considers passing when using the RCFs. The players on the left of the field (top of the diagram) consider symmetrical options to their counterparts on the right of the field. The goalie has the same options as the sweeper. The white team has the same positions as the black, except that it has no players on its left side of the field, but rather two in each position on its right.

formation used by all of the tested functions includes—from the back (left)—a goalie, a sweeper, three defenders, three midfielders, and three forwards. When a player is near its default position, it periodically announces its position to teammates; when a player leaves its position to chase the ball, it announces this fact and is no longer considered "in position" (see Table 1, Step 3). The arrows emanating from the players in Figure 2 indicate the *positions* to which each player considers passing. The clients determine which players are in which positions by listening to their teammates' announcements.

The RCFs defined and used by this paper are laid out in Table 1. As suggested by its name, the DT—*Decision Tree*—RCF uses the DT described in Section 2 to choose from among the candidate receivers. In particular, as long as one of the receivers' success confidences is positive, the DT RCF indicates that the passer should pass to the receiver with the highest success confidence, breaking ties randomly. If no receiver has a positive success confidence, the player with the ball should dribble or kick the ball forwards (towards the opponent goal or towards one of the forward corners). This use of the DT confidence factor is, to our knowledge, a novel approach to agent control. The RAND—*Random*—RCF is the same as the DT RCF except that it chooses randomly from among the candidate receivers.

The PRW—*Prefer Right Wing*—RCF uses a fixed ordering on the candidate receivers for each of the positions on the field. In general, defenders prefer to pass to the wings rather than forward, midfielders prefer to pass forward rather than sideways, and forwards tend to shoot. As indicated by the name, all players in the center of the field prefer passing to the right rather than passing to the left. The RCF simply returns the most preferable candidate receiver according to this fixed ordering. Again, if no receivers are eligible, the RCF returns "dribble" or "kick." This RCF was the initial hand-coded behavior for use in games.

### 3.2 Reasoning about action execution time

An important thing to notice in the RCF definition (Table 1) is that the clients can reason about how long they predict they have to act. In particular, if there is an opponent nearby, there is a danger of losing the ball before being able to pass or shoot it. In this situation, it is to the passer's advantage to get rid of the ball as quickly as possible.

This priority is manifested in the RCFs in two ways: (i) in Step 4 of Table 1, teammates to whom the client cannot pass immediately are eliminated from consideration; and (ii) in Step 6, the client kicks the ball away (or shoots) rather than dribbling. When a player is between the ball and the teammate to which it wants to pass, it must move out of the ball's path before passing. Since this action takes time, an opponent often has the opportunity to get to the ball before it can be successfully released. Thus, in Step 4, when there is an opponent *nearby* the RCFs only consider passing to players to whom the client can pass immediately. The concept of *nearby* could be the learned class of positions from which the opponent could steal the ball. For the purposes of this paper, "within 10m" is an empirically acceptable approximation. As mentioned above, this concept is not purely reactive: the positions of opponents that are outside an agent's field of view are remembered [2].

Similarly, the point of dribbling the ball (kicking the ball a small amount in a certain direction and staying with it) is to keep the ball for a little longer until a good pass becomes available or until the player is in a good position to shoot. However, if there is an opponent nearby, dribbling often allows the opponent time to get to the ball. In this situation, as indicated in Step 6 of Table 1, the player should kick the ball forward (or shoot) rather than dribbling.

The ability to reason about how much time is available for action is an important component of the RCFs and contributes significantly to their success in game situations (see Section 4).

### 3.3 Incorporating the RCF in a behavior

In Section 3.1, the method of using a DT as a part of an RCF is described in detail. However, the RCF is itself not a complete client behavior: it only applies when the ball is within kicking distance. This section situates the RCFs within a complete behavior that can then be used throughout the course of a game. The player's first priority is always to find the ball's location (only objects in front of

1. **IF** the client doesn't know where the ball is, **THEN** turn until finding it.
2. **IF** the ball is more than 10m away **AND** a teammate is closer to the ball than the client is, **THEN**:
   - **IF** the ball is coming towards the client, **THEN** watch the ball;
   - **ELSE** Move randomly near client position.
3. **ELSE**: (*client is the closest to the ball, or the ball is within 10m*)
   - **IF** The ball is too far away to kick ($> 2$m), **THEN** move to the ball, using the trained Neural Network when appropriate;
   - **ELSE** Pass, Dribble, or Kick the ball as indicated by the *Receiver Choice Function* (RCF).

**Table 2.** The complete behavior used by the clients in game situations.

the player are seen). If it doesn't know where the ball is, it turns until the ball is in view. When turning away from the ball, it remembers the ball's location for a short amount of time; however after about three seconds, if it hasn't seen the ball, it assumes that it no longer knows where the ball is [2].

Once the ball has been located, the client can execute its behavior. As described in Section 3.1, each player is assigned a particular position on the field. Unless chasing the ball, the client goes to its position, moving around randomly within a small range of the position. The player represents its position as $x, y$ coordinates on the field.

The client chases the ball whenever it thinks that it is the closest team-member to the ball. Notice that it may not *actually* be the closest player to the ball if some of its teammates are too far away to see, and if they have not announced their positions recently. However, if a player mistakenly thinks that it is the closest player, it will get part of the way to the ball, notice that another teammate is closer, and then turn back to its position. When the ball is within a certain small range (arbitrarily 10m), the client always goes towards the ball. When the ball is moving towards the client or when a teammate has indicated an intention to pass in its direction, the client watches the ball to see if either of the two above conditions is met. A player that was chasing the ball is predisposed to continue chasing the ball. Only if it finds that it should turn back persistently for several steps does it actually turn back. As required for use of the DT, every player is equipped with the trained Neural Network (see Section 2) which can be used to help intercept the ball.

Finally, every team member uses the same RCF. Whenever the ball is within kicking distance, the client calls its RCF to decide whether to dribble, kick, or pass, and to where. The behavior incorporating the RCFs is laid out in Table 2.

## 4 Experiments

In this section we present the results of empirically testing how the complete behavior performs when using the different RCF options. Since the behaviors differ only in their RCFs, we refer below to, for example, "the complete behavior with the DT RCF" simply as "the DT RCF." Also presented are empirical results verifying the advantage of reasoning about action-execution time.

In order to test the different RCFs, we created a team formation that emphasizes the advantage of passing to some teammates over others. When both teams use the standard formation (that of the black team in Figure 2), every player is covered by one opponent. However, this situation is an artificial artifact of having the same person program both teams. Ideally, the players would have the ability to move to open positions on the field. However at this point, such functionality represents future work (see Section 5). Instead, in order to reflect the fact that some players are typically more open than others, we tested the RCFs against the OPR—*Only Play Right*—formation which is illustrated by the white team in Figure 2. We also used the symmetrical OPL–*Only Play Left*—formation for testing. These behaviors are specified in Table 3.

- The opponent behaviors are exactly the same as the RAND behavior except that the players are assigned to different positions:

  **OPR** (*Only Play Right*): As illustrated by the white team in Figure 2, two players are at each position on the right side of the field, with no players on the left side of the field.

  **OPL** (*Only Play Left*): Same as above, except all the players are on the left side of the field.

**Table 3.** OPR and OPL behavior specification.

During testing, each run consists of 34 five-minute games between a pair of teams. We tabulate the cumulative score both in total goals and in games won (ties are not broken) as in Table 4. Graphs record the *difference* in cumulative goals scored (Figure 3) and games won (Figure 4) as the run progresses.

In order to test the effectiveness of the DT RCF, we compared its performance against the performance of the PRW and RAND RCFs when facing the same opponent: OPR. While the DT and RAND RCFs are symmetrical in their decision making, the PRW RCF gives preference to one side of the field and therefore has an advantage against the OPR strategy. Thus we also include the results of the PRW RCF when it faces the symmetrical opponent: OPL. From the table and the graphs, it is apparent that the DT RCF is an effective method of decision making in this domain.

In order to test the effectiveness of the reasoning about action-execution time, we compared the performance of the standard DT RCF against that of the same RCF with the assumption that there is *never* an opponent nearby:

| RCF (vs. OPR) | Games (W – L) | Overall Score |
|---|---|---|
| DT | 19 – 9 | 135 – 97 |
| PRW | 11 – 14 | 104 – 105 |
| PRW (vs. OPL) | 8 – 16 | 114 – 128 |
| RAND | 14 – 12 | 115 – 111 |

**Table 4.** Results are cumulative over 34 five-minute games: ties are not broken. Unless otherwise indicated, the opponent—whose score always appears second—uses the OPR formation.

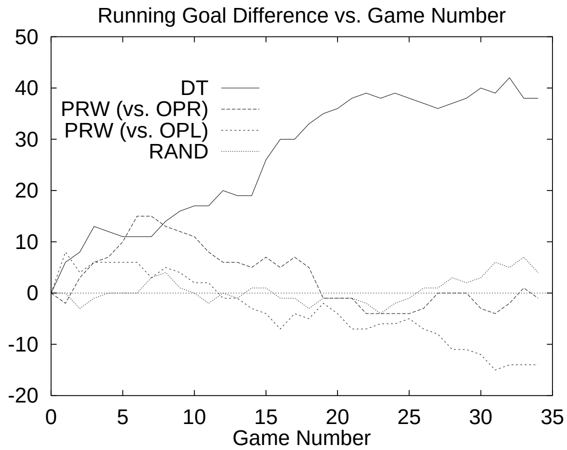


**Fig. 3.** The differences in cumulative goals as the runs progress.

even if there is, the RCF ignores it. This assumption affects Steps 4 and 6 of the RCF specification in Table 1. Both RCFs are played against the OPR behavior. As apparent from Table 5, the reasoning about action-execution time makes a significant difference.

| RCF (vs. OPR) | Games (W – L) | Overall Score |
|---|---|---|
| Standard DT | 19 – 9 | 135 – 97 |
| No-rush DT | 13 – 16 | 91 – 108 |

**Table 5.** No-rush DT is the same RCF as the standard DT except that there is no reasoning about action-execution time. The Standard DT RCF performs significantly better.

We expect that the DT RCF, including the reasoning about action-execution

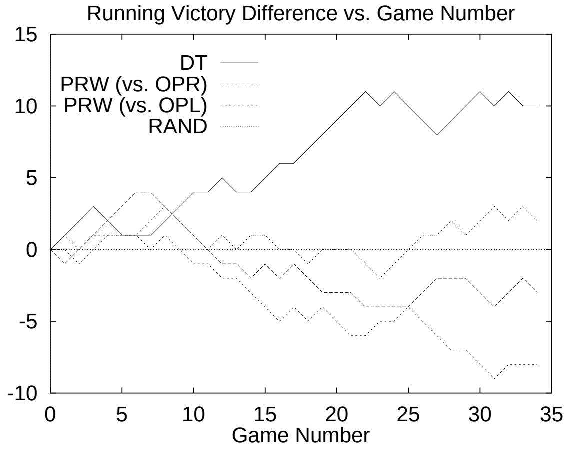


**Fig. 4.** The differences in cumulative games won as the runs progress.

time, will perform favorably against teams that cover our players unevenly so that the DT can find an open player to whom to pass. Indeed, it was used successfully as part of the CMUnited simulator team at RoboCup-97 [5] which was held at IJCAI-97. In a field of 29 teams, CMUnited made it to the semi-finals, before losing to the eventual champions[1].

## 5 Discussion and Conclusion

The experiments reported Section 4 indicate that the confidence factors provided by standard DT software can be used for effective agent control. Combined with some basic reasoning about the action-execution times of different options—necessitated by the real-time nature of this domain, the DT-based control function outperformed both random and hand-coded alternatives. Even though the DT was trained in a limited artificial situation, it was useful for agent control in a broader scenario.

Throughout this paper, the multiagent behaviors are tested against an opponent that leaves one side of the field free, while covering the other side heavily. This opponent simulates a situation in which the players without the ball make an effort to move to an open position on the field. Such collaborative reasoning has not yet been implemented in the Soccer Server. However, the fact that the DT is able to exploit open players indicates that reasoning about field positioning when a teammate has the ball would be a useful next step in the development of learned collaborative behaviors.

Along with more variable field positioning, there is still a great deal of future work to be done in this domain. First, one could build additional learned

[1] Full tournament results are available at http://www.robocup.org/RoboCup.

layers on top of the NN and DT layers described in Section 2. The behavior used in this paper *uses* the DT as a part of a hand-coded high-level multiagent behavior. However, several parameters are arbitrarily chosen. A behavior that *learns* how to map the classifications and confidence factors of the DT to passing/dribbling/shooting decisions may perform better. Second, on-line adversarial learning methods that can adapt to opponent behaviors during the course of a game may be more successful against a broad range of opponents than current methods.

Nevertheless, the incorporation of low-level learning modules into a full multiagent behavior that can be used in game situations is a significant advance towards intelligent multiagent behaviors in a complex real-time domain. Furthermore, the ability to reason about the amount of time available to act is essential in domains with continuously changing state. Finally, as DT confidence factors are effective tools in this domain, they are a new potentially useful tool for agent control in general. These contributions promise an exciting future for learning-based methods in real-time, adversarial, multiagent domains.

## Acknowledgements

This research is sponsored in part by the Defense Advanced Research Projects Agency (DARPA), and Rome Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-95-1-0018 and in part by the Department of the Navy, Office of Naval Research under contract number N00014-95-1-0591. Views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied, of the Air Force, of the Department of the Navy, Office of Naval Research or the United States Government.

## References

1. Minoru Asada, Eiji Uchibe, Shoichi Noda, Sukoya Tawaratsumida, and Koh Hosoda. Coordination of multiple behaviors acquired by vision-based reinforcement learning. In *Proc. of IEEE/RSJ/GI International Conference on Intelligent Robots and Systems 1994 (IROS '94)*, pages 917–924, 1994.
2. Mike Bowling, Peter Stone, and Manuela Veloso. Predictive memory for an inaccessible environment. In *Proceedings of the IROS-96 Workshop on RoboCup*, November 1996.
3. Andrew Garland and Richard Alterman. Multiagent learning through collective memory. In *Adaptation, Coevolution and Learning in Multiagent Systems: Papers from the 1996 AAAI Spring Symposium*, pages 33–38, Menlo Park,CA, March 1996. AAAI Press. AAAI Technical Report SS-96-01.
4. Jong-Hwan Kim, editor. *Proceedings of the Micro-Robot World Cup Soccer Tournament*, Taejon, Korea, November 1996.
5. Hiroaki Kitano, Yasuo Kuniyoshi, Itsuki Noda, Minoru Asada, Hitoshi Matsubara,

and Eiichi Osawa. RoboCup: A challenge problem for AI. *AI Magazine*, 18(1):73–85, Spring 1997.
6. Hitoshi Matsubara, Itsuki Noda, and Kazuo Hiraki. Learning of cooperative actions in multi-agent systems: a case study of pass play in soccer. In *Adaptation, Coevolution and Learning in Multiagent Systems: Papers from the 1996 AAAI Spring Symposium*, pages 63–67, Menlo Park,CA, March 1996. AAAI Press. AAAI Technical Report SS-96-01.
7. Itsuki Noda and Hitoshi Matsubara. Soccer server and researches on multi-agent systems. In *Proceedings of the IROS-96 Workshop on RoboCup*, November 1996.
8. J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.
9. Michael K. Sahota. Dynasim user guide. http://www.cs.ubc.ca/nest/lci/soccer, January 1996.
10. Michael K. Sahota, Alan K. Mackworth, Rod A. Barman, and Stewart J. Kingdon. Real-time control of soccer-playing robots using off-board vision: the dynamite testbed. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 3690–3663, 1995.
11. Randy Sargent, Bill Bailey, Carl Witty, and Anne Wright. Dynamic object capture using fast vision tracking. *AI Magazine*, 18(1):65–72, Spring 1997.
12. Peter Stone and Manuela Veloso. Beating a defender in robotic soccer: Memory-based learning of a continuous function. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 896–902, Cambridge, MA, 1996. MIT press.
13. Peter Stone and Manuela Veloso. A layered approach to learning client behaviors in the RoboCup soccer server. *To appear in Applied AI Journal*, 1998.
14. Peter Stone and Manuela Veloso. Towards collaborative and adversarial learning: A case study in robotic soccer. *in International Journal of Human-Computer Systems (IJHCS)*, 48, 1998.
15. Manuela Veloso, Peter Stone, Kwun Han, and Sorin Achim. CMUnited: A team of robotic soccer agents collaborating in an adversarial environment. In H. Kitano, editor, *RoboCup-97: The First Robot World Cup Soccer Games and Conferences.* Springer Verlag, Berlin, 1998, in this volume.