Multi-Robot Learning in a Cooperative Observation Task

Lynne E. Parker and Claude Touzet

Center for Engineering Science Advanced Research
Computer Science and Mathematics Division
Oak Ridge National Laboratory, Oak Ridge TN 37831-6355, USA
email: ParkerLE@ornl.gov

Abstract. An important need in multi-robot systems is the development of mechanisms that enable robot teams to autonomously generate cooperative behaviors. This paper first briefly presents the Cooperative Multi-robot Observation of Multiple Moving Targets (CMOMMT) application as a rich domain for studying the issues of multi-robot learning of new behaviors. We discuss the results of our hand-generated algorithm for CMOMMT, and then describe our research in generating multi-robot learning techniques for the CMOMMT application, comparing the results to the hand-generated solutions. Our results show that, while the learning approach performs better than random, naive approaches, much room still remains to match the results obtained from the hand-generated approach. The ultimate goal of this research is to develop techniques for multi-robot learning and adaptation that will generalize to cooperative robot applications in many domains, thus facilitating the practical use of multi-robot teams in a wide variety of real-world applications.

1 Introduction

Research in multi-robot cooperation has grown significantly in recent years. Before multi-robot teams will ever become widely used in practice, however, we believe that advances must be made in the development of mechanisms that enable the robot teams to autonomously generate cooperative behaviors and techniques. With the current state of the art, the implementation of cooperative behaviors on physical robot teams requires expert behavior programming and experimentation, followed by extensive tuning and revision of the cooperative control algorithms. It is unlikely that a significant real-world impact of cooperative robot teams will occur as long as the current level of effort is required to implement these systems.

Researchers have recognized that an approach with more potential for the development of cooperative control mechanisms is autonomous learning. Hence, much current work is ongoing in the field of multi-agent learning (e.g., [12]). This paper discusses our research in the learning of new behaviors in multi-robot teams. The types of applications that are typically studied for this area of multi-robot learning vary considerably in their characteristics. Some of the applications include air fleet control [9], predator/prey [2,3],

box pushing [4], foraging [6], and multi-robot soccer [10,5]. Particularly challenging domains for multi-robot learning are those tasks that are *inherently* cooperative. By this, we mean that the utility of the action of one robot is dependent upon the current actions of the other team members. Inherently cooperative tasks cannot be decomposed into independent subtasks to be solved by a distributed robot team. Instead, the success of the team throughout its execution is measured by the combined actions of the robot team, rather than the individual robot actions. This type of task is a particular challenge in multi-robot learning, due to the difficulty of assigning credit for the individual actions of the robot team members.

Of these previous application domains that have been studied in the context of multi-robot learning, only the multi-robot soccer domain addresses inherently cooperative tasks with more than two robots while also addressing the real-world complexities of embodied robotics, such as noisy and inaccurate sensors and effectors in a dynamic environment that is poorly modeled. To add to the field of challenging application domains for multi-robot learning, we have defined and have been studying a new application domain – the Cooperative Multi-robot Observation of Multiple Moving Targets (CMOMMT) – that is not only an inherently cooperative task, but, unlike the multi-robot soccer domain, is also a domain that must deal with issues of scalability to large numbers of robots. In [7] we presented the motivation for using this domain for multi-robot learning. In this paper, we briefly review this motivation, and then describe a hand-generated solution to this problem, along with the results we obtained with this approach. We then define a learning approach to enable robot teams to autonomously generate viable solutions to the CMOMMT application and compare the results to the hand-generated approach. The final section of the paper concludes with some summary remarks.

2 The CMOMMT Application

The application domain that we are studying for use as a multi-robot learning testbed is the problem we entitle *Cooperative Multi-robot Observation of Multiple Moving Targets* (CMOMMT). This problem is defined as follows. Given:

- $\mathcal{S}$: a two-dimensional, bounded, enclosed spatial region
- $\mathcal{V}$: a team of m robot vehicles, $v_i, i = 1, 2, ...m$, with 360^0 field of view observation sensors that are noisy and of limited range
- $\mathcal{O}(t)$: a set of n targets, $o_j(t)$, $j = 1, 2, ..., n$, such that target $o_j(t)$ is located within region $\mathcal{S}$ at time t

We say that a robot, v_i, is *observing* a target when the target is within v_i's sensing range. Define an $m \times n$ matrix $B(t)$, as follows:

$$B(t) = [b_{ij}(t)]_{m \times n} \text{ such that } b_{ij}(t) = \begin{cases} 1 \text{ if robot } v_i \text{ is } \textit{observing} \text{ target} \\ \quad o_j(t) \text{ in } \mathcal{S} \text{ at time } t \\ 0 \text{ otherwise} \end{cases}$$

Then, the goal is to develop an algorithm, which we call *A-CMOMMT*, that maximizes the following metric A:

$$A = \sum_{t=1}^{T} \sum_{j=1}^{n} \frac{g(B(t), j)}{T}$$

where:

$$g(B(t), j) = \begin{cases} 1 \text{ if there exists an } i \text{ such that } b_{ij}(t) = 1 \\ 0 \text{ otherwise} \end{cases}$$

That is, the goal of the robots is to maximize the average number of targets in $\mathcal{S}$ that are being observed by at least one robot throughout the mission that is of length T time units. Additionally, we define *sensor_coverage*(v_i) as the region visible to robot v_i's observation sensors, for $v_i \in \mathcal{V}$. Then we assume that, in general, the maximum region covered by the observation sensors of the robot team is much less than the total region to be observed. That is, $\bigcup_{v_i \in \mathcal{V}} \textit{sensor\_coverage}(v_i) \ll \mathcal{S}$. This implies that fixed robot sensing locations or sensing paths will not be adequate in general, and instead, the robots must move dynamically as targets appear in order to maintain their target observations and to maximize the coverage.

The CMOMMT application is an excellent domain for embodied multi-robot learning and adaptation. CMOMMT offers a rich testbed for research in multi-robot cooperation, learning, and adaptation because it is an inherently cooperative task. In addition, many variations on the dynamic, distributed sensory coverage problem are possible, making the CMOMMT problem arbitrarily more difficult. For example, the relative numbers and speeds of the robots and the targets to be tracked can vary, the availability of inter-robot communication can vary, the robots can differ in their sensing and movement capabilities, the terrain may be either enclosed or have entrances that allow objects to enter and exit the area of interest, and so forth. Many other subproblems can also be addressed, including the physical tracking of targets (e.g. using vision, sonar, IR, or laser range), prediction of target movements, multi-sensor fusion, and so forth.

3 A Hand-Generated Solution to CMOMMT

We have developed a hand-generated solution to the CMOMMT problem that performs well when compared to various control approaches. This solution

has been implemented on both physical and simulated robots to demonstrate its effectiveness. The hand-generated solution, which we call *A-CMOMMT*, is described briefly as follows. Robots use weighted local force vectors that attract them to nearby targets and repel them from nearby robots. The weights are computed in real time by a higher-level reasoning system in each robot, and are based on the relative locations of the nearby robots and targets. The weights are aimed at generating an improved collective behavior across robots when utilized by all robot team members.

The local force vectors are calculated as follows. The magnitude of the force vector attraction of robot v_l relative to target o_k, denoted $\mid \mathbf{f_{lk}} \mid$, for parameters $0 < do_1 < do_2 < do_3$, is:

$$\mid \mathbf{f_{lk}} \mid = \begin{cases} \frac{-1}{do_1} & \text{for } d(v_l, o_k) \leq do_1 \\ \frac{1}{do_2 - do_1} & \text{for } do_1 < d(v_l, o_k) \leq do_2 \\ \frac{-do_2}{do_3 - do_2} & \text{for } do_2 < d(v_l, o_k) \leq do_3 \\ 0 & \text{otherwise} \end{cases}$$

where $d(a, b)$ returns the distance between two entities (i.e., robots and/or targets). The magnitude of the force vector repulsion of robot v_l relative to robot v_i, denoted $\mid \mathbf{g_{li}} \mid$, for parameters $0 < dr_1 < dr_2$, is:

$$\mid \mathbf{g_{li}} \mid = \begin{cases} -1 & \text{for } d(v_l, v_i) \leq dr_1 \\ \frac{1}{dr_2 - dr_1} & \text{for } dr_1 < d(v_l, v_i) \leq dr_2 \\ 0 & \text{otherwise} \end{cases}$$

Determining the proper setting of the parameters do_1, do_2, do_3, dr_1, and dr_2 is one approach to solving the CMOMMT multi-robot learning task using an *a priori* model-based technique.

Using only local force vectors for this problem neglects higher-level information that may be used to improve the team performance. Thus, the hand-generated approach enhances the control approach by including higher-level control to weight the contributions of each target's force field on the total computed field. This higher-level knowledge can express any information or heuristics that are known to result in more effective global control when used by each robot team member locally. The hand-generated approach expresses this higher-level knowledge in the form of a weight, w_{lk}, that reduces robot r_l's attraction to a nearby target o_k if that target is within the field of view of another nearby robot. Using these weights helps reduce the overlap of robot sensory areas toward the goal of minimizing the likelihood of a target escaping detection.

The higher-level weight information is combined with the local force vectors to generate the commanded direction of robot movement. This direction of movement for robot v_l is given by: $\sum_{k=1}^{n} w_{lk} \mathbf{f_{lk}} + \sum_{i=1, i \neq l}^{m} \mathbf{g_{li}}$, where $\mathbf{f_{lk}}$ is the force vector attributed to target o_k by robot v_l and $\mathbf{g_{li}}$ is the force vector attributed to robot v_i by robot v_l. To generate an (x, y) coordinate indicating the desired location of the robot corresponding to the resultant

force vector, we scale the resultant force vector based upon the size of the robot. The robot's speed and steering commands are then computed to move the robot in the direction of that desired location.

4 Results from Hand-Generated Solution

Figure 1 shows two of the simulation runs of the hand-generated algorithm (out of over 1,000,000 simulation test runs). Figure 2 shows snapshots of two of the physical robot experiments (out of over 800) in which the the robots perform the task either with no obstacles in the work area or with randomly distributed obstacles.

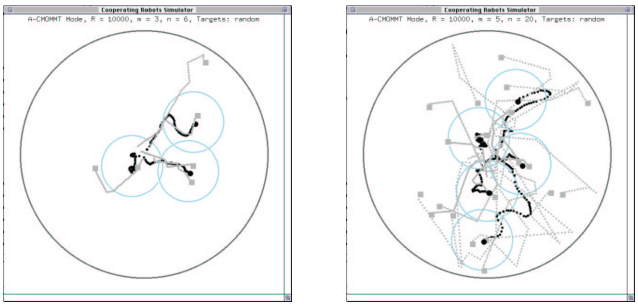

Fig. 1. Simulation results of three robots and six targets (first image), and five robots and twenty targets (second image), with the robots using the hand-generated solution to CMOMMT, and the targets moving randomly.

The results of the hand-generated approach to CMOMMT vary depending upon a number of factors, including the relative numbers of robots and targets, the size of the work area, the motions of the targets (i.e., whether random or evasive), and the setting of the weights. In general, the *A-CMOMMT* algorithm performed best for a ratio of targets to robots greater than 1/2. We compared the hand-generated *A-CMOMMT* approach with a non-weighted local force vector approach, as well as two control cases in which robots either maintained fixed positions or are moved randomly. Figure 3 gives a typical set of these comparative results.

5 Learning in the CMOMMT Application

We have studied the CMOMMT problem from a learning perspective without the assumption of an *a priori* model. This approach uses a combination of reinforcement learning, lazy learning, and a Pessimistic algorithm able to compute for each team member a lower bound on the utility of executing an action in a given situation. The challenges in this multi-robot learning

Fig. 2. Robot team executing hand-generated solution to CMOMMT. The first photo shows robots operating in an area with no obstacles. The second photo shows the robots amidst randomly distributed obstacles.

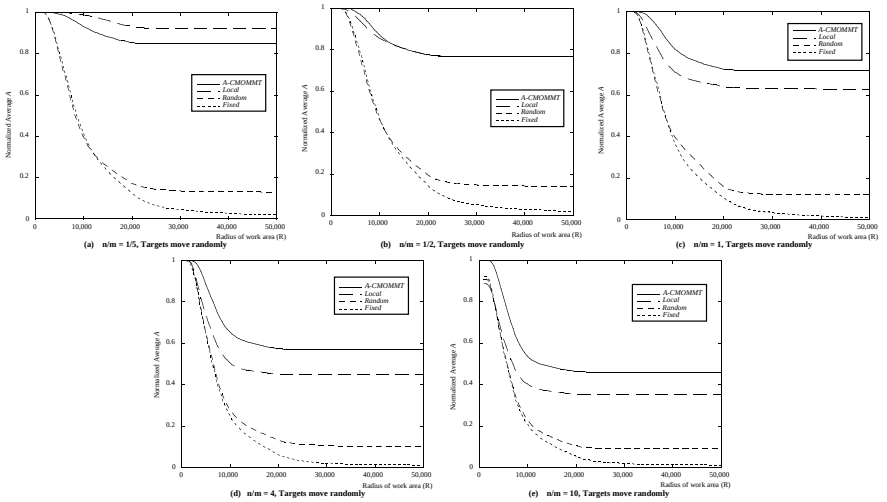

Fig. 3. Simulation results of four distributed approaches to cooperative observation, for random/linear target movements, for various ratios of number of targets (n) to number of robots (m).

problem include a very large search space, the need for communication or awareness of robot team members, and the difficulty of assigning credit in an inherently cooperative problem.

In this learning approach, lazy learning [1] is used to enable robot team members to build a memory of situation-action pairs through random exploration of the CMOMMT problem. A reinforcement function gives the utility of a given situation. The pessimistic algorithm for each robot then uses the utility values to select the action that maximizes the lower bound on utility. The resulting algorithm is able to perform considerably better than a random action policy, although it is still significantly inferior to the hand-generated algorithm described in the previous section. However, even with a performance less than that of the hand-generated solution, this approach makes an im-

portant contribution because it does not assume the existence of a model (as is the case in the Partially Observable Markov Decision Process (POMDP) domain), the existence of local indicators that help individual robots perform their tasks, nor the use of symbolic representations. The following subsections describe this approach and its results in more detail.

5.1 Lazy learning and Q-learning

Lazy learning [1] – also called instance-based learning – promotes the principle of delaying the use of the gathered information until the necessity arises (see Fig. 4). The same pool of information (i.e., memory) is used for different behavior syntheses. The lazy memory provides a good way of reducing the duration of any robotic learning application. In the context of reinforcement learning, lazy learning provides an instantaneous set of situation-action pairs (after the initial and unique sampling phase). Lazy learning samples the situation-action space according to a random action selection policy, storing the succession of events in memory and, when needed, probes the memory for the best action. The exploration phase is performed only once. By storing situation-action pairs, a lazy memory builds a model of the situation transition function.

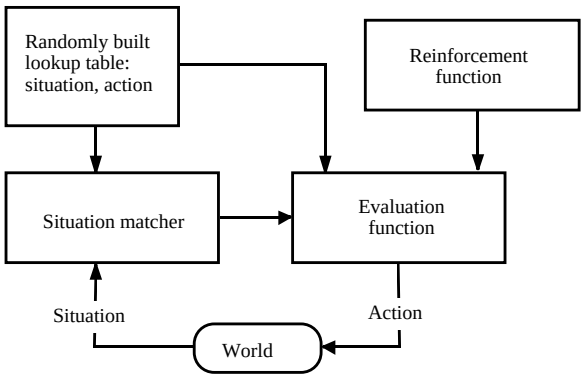

Fig. 4. Lazy learning: randomly sampled situation-action pairs in the lookup table are used by the situation matcher to select the action to execute in the current situation. The reinforcement function qualifies the actions proposed, helping to select the best one.

In order to express a behavior, the memory must be probed. To do this probing, we use a modified version of the technique proposed in [8]. In [8] the objective is to provide a method for predicting the rewards for state-action pairs without explicitly generating them. For the current real world situation, a situation matcher locates all the states in the memory that are within a

given distance. If the situation matcher has failed to find any nearby situations, the action comparator selects an action at random. Otherwise, the action comparator examines the expected rewards associated with each of these situations and selects the action with the highest expected reward. This action is then executed, resulting in a new situation. There is a fixed probability (0.3) of generating a random action regardless of the outcome of the situation matcher. New situation-action pairs are added to the memory, along with a Q-value computed in the classical way [11]. Among similar situation-action pairs in the memory, an update of the stored Q-values is made. However, there is a limit to the generality of this lazy memory because the Q-values associated with the situation-action pairs only apply for a particular behavior. With the desire of reducing the learning time as much as possible, as well as preserving the generality of the lazy memory, we modified the algorithm as follows: (1) the situation matcher always proposes the set of nearest situations – no maximum distance is involved, (2) there is no random selection of actions by the action comparator, and (3) the Q-values are not stored with the situation-action pairs, but are computed dynamically as the need arises.

5.2 The Pessimistic Algorithm

We define a Pessimistic Algorithm for the selection of the best action to execute for a given robot in its current local situation as follows: find the lower bounds of the utility value associated with the various potential actions that may be conducted in the current situation, then choose the action with the greatest utility. A lower bound is defined as the lowest utility value associated with a set of similar situations.

The idea behind the Pessimistic Algorithm is that a local robot situation is an incomplete observation of the true state of the system. Thus, instead of trying to solve the observation problem by completing the observation (usual POMDP approach), we are only interested in ranking the utility of the actions. If we use a unique instance of the memory to obtain the utility of the situation, then chances are that the utility attributed to this local situation is due in fact to other robot's actions. This probability decreases proportionally with the number of similar situations that are taken into account. If a large number of situations are considered, then there must be at least one for which the reward directly depends on the local situation. By taking the minimum utility value of the set of similar situations, we are guaranteed that, if the value is null, then the situation achieved does not imply loosing target(s).

The Pessimistic Algorithm is then given as follows:

- `Let M be the memory, a lookup table of situation-action pairs gathered during an exploration phase --` $M = [(s(1), a(1)), ..., (s(t), a(t)), (s(t+1), a(t+1)), ...]$.
- `Let` sit `be the current situation.`

- Find $S(sit)$, the set of n situations of M similar to sit.
- Let $S_{follow}(sit)$ be the set of the situations that directly follows each situation of $S(sit)$.
- Compute the lower bound (LB) of the utility value (U) associated with each situation $s(k) \in S_{follow}(sit)$:
  - $LB(s(k)) = \min(U(s(m)))$, for $s(m) \in S(s(k))$, the set of situations similar to $s(k)$.
- Execute the action that should take the robot to the new situation s^*: $s^* = \max(LB(s))$ and $s \in S_{follow}(sit)$.

The utility U associated with a given situation can be computed in many ways. It can be the exact value of the reinforcement function for this particular situation-action pair, or it can be a more elaborate variable. For example, in our experience we store the situation-action pairs, plus the number of targets under observation in the lookup table (M). However, the value that is used as utility is +1 if one or more targets have been acquired compared to the previous situation, -1 if one or more targets have been lost, or 0 otherwise. An exact Q value requires running the Q-learning algorithm with the samples stored in the memory.

5.3 Results of Learning Approach

We studied the efficiency of the Pessimistic Algorithm by comparing the performance of a team of robots with a purely random action selection policy, a user-defined non-cooperative policy and *A-CMOMMT*. In these experiments, each robot situation is a vector of two times 16 components. The first 16 components code the position and orientation of the targets. It simulates a ring of 16 sensors uniformly distributed around the robot body. Each sensor measures the distance to the nearest target. The sensor position around the body gives the orientation. The second ring of 16 components code in the same manner the position and orientation of neighboring robots. The maximum range for a target or a robot to be seen is 1, for an arena radius of 5. The actions of each robot are rotation and forward movement. The measure of performance is the mean observation time of all targets.

Figure 5 shows the performance of a Pessimistic lazy Q-learning policy versus the size of the lazy memory, from 100 to 900 situation-action pairs. Each point is the average of 10 experiments. The lazy memories are obtained through an initial exploration involving from 15 to 25 targets and a single robot. During the sampling, the targets are fixed and the robot's policy is random action selection (with 5% chance of direction and orientation changes). The reinforcement function returns +1 if the total number of targets under observation increases, -1 if this number decreases, or 0 otherwise.

As we see there is an important performance gain associated with the Pessimistic lazy Q-learning over a purely random selection policy. This clearly

demonstrates the importance of lazy Q-learning as a learning technique. Even more interestingly, lazy Q-learning performs much better than the user-defined non-cooperative policy (*Local*). It is important to note that neither policy is aware of the existence of the other robots. Both policies use the same sensory information – i.e., the distance and orientation of nearby targets. It is our opinion that the variation of performance is due to the fact that the lazy Q-learned behavior is somewhat less rigid than the user-defined policy. A lazy Q-learning guided robot will follow a target further than it could be, and, in doing so, will exhibit an erratic path, moving from one side of the target to another, back and forth without losing the target. In doing so, the surface under observation per unit of time is larger than the covered surface by the more rigid center-of-gravity-oriented robot. On the other hand, because it does not take into account the neighboring robots, it is easy to understand why the lazy Q-learned behavior performance cannot reach the level of the *A-CMOMMT* performance.

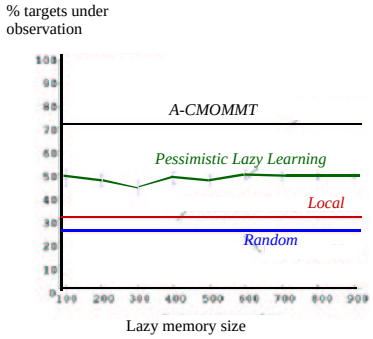

Fig. 5. Performances of the Pessimistic lazy Q-learning approach compared to a random action selection policy, a user-defined non-cooperative policy and the hand-generated solution *A-CMOMMT*, for 10 robots and 10 randomly moving targets. The results are the mean of 10 different experiments per point for lazy learning policy, and 100 experiments for the other 3 policies. Each experiment duration is 1000 iterations.

6 Conclusions

In this paper, we have proposed that the Cooperative Multi-robot Observation of Multiple Moving Targets (CMOMMT) application domain provides a rich testbed for learning and adaptation in multi-robot cooperative teams. We have described the need for learning and adaptation in multi-robot teams, and have defined the CMOMMT application, along with the characteristics that make it an interesting testbed for learning and adaptation. We reported on a hand-generated solution to the CMOMMT problem and discussed how

the results from the implementation of this solution reveal the need for learning and adaptation in this domain. We discussed our work that uses the CMOMMT problem as a learning domain. The ultimate objective is to develop learning techniques using the CMOMMT domain that will generalize to other real-world domains, and will thus help realize the ultimate goal of enabling the widespread, practical use of multi-robot teams.

Acknowledgments

This research is sponsored by the Engineering Research Program of the Office of Basic Energy Sciences, U. S. Department of Energy. Accordingly, the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U. S. Government purposes. Oak Ridge National Laboratory is managed by UT-Battelle, LLC for the U.S. Dept. of Energy under contract DE-AC05-00OR22725.

References

1. D. Aha, editor. *Lazy Learning.* Kluwer Academic Publishers, 1997.
2. M. Benda, V. Jagannathan, and R. Dodhiawalla. On optimal cooperation of knowledge sources. Technical Report BCS-G2010-28, Boeing AI Center, August 1985.
3. Thomas Haynes and Sandip Sen. Evolving behavioral strategies in predators and prey. In Gerard Weiss and Sandip Sen, editors, *Adaptation and Learning in Multi-Agent Systems*, pages 113–126. Springer, 1986.
4. S. Mahadevan and J. Connell. Automatic programming of behavior-based robots using reinforcement learning. In *Proceedings of AAAI-91*, pages 8–14, 1991.
5. S. Marsella, J. Adibi, Y. Al-Onaizan, G. Kaminka, I. Muslea, and M. Tambe. On being a teammate: Experiences acquired in the design of RoboCup teams. In O. Etzioni, J. Muller, and J. Bradshaw, editors, *Proceedings of the Third Annual Conference on Autonomous Agents*, pages 221–227, 1999.
6. Maja Mataric. *Interaction and Intelligent Behavior.* PhD thesis, Massachusetts Institute of Technology, 1994.
7. L. E. Parker. A case study for life-long learning and adaptation in cooperative robot teams. In *Proceedings of SPIE Sensor Fusion and Decentralized Control in Robotic Systems II*, volume 3839, pages 92–101, 1999.
8. J.W. Sheppard and S.L. Salzberg. A teaching strategy for memory-based control. In D. Aha, editor, *Lazy Learning*, pages 343–370. Kluwer Academic Publishers, 1997.
9. Randall Steeb, Stephanie Cammarata, Frederick Hayes-Roth, Perry Thorndyke, and Robert Wesson. Distributed intelligence for air fleet control. Technical Report R-2728-AFPA, Rand Corp., 1981.
10. P. Stone and M. Veloso. A layered approach to learning client behaviors in the robocup soccer server. *Applied Artificial Intelligence*, 12:165–188, 1998.
11. C. Watkins. *Learning from Delayed Rewards.* PhD thesis, King's College, Cambridge, 1989.
12. Gerhard Weiss and Sandip Sen, editors. *Adaption and Learning in Multi-Agent Systems.* Springer, 1996.