Reinforcement Learning in the Multi-Robot Domain

Maja J Matarić
Volen Center for Complex Systems
Computer Science Department
Brandeis University
Waltham, MA 02254
tel: (617) 736-2708 fax: (617) 736-2741
maja@cs.brandeis.edu

Abstract

This paper describes a formulation of reinforcement learning that enables learning in noisy, dynamic environemnts such as in the complex concurrent multi-robot learning domain. The methodology involves minimizing the learning space through the use behaviors and conditions, and dealing with the credit assignment problem through shaped reinforcement in the form of heterogeneous reinforcement functions and progress estimators. We experimentally validate the approach on a group of four mobile robots learning a foraging task.

1 Introduction

Developing effective methods for real-time learning has been an on-going challenge in autonomous agent research and is being explored in the mobile robot domain. In the last decade, reinforcement learning (RL), a class of approaches in which the agent learns based on reward and punishment it receives from the environment, has become the methodology of choice for learning in a variety of domains, including robotics.

In this paper we describe a formulation of reinforcement learning that minimizes the state space through the use of *behaviors* and their associated *conditions*, and shapes reinforcement with *heterogeneous reinforcement functions* and *progress estimators* that take advantage of implicit domain knowledge in order to both enable and accelerate learning. Our approach was validated on a group of four mobile robots given the task of automatically synthesizing an effective strategy for collective foraging, i.e., finding pucks and delivering them to a specified area. The main challenges of the validation domain included sensing and acting uncertainty, inconsistent reinforcement, the lack of a world model, and the need to deal with concurrent goals.

This paper is organized as follows. Section 2 introduces the challenges for learning in multi-robot domains. Section 3 describes our formulation of the learning problem in response to those challenges. Section 4 describes the experimental environment. Section 5 describes the learning task, Section 6 the learning and control algorithms, and Section 7 gives the experimental results and their evaluation. Section 8 overviews related work and Section 9 concludes the paper.

2 Challenges

This work is inspired by the challenges of learning in noisy, dynamic environments, like the ones mobile robots exist in. We are particularly interested in the complex case of concurrent multi-robot learning and will use it in this paper as an instance of the broader learning problem and the experimental domain of choice.

We classify the many challenges of learning in dynamic environments into two main categories: 1) managing complexity in the size of the state space and 2) dealing with structuring and assigning reinforcement. In this section we discuss each in general, and in the next we describe our specific approaches to both.

2.1 State and Action Representation

The world of a mobile robot is partially observable through its noisy sensors which provide the input state consisting of a combination of discrete and continuous sensory inputs, ranging from infra red sensors, to (x,y) positions, to wheel velocities. While mobile robots are largely considered to be sensory-impoverished for the purposes of control, their input state is typically very large for the purposes of learning. The problem is made worse in the multi-robot case where the state also includes information about other robots in the environment. Consequently, some form of input generalization or state clustering becomes necessary for most non-trivially sized learning problems. Various statistical approaches to input state clustering have been explored, and connectionist methods for state generalization have been successfully used, and a small subset of the work was applied to mobile robots (Mahadevan & Connell 1990, Asada, Uchibe, Noda, Tawaratsumida & Hosoda 1994). In this paper we describe a method for state clustering through the use of behaviors and conditions.

Regardless of the state representation used, reinforcement learning algorithms rely on their dynamic programming roots for clean theoretical convergence properties (Watkins & Dayan 1992, Barto, Bradtke & Singh 1993) which in turn require large numbers of learning trials that are prohibitive in physical robot domains. Model-based approaches (Atkeson, Aboaf, McIntyre & Reinkensmeyer 1988, Atkeson 1989, Moore 1992, Schaal & Atkeson 1994) have been effective at cutting down the number of learning trials, but in discrete mobile robot domains such models may be very difficult to obtain.

From the perspective of a learning mobile robot, in particular one situated in a dynamic environment, state transitions are largely externally induced and asynchronous, and their causes cannot always be sensed, and the results of actions are often inconsistent. Consequently, building a predictive model of such a world and deriving realistic state transition probabilities can be very slow, given that it must be obtained empirically, whether it is done by the robot through a learning process, or statistically by the designer. As a result, for certain problems in such domains, it is much more efficient to learn a policy for some fixed set of goals of the robot(s). In this paper we discuss and demonstrate an example of just such a policy-learning model-free system.

2.2 Reinforcement

A typical mobile robot environment does not provide a direct source of immediate reinforcement so the problem of assigning appropriate delayed credit and blame is endemic. The situation is worse in multi-robot systems since a given event that induces reinforcement may be due to any of a robot's past actions which could have been either i) attempts to reach the goal or ii) reactions to the presence and/or behavior of another robot. Consequently, the interaction between the robot and its complex environment produces a confounding credit assignment problem. In this paper we discuss how this problem can be addressed through the use of shaped reinforcement in the form of heterogeneous reward functions and progress estimators.

Reinforcement learning tasks are typically implemented using a monolithic reinforcement function that enables the learner to eventually acquire the optimal policy. Constructing such a reinforcement function can be a complicated task in the mobile robot domain, since the environment may provide some immediate rewards, some intermittent reinforcement, and a handful or fewer of very delayed signals associated with reaching one or more high-level goals. In this paper we discuss how such *heterogeneous* multi-modal reinforcement can be combined and utilized to accelerate the learning process.

3 Formulating the Learning Problem

As discussed above, two main challenges arise when applying RL to a multi-robot domain: 1) a prohibitively large state space, and 2) credit assignment. This section describes how we addressed each one.

3.1 Behaviors and Conditions

Our work can be classified as *behavior-based* since it involves the use of behaviors as the basic representation level for control and learning. *Behaviors* are goal-driven control laws that achieve and/or maintain particular goals (Matarić 1994*a*). Behaviors with achievement goals, such as *homing*, terminate when the goal is reached, while behaviors with maintenance goals, such as *wall-following*, continue execution as long as their conditions are satisfied. Both kinds of behaviors can be externally terminated. They are designed (or learned) so as to provide the desired outputs while abstracting away the low level details of control. Well-designed behaviors utilize the dynamics of the system and its interaction with the world in order to achieve robust and repeatable performance (Matarić 1994*a*).

Behaviors are triggered by *conditions*, predicates on sensor readings that map into a proper subset of the state space. Each condition is defined as the part of the state that is necessary and sufficient for activating a particular behavior. For instance, the necessary and sufficient conditions for picking up a puck are that a puck is in the robot's gripper. The truth value of a condition determines when a behavior can be executed and when it should be terminated, thus providing a set of events for a learner's control algorithm. In general, conditions for execution of any behavior are given by the formal specification of that behavior (Matarić 1994*a*). Thus, if a fixed behavior set is used by a learning robot, its condition set can be computed off-line. This set is typically much smaller than the robot's complete state space.

Behaviors abstract away the details of the low-level controllers driving the robot, while conditions abstract away the details of the agent's state space. Combined, the two define the learning space at a higher level of description.

3.2 Shaped Reinforcement

Reformulating states and actions into conditions and behaviors effectively reduces the learning space. This is of particular importance for making learning possible in the complex multi-robot domain. The next step is to make learning efficient, by using appropriate reinforcement.

Simplifying and minimizing reinforcement, as practiced by some early RL algorithms (Sutton 1988), diminishes programmer bias, but also greatly handicaps and in some domains completely debilitates the learner. We propose *shaped reinforcement* as a means of taking advantage of as much information as is available to the robot at any point. Shaping is based on principled embedding of domain knowledge in order to convert intermittent feedback into a more continuous error signal using two types of reinforcement: heterogeneous reward functions and progress estimators.

Heterogeneous reward functions combine multi-modal feedback from the available external (i.e., sensory) and internal (i.e., state) modalities. The combination is a weighted sum of inputs from the individual event-driven functions. The more subgoals the system recognizes, the more frequently reinforcement can be applied, and the faster the learner can converge. In our representation of the system, each behavior provides a goal whose achievement can be detected as an event, and can also be directly translated into a reinforcement signal. The inputs from these signals contribute to the heterogeneous reward function.

These reward functions, much like most typically used in reinforcement learning, deliver reinforcement in response to events, i.e. between behaviors. In contrast, *progress estimators* are evaluation metrics relative to a current goal that the robot can estimate during the execution of a behavior. Progress estimators diminish brittleness of the learning algorithm in the following three ways:

• They decrease sensitivity to noise by strengthening appropriate condition-behavior correlations. Noise-induced events are not consistently supported by progress estimator credit, and thus have less impact on the learner. The domain knowledge behind progress estimators provides a continuous source of reinforcement to counter intermittent and potentially incorrectly assigned credit.

• They encourage exploration by using lack of progress to terminate behaviors principally. A robot may have no impetus for terminating a behavior and attempting alternatives, since any behavior may eventually produce a reward, and can last arbitrarily long until some event terminates it. Progress estimators provide a non-arbitrary method for behavior termination.

• They decrease fortuitous rewards that may be received for an inappropriate behavior that happened to achieve the desired goal in the particular situation, but would not have that effect in general. Progress estimators achieve this effect incrementally, because behaviors have some measurable duration that allows the estimators to contribute reinforcement, and thus give less credit to intermittent and fortuitous success.

The learning algorithm, described in section 6, combines the credit from both sources, i.e., from heterogeneous reward functions and from progress estimators.

4 The Experimental Environment

The learning experiments were performed on a group of four fully autonomous IS Robotics R2 mobile robots with on-board power and sensing. Each robot has a differentially steerable wheeled base and a gripper for grasping and lifting objects. The robots' sensory capabilities include piezo-electric bump sensors for detecting contact-collisions and monitoring the grasping force on the gripper, and a set of infra-red (IR) sensors for obstacle avoidance and grasping (Figure 1).

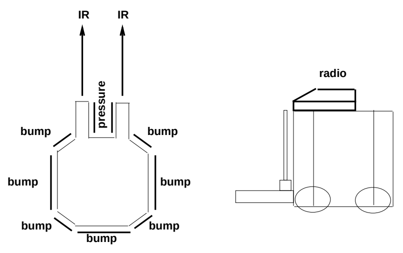

Figure 1: The robot's sensory capabilities include piezo-electric bump and gripper sensors used to detect collisions and to grasp pucks, infra-red sensors for collision avoidance, and a radio transmitter for absolute positioning and message passing.

The robots are also equipped with radio transceivers, used for determining absolute position and for inter-robot communication. Position information is obtained by triangulating the distance computed from synchronized ultrasound pulses from two fixed beacons. Inter-robot communication consists of broadcasting 6-byte messages at the rate of 1 Hz. In the experiments described here, the radios are used to determine the presence of other nearby robots. The robots are programmed in the Behavior Language (Brooks 1990) and tested in the workspace shown in Figure 2.

5 The Learning Task

The learning task consists of finding a mapping from conditions to behaviors into the most effective policy for group foraging. Individually, each robot learns to select the behavior with the highest value for each condition, in order to find and take home the most pucks. Foraging was chosen because it is a complex and biologically inspired task, because it serves as a canonical abstraction of a variety of real-world applications (such as demining and toxic waste clean up), and because our previous group behavior work (Matarić 1992*b*, Matarić 1993) provided the basic behavior repertoire from which to learn behavior selection. That repertoire, given to the robots *a priori*, consisted of the following fixed behavior set:

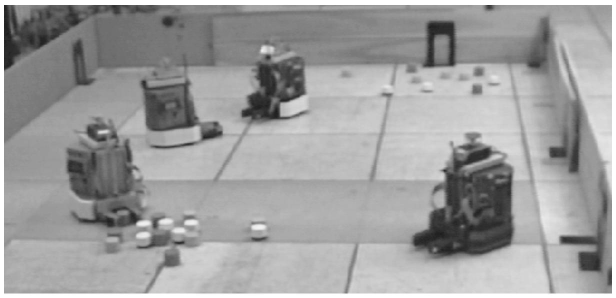

Figure 2: A typical environment state during the course of a learning experiment. The home region is in the upper right corner of the workspace.

- *safe-wandering* - keeps the robot moving about without colliding with any objects, including other robots.
- *dispersion* - achieves and maintains a minimum distance between all robots within sensing range of each other.
- *resting* - intended as a part of a regular recharging cycle, keeps the robot parked at home for a fixed period of time (Figure 3).
- *homing* - takes the robot to a particular location.

The robots were given the task of learning the appropriate conditions for triggering each of the above behaviors. By considering only the space of conditions necessary and sufficient for triggering the behavior set, the state space is reduced to the power set of the following clustered condition predicates:

- *have-puck?* - based on the state of the gripper sensors.
- *at-home?* - based on the robot's (x, y) position.
- *near-intruder?* - based on the distance between the robot and its nearest neighbor.
- *night-time?* - based on the state of an internal clock that imposes a cyclic schedule of "day-time" and shorter "night-time" periods.

Utility behaviors for *grasping* and *dropping* were also included in the robots' capabilities, but their conditions were built-in, so they do not constitute a part of the learning space. As soon as a robot detects a puck in the gripper, it grasps it. Similarly, as soon as a robot reaches the home region, it drops the puck if it is carrying one. Finally, whenever a robot is too near an obstacle, it avoids. The rationale for building in those particular behaviors and learning the rest was based on the fact that they are easily programmed and can be potentially damaging to the robot to learn. Building-in the above behaviors does simplify our learning task; however, we will demonstrate that learning the remaining behaviors in this robot domain is sufficiently challenging to illustrate our approach.

As described, the foraging task may appear quite simple, since its learning space has been appropriately minimized. In theory, an agent should be able to quickly search it and learn the optimal policy. In practice, however, such quick and uniform exploration is not possible. Even this relatively small learning space presents a challenge for reinforcement learning in this highly dynamic, uncertain environment.

6 The Learning Algorithm

The algorithm learns a value function that maps conditions c to behaviors b. This function is used by the behavior selection algorithm to choose the most "appropriate" behavior for each condition the robot finds itself in. The learning system is table-based and maintains a matrix $A(c,b)$ whose entries reflect a normalized sum of the reinforcement R received for each condition-behavior pair over time t:

$$A(c,b) = \sum_{t=1}^{T} R(c,t)$$

The values in the matrix fluctuate over time based on received reinforcement. They are collected during the execution of a behavior and updated and normalized when behaviors are switched.

The following events produce immediate positive reinforcement:

- E_p: grasped-puck
- E_{gd}: dropped-puck-at-home

- E_{gw}: woke-up-at-home[1]

The following events result in immediate negative reinforcement:

- E_{bd}: dropped-puck-away-from-home
- E_{bw}: woke-up-away-from-home

The events are combined into the following heterogeneous reinforcement function:

$$R_E(c) = \begin{cases} p & \text{if } E_p \\ gd & \text{if } E_{gd} \\ bd & \text{if } E_{bd} \\ gw & \text{if } E_{gw} \\ bw & \text{if } E_{bw} \\ 0 & \text{otherwise} \end{cases}$$

$$p,\ gd,\ gw > 0, \quad bd, bw < 0$$

Two progress estimating functions are used: I and H. I is associated with minimizing interference and is triggered whenever a robot is close to another. If the behavior being executed has the effect of increasing the physical distance to the neighbor, the robot receives positive reinforcement. Conversely, lack of progress away from the neighbor is punished, and after a fixed time period of no progress, the current behavior is terminated.

Formally, I is the intruder avoidance progress function such that:

$$R_I(c,t) = \begin{cases} i & \text{distance to intruder increased} \\ d & \text{otherwise} \end{cases}$$

$$near\_intruder \in c, \quad i > 0, \quad d < 0$$

The other progress estimator, H, is associated with *homing*, and is initiated whenever a puck is grasped. If the distance to home is decreased while H is active, the robot receives positive reinforcement, *status quo* delivers no reinforcement, and movement away from home is punished.

[1] "Waking-up" refers to the event of the internal clock indicating the end of night-time and the beginning of day-time.

Formally, H is the homing progress function such that:

$$R_H(c,t) = \begin{cases} n & \text{nearer to home} \\ f & \text{farther from home} \\ 0 & \text{otherwise} \end{cases}$$

$$have\_puck \in c, \quad n > 0, \quad f < 0$$

The simplest learning algorithm that uses the above reinforcement functions was implemented and tested. The algorithm simply sums the reinforcement over time. The influence of the different types of feedback was weighted by the values of the feedback constants. This is equivalent to weighting their contributions to the sum, as follows:

$$R(c,t) = uR_E(c,t) + vR_I(c,t) + wR_H(c,t)$$

$$u,\ v,\ w \geq 0, \quad (u+v+w) = 1$$

Binary-valued and several real-valued R_E, R_H, and R_I functions were tested. Our results showed that an even distribution of the weights was the most stable but that relatively small differences and variations did not result in faster or more stable learning. This is likely the case because the subgoals in the foraging task are independent and thus their learning speed should not be correlated. Large relative differences in the weights may have had a more significant effect. We also experimented with scaling the reinforcement values in the individual functions, but since all of the values in the $A(c,b)$ matrix were renormalized between behaviors, small changes had no noticeable effects.

6.1 The Control Algorithm

The following is the complete control algorithm used for learning foraging. Behavior selection is induced by events, each of which is a change in the condition predicates. Events are triggered:

1. **externally:** e.g., a robot gets in the way of another. External events include: E_p, E_{gd}, and E_{bd}.

2. **internally:** e.g., the internal clock indicates night-time. Internal events include: E_{gw} and E_{bw}.

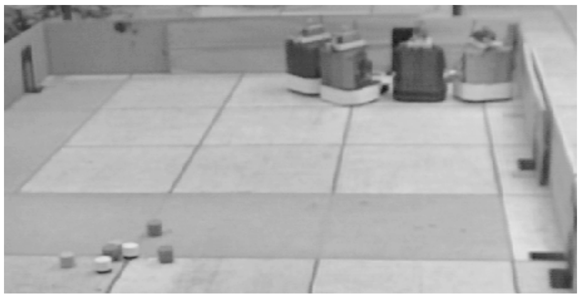

Figure 3: An example of the resting/recharging behavior, triggered by their internal clocks. In this case, the robots have all learned to go home to rest. The figure shows a late stage in the learning, as demonstrated by the small number of remaining pucks.

3. **by progress estimators:** e.g., the interference estimator detects a lack of progress and terminates the current behavior. Estimator events are triggered by: $R_I(c,t) < intruder\_threshold$ and $R_H(c,t) < homing\_threshold$.

Whenever an event is detected, the following control sequence is executed:

1. the current condition-behavior pair is reinforced

2. the current behavior is terminated

3. another behavior is selected, according to the following rule:

   (a) choose an untried behavior if one is available,

   (b) otherwise choose the "best" behavior.

Choosing untried behaviors first encourages exploration. The "best" behavior b for a given condition c is defined to be the one with the highest associated value, $MaxA(c,b)$. We did not find any need to add randomness to the selection mechanism, most likely for the following reasons. First, the intrinsic noisiness of the environment and the robots' sensors and effectors provide enough nondeterministic behavior selection without our explicit control. Second, progress estimators serve to further induce exploration.

Condition				**Behavior**
near-intruder?	have-puck?	at-home?	night-time?	
0	0	0	0	safe-wandering
0	0	0	1	homing
0	0	1	0	safe-wandering
0	0	1	1	resting
0	1	0	0	homing
0	1	0	1	homing
0	1	1	0	safe-wandering
0	1	1	1	resting
1	0	0	0	safe-wandering
1	0	0	1	safe-wandering
1	0	1	0	dispersion
1	0	1	1	resting
1	1	0	0	homing
1	1	0	1	homing
1	1	1	0	safe-wandering
1	1	1	1	resting

Table 1: A part of the desired foraging policy. Only the top-ranked behavior is shown for each condition. The full table contains a total ordering for each condition, resulting in 64 entries.

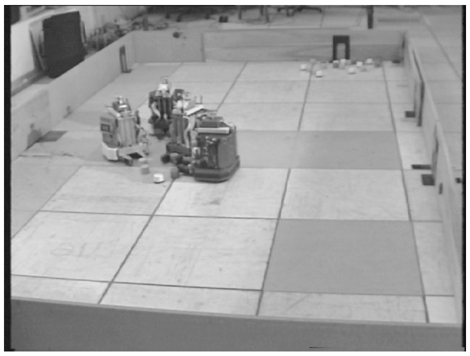

Figure 4: A typical environment state after learning. Most pucks have been collected and brought to the home region. The robots have all learned when to go get the pucks, and are thus competing for those remaining to be moved.

Learning is continuous and incremental over the lifetime of the robot. Figure 4 shows the environment toward the end of the experiment, when most of the pucks have been collected. The learning process consists of adjusting the values in a table accessible to all basic behaviors and reinforcement functions concurrently. The table consists of 64 entries: 2^4 conditions $*$ 4 behaviors. The table is initialized to the average of the minimum and maximum possible $A(c, b)$ values. The desired policy, shown in Table 1, was derived by hand, based on empirical data from the hard-coded foraging experiments in which it was independently tested and compared to alternative solutions (Matarić 1994*a*).

7 Experimental Results and Evaluation

The effectiveness of the proposed reinforcement functions was evaluated by testing three different types of reinforcement. The following three approaches were compared:

1. A monolithic single-goal (puck delivery to the home region) reward function $R(c, t) = R_{E_{gd}}(c, t)$, using the Q-learning algorithm.

2. A heterogeneous reward function using multiple goals: $R(t) = R_E(t)$, using the reinforcement summation algorithm $A(c, b) = \sum_{t=1}^{T} R(c, t)$.

3. A heterogeneous reward function using multiple goals $R(t) = R_E(t)$ and two progress estimator functions $R_H(c, t)$ and $R_I(c, t)$, using the reinforcement summation algorithm $A(c, b) = \sum_{t=1}^{T} R(c, t)$.

Data from sixty trials, twenty of each of the three strategies, were collected and averaged. The experiments were run with groups of four robots, and no significant robot-specific differences were found. Data from runs in which persistent sensor failures occurred were discarded. The data were based on values of $A(c, b)$, which were collected twice per minute during each learning experiment, and once at the completion of the experiment. All experiments lasted 15 minutes. The 15 minute threshold was empirically derived, since the majority of the learning trials reached a steady state after about 10 minutes, except for a small number of rare conditions, discussed below.

Evaluating robot performance is notoriously difficult at least in part because standard metrics for evaluating learning mechanisms, such as absolute time to convergence, do not directly apply. The amount of time required for a robot to discover the correct policy depends on the frequency of external events that trigger different states in its learning space. Additionally, noise and error can make certain parts of the policy fluctuate so waiting for a specific point of absolute convergence is not feasible. Instead, we define convergence as a particular desired policy.

The performance of the three approaches is compared in Figure 5. As described above, Q-learning was tested on the reduced learning space using the enumerated conditions and behaviors and positive reinforcement when a puck is delivered in the home region. This single goal provides insufficient feedback for learning all aspects of foraging, in particular those that rely on accurate delayed credit assignment. The performance of Q-learning was vulnerable to interference from other robots, and declined most rapidly of the three approaches when tested on an increased group size. It is important to note that Q-learning is unable to take advantage of reward discounting in this domain because there is no particularly useful ordering to the sequence of behaviors an agent executes at any time, since the agent's behavior is dependent on the behavior of the others that interact with it during that time.

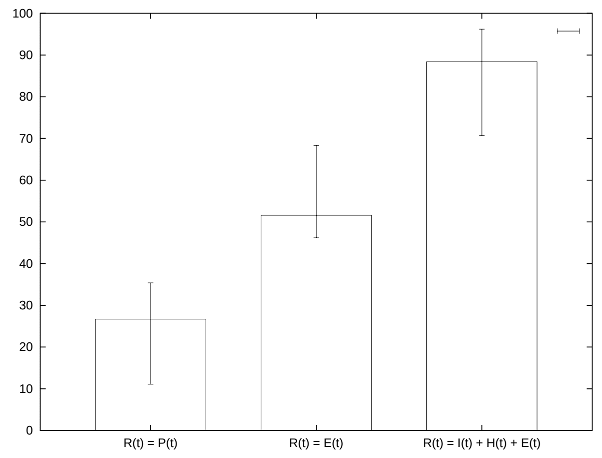

Figure 5: The performance of the three reinforcement strategies on learning to forage. The x-axis shows the three reinforcement strategies. The y-axis maps the percent of the correct policy the robots learned in 15 minutes, averaged over twenty trials. The error bars show the best and worst performance, and the histograms plot the average value.

The use of heterogeneous reward functions results in better performance but also suffers from the credit assignment problem. The non-deterministic environment, induced by the behaviors of the other agents, does not provide consistent rewards over time. Furthermore, this strategy does not prevent thrashing, so certain behaviors are active for an unnecessarily long time. For example, *safe-wandering* and *grasping* are pursued persistently, at the expense of behaviors with delayed reinforcement, such as *homing*. With around 60% of the correct policy learned on the average, it demonstrates that additional structure is necessary to aid the learner in acquiring the rest.

The addition of progress estimators maximizes the use of potentially available information for every condition-behavior pair. As predicted, thrashing is eliminated both in the case of learning the conditions for *dispersion* and *homing* because the progress estimator functions encourage exploration. Furthermore, the impact of fortuitous rewards is minimized. The implicit domain knowledge is effectively spread over the reinforcement in order to guide the learning process continually, thus maximizing the utility of each of the learning trials and consequently speeding up the learning.

Each robot's estimate of its position and the proximity of others was frequently inaccurate due to radio transmission delays. These errors resulted in faulty homing and interference progress estimates. Nonetheless, all condition-behavior pairs that involved carrying a puck converged quickly and did not oscillate. Conversely, the set of conditions associated with finding pucks uniformly took longer to learn, since they had no direct progress measure. Furthermore, the learned values initially tended to oscillate, since the differences between the behavior alternatives were not great, again due to a lack of intermediate rewards. Empirical results showed that noise- and error-induced inconsistencies in the progress estimators did not significantly diminish the benefit of their use in this domain.

Rareness of occurrence of some combinations of conditions was a source of difficulty. In particular, the condition consisting of the onset of night-time while a robot is carrying a puck and avoiding another robot rarely occurred. Consequently, the correct mapping was difficult to learn since the robots did not get a chance to explore all behavior alternatives. This accounts for the incomplete policy even in the case of the most successful reinforcement strategy.

We evaluated the three reinforcement alternatives on groups of three and four robots and found that interference was a detriment in all cases. In general, the more robots were learning at the same time, the longer it took for each individual to converge. This was particularly pronounced for condition-behavior pairs without directly associated progress estimators, such as those involved in the conditions that did not involve carrying a puck.

In an ideal scenario, the presence of other agents would speed up rather than slow down individual learning. However, such synergy requires an environment where individuals benefit from each other's experience and interact according to mutually beneficial social rules. Learning such rules is a challenging problem since they may not necessarily have immediate or even delayed payoff to the individual. Our more recent work has successfully demonstrated an extension of the described approaches to learning such social rules, including *yielding*, *proceeding*, and *broadcasting* (Matarić 1994*b*).

8 Related Work

Very few demonstrations of group behavior on physical robots have been performed to date. This section reviews the most relevant multi-robot learning work as well as the related work on applying reinforcement learning to a single robot.

Kaelbling (1990) used a simple mobile robot to validate several RL algorithms using immediate and delayed reinforcement applied to learning obstacle avoidance. Maes & Brooks (1990) applied a statistical reinforcement learning technique using immediate reward and punishment in order to learn behavior selection for walking on a six-legged robot. The approach was appropriate given the appropriately reduced size of the learning space and the available immediate and accurate reinforcement derived from a contact sensor on the belly of the robot, and a wheel for estimating walking progress.

More delayed reinforcement was used by Mahadevan & Connell (1991*a*) in a box-pushing task implemented on a mobile robot, in which subgoals were introduced to provide more immediate reward. Mahadevan & Connell (1991*b*) experimented with Q-learning using monolithic and partitioned goal functions for learning box-pushing, and found subgoals necessary. Lin (1991*a*) used reinforcement learning on a simulated robot by breaking the navigation task into three behaviors in a similar fashion. Pomerleau (1992) used a supervised connectionist learning approach to train steering control in an autonomous vehicle based on generalizing visual snapshots of the road ahead. Thrun & Mitchell (1993) demonstrated a connectionist approach to learning visual features with a camera mounted on a mobile robot. The features are not assigned by the designer but are instead selected by the network's intermediate representations and thus well suited for the robot's navigation task.

Millán (1994) implemented a connectionist RL scheme on a mobile robot learning navigation in office corridors based on dead-reckoning. The approach utilizes several methods for improving the learning rate, including a coarse codification, or generalization, of the sensory inputs, a hard-wired set of basic reflexes in situations of incorrect generalization, a modular network, and constrained search of the action space. Parker (1994) implemented a non-RL memory-based style of parameter-learning for adjusting activation thresholds used to perform task allocation in a multi-robot system.

Asada et al. (1994) demonstrated coordination of behaviors learned using vision-based reinforcement on a soccer-playing mobile robot shooting at a goal. Tan (1993) explored reinforcement learning in a situated multi-agent domain utilizing communication to share learned information. Lin (1991*b*) studied reinforcement learning in a group of simulated agents.

The formulation we described is a direct extension of behavior-based control (Matarić 1992*a*, Brooks 1991, Brooks 1986). The presented heterogeneous reward functions are related to subgoals (Mahadevan & Connell 1991*a*) as well as subtasks (Whitehead, Karlsson & Tenenberg 1993).

9 Summary

This paper described an approach to formulating reinforcement learning for applying it in noisy, dynamic domains. The concurrent multi-robot learning domain was chosen as the validation environment for the task of learning to forage in a group of four robots. Conditions and behaviors were used to effectively diminish the otherwise prohibitively large learning space. Shaped reinforcement in the form of heterogeneous reward functions and progress estimators, both based on multi-modal sensory feedback, was used to pricipally provide richer and more continuous reinforcement. Experimental results showed that, in the given test domain, both were crucial given the complexity of the environment-robot and robot-robot interactions.

The described formulation is general and compatible with other reinforcement learning algorithms, and should serve to make learning more efficient for a variety of robotic tasks.

References

Asada, M., Uchibe, E., Noda, S., Tawaratsumida, S. & Hosoda, K. (1994), Coordination of Multiple Behaviors Acquired by A Avision-Based Reinforcement Learning, *in* 'Proceedings, IEEE/RSJ/GI International Conference on In telligent Robots and Systems', Munich, Germany.

Atkeson, C. G. (1989), Using Local Models to Control Movement, *in* 'Proceedings, Neural Information Processing Systems Conference'.

Atkeson, C. G., Aboaf, E. W., McIntyre, J. & Reinkensmeyer, D. J. (1988), Model-Based Robot Learning, Technical Report AIM-1024, MIT.

Barto, A. G., Bradtke, S. J. & Singh, S. P. (1993), 'Learning to Act using Real-Time Dynamic Programming', *AI Journal*.

Brooks, R. A. (1986), 'A Robust Layered Control System for a Mobile Robot', *IEEE Journal of Robotics and Automation* **RA-2**, 14–23.

Brooks, R. A. (1990), The Behavior Language; User's Guide, Technical Report AIM-1227, MIT Artificial Intelligence Lab.

Brooks, R. A. (1991), Intelligence Without Reason, *in* 'Proceedings, IJCAI-91'.

Kaelbling, L. P. (1990), Learning in Embedded Systems, PhD thesis, Stanford University.

Lin, L.-J. (1991*a*), Programming Robots Using Reinforcement Learning and Teaching, *in* 'Proceedings, AAAI-91', Pittsburgh, PA, pp. 781–786.

Lin, L.-J. (1991*b*), Self-improving Reactive Agents: Case Studies of Reinforcement Learning Frameworks, *in* 'From Animals to Animats: International Conference on Simulation of Adaptive Behavior', The MIT Press.

Maes, P. & Brooks, R. A. (1990), Learning to Coordinate Behaviors, *in* 'Proceedings, AAAI-91', Boston, MA, pp. 796–802.

Mahadevan, S. & Connell, J. (1990), Automatic Programming of Behavior-based Robots using Reinforcement Learning, Technical report, IBM T. J. Watson Research Center Research Report.

Mahadevan, S. & Connell, J. (1991*a*), Automatic Programming of Behavior-based Robots using Reinforcement Learning, *in* 'Proceedings, AAAI-91', Pittsburgh, PA, pp. 8–14.

Mahadevan, S. & Connell, J. (1991*b*), Scaling Reinforcement Learning to Robotics by Exploiting the Subsumption Architecture, *in* 'Eighth International Workshop on Machine Learning', Morgan Kaufmann, pp. 328–337.

Matarić, M. J. (1992*a*), Behavior-Based Systems: Key Properties and Implications, *in* 'IEEE International Conference on Robotics and Automation, Workshop on Architectures for Intelligent Control Systems', Nice, France, pp. 46–54.

Matarić, M. J. (1992*b*), Designing Emergent Behaviors: From Local Interactions to Collective Intelligence, *in* J.-A. Meyer, H. Roitblat & S. Wilson, eds, 'From Animals to Animats: International Conference on Simulation of Adaptive Behavior'.

Matarić, M. J. (1993), Kin Recognition, Similarity, and Group Behavior, *in* 'Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society', Boulder, Colorado, pp. 705–710.

Matarić, M. J. (1994*a*), Interaction and Intelligent Behavior, Technical Report AI-TR-1495, MIT Artificial Intelligence Lab.

Matarić, M. J. (1994*b*), Learning to Behave Socially, *in* D. Cliff, P. Husbands, J.-A. Meyer & S. Wilson, eds, 'From Animals to Animats: International Conference on Simulation of Adaptive Behavior', pp. 453–462.

Millán, J. D. R. (1994), Learning Reactive Sequences from Basic Reflexes, *in* 'Proceedings, Simulation of Adaptive Behavior SAB-94', The MIT Press, Brighton, England, pp. 266–274.

Moore, A. W. (1992), 'Fast, Robust Adaptive Control by Learning only Forward Models', *Advances in Neural Information Processing 4* pp. 571–579.

Parker, L. E. (1994), Heterogeneous Multi–Robot Cooperation, PhD thesis, MIT.

Pomerleau, D. A. (1992), Neural Network Perception for Mobile Robotic Guidance, PhD thesis, Carnegie Mellon University, School of Computer Science.

Schaal, S. & Atkeson, C. C. (1994), 'Robot Juggling: An Implementation of Memory-Bassed Learning', *Control Systems Magazine* **14**, 57–71.

Sutton, R. (1988), 'Learning to Predict by Method of Temporal Differences', *Machine Learning* **3**(1), 9–44.

Tan, M. (1993), Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents, *in* 'Proceedings, Tenth International Conference on Machine Learning', Amherst, MA, pp. 330–337.

Thrun, S. B. & Mitchell, T. M. (1993), Integrating Inductive Neural Network Learning and Explanation–Based Learning, *in* 'Proceedings, IJCAI-93', Chambery, France.

Watkins, C. J. C. H. & Dayan, P. (1992), 'Q-Learning', *Machine Learning* **8**, 279–292.

Whitehead, S. D., Karlsson, J. & Tenenberg, J. (1993), Learning Multiple Goal Behavior via Task Decomposition and Dynamic Policy Merging, *in* J. H. Connell & S. Mahadevan, eds, 'Robot Learning', Kluwer Academic Publishers, pp. 45–78.