Reward Functions for Accelerated Learning

Maja J Mataric
MIT Artificial Intelligence Laboratory
545 Technology Square #721
Cambridge, MA 02139
maja@ai.mit.edu

Abstract

This paper discusses why traditional reinforcement learning methods, and algorithms applied to those models, result in poor performance in situated domains characterized by multiple goals, noisy state, and inconsistent reinforcement. We propose a methodology for designing reinforcement functions that take advantage of implicit domain knowledge in order to accelerate learning in such domains. The methodology involves the use of heterogeneous reinforcement functions and progress estimators, and applies to learning in domains with a single agent or with multiple agents. The methodology is experimentally validated on a group of mobile robots learning a foraging task.

1 INTRODUCTION

Reinforcement learning (RL) has become the methodology of choice for learning in a variety of different domains. Its convergence properties and potential biological relevance make it an approach worth studying. RL has been shown to perform well in Markovian domains, such as games (Tesauro 1992) and simulations (Sutton 1990). However, it has not yet been proven useful in situated agent domains, in particular when applied to physical robots.

In this paper we discuss why traditional reinforcement learning methods and associated algorithms perform poorly in situated domains with multiple goals, noisy state, and inconsistent reinforcement. We propose a strategy that uses heterogeneous reinforcement functions and progress estimators which take advantage of implicit domain knowledge in order to both enable and accelerate learning. The approach applies to single–agent and multi–agent learning, and is experimentally validated on a group of mobile robots learning to forage. Its performance is compared to standard alternatives.

2 LEARNING IN SITUATED DOMAINS

Reinforcement learning has been studied extensively and its properties are well known (Sutton 1988, Watkins 1989, Kaelbling 1990). Successful applications of RL methodologies to well–behaved domains have encouraged researchers to hypothesize about its value for learning on situated agents such as mobile robots. However, while simulation results are encouraging, work on physical robots has not repeated that success.

The underlying cause for this inconsistency lies in the fundamental assumption of most RL models, the belief that the agent–environment (A–E) interaction can be modeled as a Markov Decision Process (MDP) such that:

1. A and E are synchronized finite state automata.
2. A and E interact in discrete time intervals.
3. A can sense the state of E and use it to act.
4. After A acts, E transitions to a new state.
5. A receives a reward after performing an action.

Unfortunately, the MDP assumption cannot be applied to situated domains. To explain why, we address each of the key aspects of the MDP assumption in turn.

2.1 STATES VS. DESCRIPTORS

The state of a situated agent consists of a collection of properties, some of which are discrete (e.g., inputs from binary sensors), others continuous (e.g., velocities of the wheels). A monolithic descriptor of all of those properties, even for the simplest of agents, is very large, and thus results in a combinatorial explosion in standard RL.

Since not all aspects of a complex state descriptor are relevant at all times, groups of input states can be generalized. Chapman & Kaelbling (1991) and Mahadevan & Connell (1991) demonstrate complementary approaches for generating sufficiently differentiated state spaces that eliminate irrelevant state bits. Although effective, this process requires a large number of trials to obtain the necessary statistical information for pruning the state space.

Much of simulation work attempts to hide continuous state, such as the inputs from complex sensors, by presuming higher–level filters. (e.g., "I see a chair in front of me."). These assumptions have proven unrealistic in physical systems (Brooks & Matarić 1993, Agre & Chapman 1990). In general, in situated domains, which are dynamic and noisy, there is no guarantee that the agent can sense its own state correctly. Furthermore, the agent can usually only perceive local, not global, external state, and do so with inherent limitations. Whitehead & Ballard (1990) addressed a part of this problem in their work on perceptual aliasing, the many–to–one mapping between world and perceptual states, resulting from limited sensing. Finally, sensors are noisy, and often deliver inconsistent data. All of these properties of situated domains drastically interfere with traditional RL algorithms that depend on accurate state information.

2.2 TRANSITIONS VS. EVENTS

Traditional notions of state fit nicely into deterministic state transition models. However, discrete synchronous automata models are inappropriate for situated domains. World and agent states change asynchronously, in response to events not all of which are caused by and in control of the agent. Events take various amounts of time to execute; the same event (as perceived by the agent) can vary in duration under different circumstances and have different consequences. Situated domains are not deterministic, and cannot be usefully modeled by deterministic automata.

The noise and uncertainty in situated domains has specific semantic properties. Consequently, it is not sufficient, or even useful, to add artificial noise to simple finite automata models. For example, while simple and practical, Gaussian noise models are not realistic, and produce artificial dynamics.

Nondeterministic and stochastic models with probabilistic state transitions can more closely model situated domains. However, the information for establishing a stochastic model is not usually readily available, and may take as long to acquire experimentally as a nontrivial learning algorithm would to learn an implicit policy. In general, the problem of constructing world models with enough precision to provide predictive power is unsolved in situated AI.

The nondeterminism and uncertainly properties of situated domains that make modeling difficult also make learning world models even more challenging. Whitehead (1992) eloquently describes why assumptions about world models commonly held in RL do not apply to real–world tasks. Thus, insightful work on building world models for more intelligent exploration (Sutton 1990, Kaelbling 1990) is yet to be generalized to situated domains.

2.3 LEARNING TRIALS

Traditional RL models allow for proving convergence properties of various forms of temporal differencing (TD) applied to deterministic MDP environments (Watkins & Dayan 1992, Barto, Bradtke & Singh 1993, Jaakkola & Jordan 1993). Asymptotic convergence of TD and related learning strategies based on dynamic programming requires infinite trials (Watkins 1989). Simply generating a complete policy requires time exponential in the size of the state space, and the policy approaches optimality as the number of trials approaches infinity. Thus, even in ideal Markovian worlds the number of trials required for learning is prohibitive for all but the smallest state spaces. In situated domains, the agent cannot chose what states it will transition to, and cannot visit all states with equal frequency. Furthermore, experimentation in situated worlds takes time, even if the behavior is performed satisfactorily and appropriate feedback is received.

Thus, convergence of situated learning depends on focusing only on the relevant parts of state and maximizing the amount of information learned from each trial. The popular form of estimating the complexity of a learning problem by measuring the size of the state space is inappropriate. Instead, the smaller the state space, the better the problem is formulated and the fewer learning trials are required. The situated learning problem is still more difficult since even in an appropriately minimized state space a learner may still fail to converge due to insufficient reinforcement. The next section describes the source of this problem.

2.4 REINFORCEMENT VS. FEEDBACK

Design of reinforcement functions is not often discussed, although it is perhaps the most difficult aspect of setting up an RL system. A variation of RL with immediate reinforcement has been successfully applied to a six–legged robot learning to walk (Maes & Brooks 1990). The approach was appropriate given the small size of the search space and the immediate and accurate reinforcement. More delayed reinforcement was used by Mahadevan & Connell (1991) in a box–pushing task, in which subgoals had to be introduced to provide more immediate reward for making the task learnable.

Besides examples on physical learning systems, much of current reinforcement learning work uses two types of reward: immediate, and very delayed. Situated domains, however, tend to fall in between the two popular extremes, providing some immediate rewards, plenty of intermittent delayed ones, and only few very delayed ones. Although very delayed reinforcement, and particularly impulse reinforcement, eliminates the possibility for biasing the learner, most realistic learning problems do not resemble mazes in which the reward is only found at the end. Instead, they often offer some estimate of progress which, even if intermittent, internally biased, and inconsistent, can provide an informative learning signal. Taking advantage of such estimates will be described in a later section.

2.5 MULTIPLE GOALS

Delayed reinforcement is related to the way goals are formulated in a learning system. One of the main properties of situated domains is that agents pursues multiple goals, some of which are maintained continuously, while others are achieved and terminated. In contrast, traditional RL approaches tend to deal with specialized problems in which the learning task can be specified with a single, monolithic goal, and thus directly translated into a monolithic reward function. The policy learned by such a system is very specific, and conflicts with any future learning that may involve different goals resulting from changes in the environment and the agent.

In traditional RL, learning a multi–goal policy required that the goals must be formulated as sequential and consistent subgoals of a monolithic reward function. Singh (1991) addresses such an approach in a simplified navigation problem in which a simulated agent must reach three special states in a particular order. In order to enforce the specific sequence, the state space must explicitly encode what goals have been reached so far, so extra bits are added to the state vector. Although more realistic than single–goal methods, this approach is a direct extension of traditional RL, and therefore scales poorly and fails to address concurrent goals.

Another solution to multiple goals within the traditional framework is to use separate state spaces and reinforcement functions for each of the goals. Whitehead, Karlsson & Tenenberg (1993) describe such an approach and discuss methods for merging existing policies. The work is based on the assumption that the necessary information for utility evaluation is available. This approach scales better than the former and is more general, but pushes the multiple–goal problem to a higher level of control.

This section overviewed the key properties of traditional RL models and algorithms, and their implications for situated domains. In the next section we propose a way of reformulating reinforcement to make learning feasible in such domains.

3 DESIGNING REWARD FUNCTIONS

Traditional RL formulation makes learning in situated domains extremely difficult. Given the complexity and uncertainty of such domains, a more appropriate learning model is needed, one that minimizes the state space and maximizes the amount of learning at each trial. Our previous work has described a reformulation of states and actions into conditions and behaviors in order to significantly diminish the state space (Matarić 1994). In this paper we propose a method for accelerated learning by extending and structuring reward functions to take advantage of domain knowledge.

Rather than encode knowledge explicitly, RL methods hide it in the reinforcement. Domain knowledge can be utilized through a reward–rich and complex reinforcement function, but the process of embedding semantics is usually *ad hoc*. A direct way to utilize implicit domain knowledge is to convert reward functions into error signals, akin to those used in learning control (Jordan & Rumelhart 1992, Atkeson 1990, Schaal & Atkeson 1994). Immediate reinforcement in RL is a weak version of error signals, using only the sign of the error but not the magnitude. Such reinforcement is often not available in situated domains, but intermittent reinforcement can be used similarly, by weighting the reward according to the accomplished progress.

Intermediate reinforcement can be introduced in two ways: 1) by reinforcing multiple goals, and 2) by using progress estimators. We have argued that situated agents have multiple goals, it is most natural to reinforce them individually rather than to attempt to collapse them into a monolithic goal function. We call such a reward scheme a **heterogeneous reinforcement function**. A similar approach was successfully used by Mahadevan & Connell (1991) to speed up learning of a box–pushing task. Multiple goals, however, are not sufficient, if each of the goals requires a complex sequence of actions to accomplish, and thus results in delayed reinforcement. In such cases some progress metric is also necessary.

Progress estimators are partial internal critics. They are associated with specific goals and, when active, provide a metric of improvement relative to those goals. Unlike external critics (Whitehead 1992), progress estimators do not provide a complete oracle but only partial, goal–specific "advice."

Progress estimators are important in noisy worlds because they decrease the learner's sensitivity to intermittent errors by associating a continuous metric with the behavior being executed. Progress estimators also encourage exploration. Without them, the agent can thrash, repeatedly attempting inappropriate behaviors. In synchronized, discrete worlds this effect can be minimized, but not in continuous domains where state changes are triggered by events. In such worlds, the agent has no impetus to terminate behaviors since any behavior may eventually produce a delayed reward. Without built–in knowledge the agent cannot rationally decide when to switch behaviors. Various *ad hoc* solutions can be used, but a principled, domain–related progress estimator is preferable. Finally, progress estimators also decrease the probability of fortuitous rewards, i.e., rewards for an inappropriate behavior that happened, by chance, to achieve the desired goal. While in theory there is not way to eliminate this effect, it can be minimized by using the progress estimate to weight the received reward.

Having proposed multiple goals and progress estimators as two ways of embedding domain knowledge into the reward function in order to accelerate learning, we now validate these ideas experimentally.

4 EXPERIMENTAL DESIGN

In order to validate the proposed approach, we designed experiments for comparing our reinforcement ideas with traditional RL approaches. This section describes the experimental environment, the learning task, and the learning algorithm.

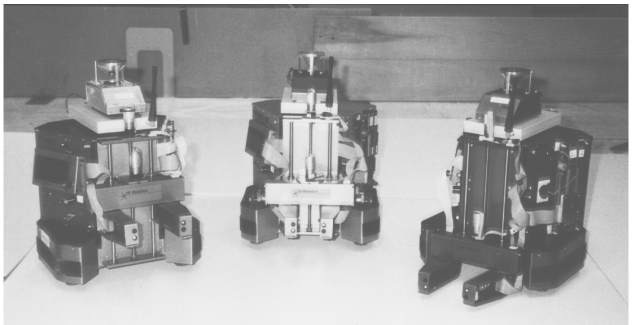

Figure 1: Up to four robots were used in the learning experiments, each consisting of a differentially steerable wheeled base and a gripper for grasping and lifting objects. The robots' sensory capabilities include piezo–electric bump and gripper sensors, infra–red sensors for collision avoidance, internal sensors of motor current, voltage, and position, and a radio transmitter for absolute positioning.

4.1 THE ROBOTS

The learning experiments were conducted on a group of up to four fully autonomous R2 mobile robots with on–board power and sensing. Each robot consists of 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 (see Figure 1). The robots are also equipped with radio transceivers, used for determining absolute position. Position information is obtained by triangulating the distance computed from synchronized ultrasound pulses from two fixed beacons.

The robots are programmed in the Behavior Language, a parallel programming language based on the Subsumption Architecture (Brooks 1990). Their control systems are collections of parallel, concurrently active behaviors, some of which gather sensory information, some drive effectors, and some monitor progress and contribute reinforcement.

4.2 THE LEARNING TASK

The learning task consists of finding a mapping of all conditions and behaviors into the most efficient policy for group foraging. Individually, each robot learns to select the best behavior for each condition, in order to find and take home the most pucks. Foraging was chosen because it is a nontrivial and biologically inspired task, and because our previous group behavior work (Matarić 1992, Matarić 1993) provided the basic behavior repertoire from which to learn behavior selection. The fixed repertoire consisted of the following behaviors:

- *avoiding*
- *dispersing*
- *searching*
- *homing*
- *resting*

Utility behaviors for grasping and dropping objects were included in the robot's capabilities, but were not learned. The robots were expected to learn to associate appropriate conditions for triggering each of the above behaviors. By considering only the space of conditions necessary and sufficient for triggering the behaviors, the state space can be reduced to the cross–product of the following state variables:

- *have-puck?*
- *at-home?*
- *near-intruder?*
- *night-time?*

The conditions for *grasping*, *dropping*, and *avoiding* were built–in. As soon as a robot detects a puck between its fingers, 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 three behaviors were deemed to be "instinctive" because learning them has a high cost. Learning to avoid has a potentially prohibitive damaging cost for the robot, and is an un–intuitive learning task, as it appears to be innate in nature, and can be easily programmed on most systems. Puck manipulation requires a fast and accurate response from the gripper motors, and, like the other basic behaviors, is best suited for parameter learning.

As described, the foraging task may appear trivial, since its state space has been appropriately minimized. In theory, an agent should be able to quickly explore it and learn the optimal policy. In practice, however, such quick and uniform exploration is not possible. Even the relatively small state space presents a challenge to the learner situated in a nondeterministic, noisy and uncertain world. We will see that in its current reformulated version this problem poses a challenge for the traditional RL methodologies, and that it necessitates the proposed reformulation of reinforcement.

This shown learning problem is made more difficult by the fact that the learner is not provided with a model of the world. As discussed earlier, such a model is difficult to obtain, and cannot be assumed to be available. Without it, the agent is faced with implicitly deducing the structure of a dynamic environment including other agents. The behavior of others occasionally facilitates but largely interferes with the individual learning process (see figure 2). Thus, the shown scenario poses a difficult challenge for the reinforcement learning paradigm. The next section shows our solution to the challenge.

4.3 THE LEARNING ALGORITHM

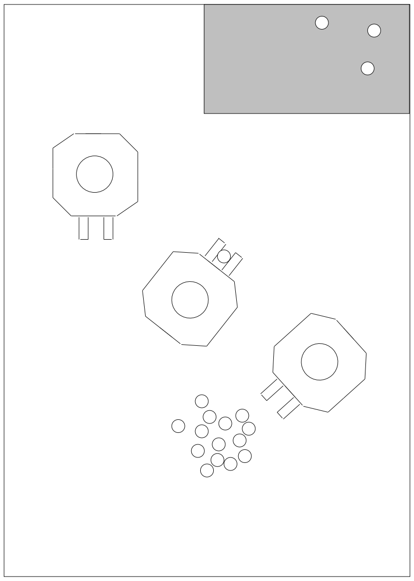

Figure 2: The experimental area in which the learning task was conducted. The workspace is small enough to result in frequent interaction and interference between the robots. The home region is shaded.

The learning algorithm produces and maintains a total order on the appropriateness of behaviors associated with every situation, expressed as a matrix $A(c, b)$. The values in the matrix fluctuate over time based on received reinforcement, and are updated asynchronously, with any received reinforcement.

The following events produce immediate positive reinforcement:

- grasped-puck (p)
- dropped-puck-at-home (gd)
- woke-up-at-home (gw).

The following events result in immediate negative reinforcement:

- dropped-puck-away-from-home (bd)
- woke-up-away-from-home (bw)

The events are combined into the following heterogeneous reinforcement function:

$$E(c) = \begin{cases} p & \text{if new puck grasped} \\ gd & \text{if puck dropped at home} \\ bd & \text{if puck dropped not at home} \\ gw & \text{if woke up at home} \\ bw & \text{if woke up away from home} \\ 0 & \text{otherwise} \end{cases}$$

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

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

Formally, I is the intruder avoiding progress function s.t.:

$$I(c,t) = \begin{cases} m & \text{distance to intruder decreased} \\ n & \text{otherwise} \end{cases}$$

$$near - intruder \in c, \quad m > 0, \quad n < 0$$

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 agent receives positive reinforcement, *status quo* delivers no reinforcement, and movement away from home is punished.

Formally, H is the homing progress function s.t.:

$$H(c,t) = \begin{cases} i & \text{closer to home} \\ j & \text{farther from home} \\ 0 & \text{otherwise} \end{cases}$$

$$have - puck \in c, \quad i > 0, \quad j < 0$$

For the sake of a bottom–up methodology, we implemented and tested the simplest learning algorithm that uses the above reinforcement functions. The algorithm sums the reinforcement over time:

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

$$R(c,t) = E(c,t) + I(c,t) + H(c,t)$$

The influence of the different types of feedback was weighted by the values of the feedback constants. This is equivalent to the alternative of weighting their contributions to the sum:

$$R(c,t) = uE(c,t) + vI(c,t) + wH(c,t)$$

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

Binary–valued and real–valued E, H, and I functions were tested. The results conclusively indicated that differentially weighted reinforcement does not result in faster or more stable learning. This is not surprising, since the subgoals in the foraging task are independent and thus their learning speed is uncorrelated.

4.4 THE CONTROL ALGORITHM

The following is the complete control algorithm used for learning foraging. Behavior selection is induced by events, each of which can be triggered:

1. **externally:** e.g., a robot gets in the way of another,
2. **internally:** e.g., the internal clock indicates night time,
3. **by progress estimators:**, e.g., the interference estimator detects a lack of progress and terminates the current behavior.

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

1. appropriate reinforcement is delivered for the current condition–behavior pair,
2. the current behavior is terminated,
3. another behavior is selected.

Behaviors are selected according to the following rule:

1. choose an untried behavior if one is available,
2. otherwise choose the best behavior.

Learning is continuous and incremental over the lifetime of the agent, thus ensuring that the agent remains responsive to changes in the environment (e.g., no more pucks are left at a particular location) and internal changes in function (e.g., dying battery slows motion down, or a broken sensor affects the perception of conditions). Experimental results are described next.

5 EXPERIMENTAL RESULTS

In order to evaluate the effectiveness of the proposed heterogeneous reinforcement functions and progress estimators, we compared their performance against two alternative learning strategies. The following three approaches were compared:

1. a monolithic single–goal (puck delivery to the home region) reward function using Q-learning, $R(t) = P(t)$,
2. a heterogeneous reinforcement function using multiple goals: $R(t) = E(t)$,
3. a heterogeneous reinforcement function using multiple goals and two progress estimator functions: $R(t) = E(t) + I(t) + H(t)$.

Data from twenty trials of each of the three strategies was collected and averaged. The experiments were run on four different robots, and no significant machine–specific differences were found. Data from runs in which persistent sensor failures occurred were discarded. Values of $A(c, b)$ were collected twice per minute during each learning experiment and the final learning values after each 15 minute run were collated. The 15 minute threshold was chosen empirically, since the majority of the learning trials reached a steady state after about 10 minutes, except for a small number of rare conditions, as discussed below.

Evaluating performance of situated systems is notoriously difficult. Standard metrics for evaluating learning mechanisms, such as 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 make certain parts of the policy fluctuate so waiting for a specific point of absolute convergence is not feasible. Instead, convergence is defined as a relative ordering of condition–behavior pairs.

In the described learning task, the optimal policy was derived empirically, by implementing foraging strategies by hand. The performance of the desired policy was tested independently and compared to alternative solutions in order to establish its superiority relative to the imposed evaluation criteria.

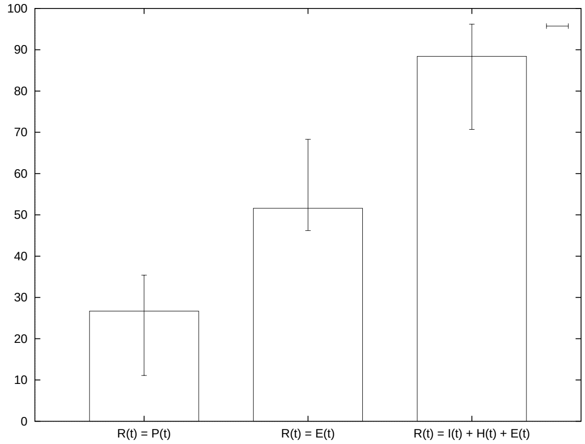

Figure 3: The performance of the three reinforcement strategies on learning to forage. The x-axis shows the three reinforcement strategies. The y-axis maps the percent of the correct policy the agents learned, averaged over twenty trials.

5.1 EVALUATION

The performance of the three approaches is compared in figure 3. The x-axis shows the three reinforcement strategies. The y-axis maps the percent of the correct policy the agents learned, averaged over twenty trials, i.e., the ratio of correct condition–behavior pairings and the optimal policy.

The Q-learning algorithm was tested on the reduced state space using the enumerated conditions and behaviors. The algorithm is functionally equivalent to a simplified version of the second algorithm, using only positive reinforcement for the single goal of dropping a puck in the home region. Given the nondeterminism of the world, and the uncertainty in sensing and state transitions, the 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 very vulnerable to interference from other robots, and declined most rapidly of the three approaches when tested on increased group sizes.

As shown in figure 3, Q does not perform better than chance. However, the partial policy it discovers is not random. It is consistent over all trials and consists of the few condition–behavior pairings that receive immediate and reliable reinforcement. Thus, the performance of Q indicates the difficulty of the learning task at least to the extent of demonstrating the immediately reinforced parts as the only parts it is capable of learning.

The second learning strategy, utilizing reinforcement from multiple goals, outperforms Q because it detects the achievement of the subgoals on the way of the top-level goal of depositing pucks at home. However, it also suffers from the credit assignment problem in the cases of very delayed reinforcement, since the nondeterministic environment with other other agents does not guarantee consistency of rewards over time.

Furthermore, this strategy does not prevent thrashing, so certain behaviors are active unnecessarily long. For example, *searching* and grasping are pursued persistently, at the expense of behaviors with delayed reinforcement, such as *homing*. The performance of heterogeneous reinforcement gives us another evaluation of the difficulty of the proposed learning task. With 60% average performance, it demonstrates that additional structure is necessary to aid the learner. This structure is provided by progress estimators.

The complete heterogeneous reinforcement and progress estimator approach outperforms the other two approaches because it maximizes the use of all available information for every condition and behavior. As predicted, thrashing is eliminated both in the case of learning the conditions for *dispersing* and *homing* because the progress estimator functions enforce better exploration. Furthermore, fortuitous rewards have less impact than in the alternative algorithms. The implicit domain knowledge is effectively spread over the reinforcement in order to guide the learning process continually, thus minimizing the utility of each of the learning trials and consequently speeding up the learning.

5.1.1 Additional Evaluation

In addition to averaging the performance of each algorithm on learning the complete policy, we also evaluated each part of the policy separately, thus measuring when and what each robot was learning. To capture the dynamics of the learning process, each condition–behavior pair was evaluated according to the following three criteria:

1. number of trials required,
2. correctness,
3. stability.

The number of trials was measured relative to a stable solution, whether the solution was optimal or not. The second criterion sought out any tendencies toward incorrect solutions. Finally, the third criterion focused on unstable policies, looking for those in which the behavior orderings tended to fluctuate.

Based on those criteria, some condition–behavior pairs proved to be much more difficult to learn than others. The most prominent source of difficulty was the delay in reinforcement, which had predictable results clearly demonstrated by the performance differences between the three strategies. Learning the conditions for *searching* was difficult as there was no available progress estimator, and the robot could be executing the correct behavior for a long while before reaching pucks and receiving reward. In the mean time it could be repeatedly interrupted by other activities, such as *avoiding* obstacles and intruders, and *resting*.

Another source of difficulty was rareness of occurrence of some combinations of conditions. For example, 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.

6 DISCUSSION

6.0.2 Summing Reinforcement

The presence of positive and negative reinforcement in the continuous learning algorithm guarantees that the learner will not get stuck in local minima, but it also allows for oscillations. Two of the conditions oscillated because the alternatives resulted in equally effective solutions. For example, in situations when the robot is not carrying a puck and encounters an intruder, any motion away from the intruder will be beneficial and rewarded by the progress monitor I. Consequently, *homing* and *searching* are often as effective as *dispersing*. In contrast, if the robot is carrying a puck, then *dispersing* followed by *homing* is optimal and is manifested in the contributions of the I and H progress estimators. As described earlier, it is the combination of the two estimators that speeds up exploration as well as minimizes fortuitous rewards. Only a very specific progress measure that minimizes the travel time to the goal can eliminate this effect. Such optimization is difficult in systems using largely local sensing and control and dealing with interference from other agents. Consequently, the policy the robots found was optimal for the properties of their domain.

In theory, the more reinforcement is used the faster the learning should be. In practice, noise and error in reinforcement could have the opposite effect. The experiments described here demonstrate that even with a significant amount of noise, multiple reinforcers and progress estimators significantly accelerate learning. For example, 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. The fast rate of convergence for behaviors that involved *dispersing* and *homing* result directly from the effects of the two progress estimators. When the two are removed, as in the second algorithm we tested, the performance declines accordingly.

6.0.3 Scaling

The three reinforcement alternatives were evaluated on groups of three and four robots and it was found that interference was a detriment to all three. In general, the more robots were learning at the same time, the longer it took for each to converge. This was particularly pronounced for condition–behavior pairs without directly associated progress estimators, such as those involved in the non–puck carrying states. The only behavior capable of reaping benefits from interference was *dispersing*, which was learned faster and more accurately in crowded situations.

In terms of global effectiveness, the amount of time required to collect all of the pucks, the learned group foraging strategy outperformed hand–coded strategies for individualist foraging in which each agent behaved greedily. Nonetheless, the overall foraging performance could be further improved through the use of additional social behaviors. We have recently begun work on learning a social behavior called *yielding* in order to minimize interference by having only one robot move at a time in crowded situations.

6.0.4 Transition Models

The learning problem presented here, involving a collection of concurrently learning agents in a noisy and uncertain environment, was purposefully chosen for its complexity. In particular, the fact that a state transition model was not available to aid the learner presented one of the major challenges. In particular, the absence of a model made it difficult to compute discounted future rewards for Q-learning, and back–propagation of rewards was used in its place with the same effect.

As argued earlier, such models are not generally available, but partial models could be constructed empirically, either before or during the learning process. The reinforcement functions we proposed take advantage of immediate information from the world to generate reinforcement. Thus, they would have an accelerating effect on any learning domain, regardless of whether a transition model is available. We are interested in applying our reinforcement approach to problems that involve incomplete and approximate state transition models so that we can study the effects of combining immediate reinforcement of the kind we have discussed with discounted future rewards commonly applied to RL problems.

6.0.5 Related Work

Work related to the approaches introduced here has been discussed throughout the paper. To summarize, the heterogeneous reward functions we presented are related to subgoals used by Mahadevan & Connell (1991) as well as subtasks used by Whitehead et al. (1993). However, unlike previous work, which has focused on learning action sequences, we use a higher level of description. The proposed subgoals are directly tied to behaviors which are used as the basis of learning. Similarly, progress estimators are also mapped to one or more behaviors, and expedite learning of the associated goals, unlike a single complete external critic used with a monolithic reinforcement function (Whitehead 1992).

The presented work is, to the best of our knowledge, the first attempt at applying reinforcement learning to a collection of physical robots learning a complex task consisting of multiple goals. Tan (1993) has applied traditional RL to a simulated multi–agent domain. Due to the simplicity of the simulated environment, the work can assume all of the MDP axioms that we have shown to be invalid for physical robots. Furthermore, this and other work that uses communication between agents relies on the assumption that agents can correctly exchange learned information. This does not hold true on physical systems whose noise and uncertainty properties extend to the communication channels.

7 SUMMARY

The goal of this paper has been to bring to light some of the important properties of situated domains, and their impact on the existing reinforcement learning strategies. We have discussed why modeling agent–

world interactions as Markov Decision Processes is unrealistic, and how the traditional notions of state, state transitions, goals, and reward functions fail to transfer to the physical world.

We have argued that the noisy and inconsistent properties of complex worlds require the use of domain knowledge. We proposed a principled approach to embedding such knowledge into the reinforcement based on utilizing heterogeneous reward functions and goal–specific progress estimators. We believe that these strategies take advantage of the information readily available to situated agents, make learning possible in complex dynamic worlds, and accelerate it in any domain.

Acknowledgements

I am grateful to to the two anonymous reviewers for detailed helpful comments on an earlier draft of this paper, and to Rich Sutton for encouraging me to write up the the results. The research reported here was done at the MIT Artificial Intelligence Laboratory, and supported in part by the Jet Propulsion Laboratory contract 959333 and in part by the Advanced Research Projects Agency under Office of Naval Research grant N00014–91–J–4038.

References

Agre, P. E. & Chapman, D. (1990), What are plans for?, *in* P. Maes, ed., 'Designing Autonomous Agents', MIT Press, pp. 17–34.

Atkeson, C. G. (1990), Memory-Based Approaches to Approximating Continuous Functions, *in* 'Proceedings, Sixth Yale Workshop on Adaptive and Learning Systems'.

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

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

Brooks, R. A. & Matarić, M. J. (1993), Real Robots, Real Learning Problems, *in* 'Robot Learning', Kluwer Academic Press, pp. 193–213.

Chapman, D. & Kaelbling, L. P. (1991), Input Generalization in Delayed Reinforcement Learning: An Algorithm and Performance Comparisons, *in* 'Proceedings, IJCAI-91', Sydney, Australia.

Jaakkola, T. & Jordan, M. I. (1993), 'On the Convergence of Stochastic Iterative Dynamic Programming Algorithms', *Submitted to Neural Computation*.

Jordan, M. I. & Rumelhart, D. E. (1992), 'Forward Models: Supervised Learning with a Distal Teacher', *Cognitive Science* **16**, 307–354.

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

Maes, P. & Brooks, R. A. (1990), Learning to Coordinate Behaviors, *in* 'Proceedings, AAAI-90', Boston, MA, pp. 796–802.

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

Matarić, M. J. (1992), 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), Interaction and Intelligent Behavior, PhD thesis, MIT.

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

Singh, S. P. (1991), Transfer of Learning Across Compositions of Sequential Tasks, *in* 'Proceedings, Eighth International Conference on Machine Learning', Morgan Kaufmann, Evanston, Illinois, pp. 348–352.

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

Sutton, R. S. (1990), Integrated Architectures for Learning, Planning and Reacting Based on Approximating Dynamic Programming, *in* 'Proceedings, Seventh International Conference on Machine Learning', Austin, Texas.

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

Tesauro, G. (1992), Practical Issues in Temporal Difference Learning, *in* J. E. Moody, S. J. Hanson & R. P. Lippmann, eds, 'Advances in Neural Information Processing Systems 4', Morgan Kaufmann, pp. 259–267.

Watkins, C. J. C. H. (1989), Learning from Delayed Rewards, PhD thesis, King's College, Cambridge.

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

Whitehead, S. D. (1992), Reinforcement Learning for the Adaptive Control of Perception and Action, PhD thesis, University of Rochester.

Whitehead, S. D. & Ballard, D. H. (1990), Active Perception and Reinforcement Learning, *in* 'Proceedings, Seventh International Conference on Machine Learning', Austin, Texas.

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.