Hierarchical Optimal Control of MDPs

Amy McGovern
Univ. of Massachusetts
Amherst, MA 01003
amy@cs.umass.edu

Doina Precup
Univ. of Massachusetts
Amherst, MA 01003
dprecup@cs.umass.edu

Balaraman Ravindran
Univ. of Massachusetts
Amherst, MA 01003
ravi@cs.umass.edu

Satinder Singh
Univ. of Colorado
Boulder, CO 80309
baveja@cs.colorado.edu

Richard S. Sutton
Univ. of Massachusetts
Amherst, MA 01003
rich@cs.umass.edu

Abstract

Fundamental to reinforcement learning, as well as to the theory of systems and control, is the problem of representing knowledge about the environment and about possible courses of action hierarchically, at a multiplicity of interrelated temporal scales. For example, a human traveler must decide which cities to go to, whether to fly, drive, or walk, and the individual muscle contractions involved in each step. In this paper we survey a new approach to reinforcement learning in which each of these decisions is treated uniformly. Each low-level action and high-level course of action is represented as an *option*, a (sub)controller and a termination condition. The theory of options is based on the theories of Markov and semi-Markov decision processes, but extends these in significant ways. Options can be used in place of actions in all the planning and learning methods conventionally used in reinforcement learning. Options and models of options can be learned for a wide variety of different subtasks, and then rapidly combined to solve new tasks. Options enable planning and learning simultaneously at a wide variety of times scales, and toward a wide variety of subtasks, substantially increasing the efficiency and abilities of reinforcement learning systems.

Introduction

The field of reinforcement learning is entering a new phase, in which it considers learning at multiple levels, and at multiple temporal and spatial scales. Such hierarchical approaches are advantageous in very large problems because they provide a principled way of forming approximate solutions. They also allow much greater flexibility and richness in *what* is learned. In particular, we can consider not just one task, but a whole range of tasks, solve them independently, and yet be able to combine their individual solutions quickly to solve new overall tasks. We also allow the learner to work not just with primitive actions, but with higher-level, temporally-extended actions, called *options*. In effect, the learner can choose among subcontrollers rather than just low-level actions. This new direction is also consonant with reinforcement learning's roots in artificial intelligence, which has long focused on planning and knowledge representation at higher levels. In this paper we survey our work in recent years forming part of this trend.

From the point of view of classical control, our new work constitutes a hierarchical approach to solving Markov decision problems (MDPs), and in this paper we present it in that way. In particular, our methods are closely related to semi-Markov methods commonly used with discrete-event systems. The work we describe differs from most other hierarchical approaches in that we do not lump states together into larger states. We keep the original state representation and instead alter the temporal aspects of the actions.

In this paper we survey our recent and ongoing work in temporal abstraction and hierarchical control of Markov decision processes (Precup, Sutton & Singh 1998a,b, in prep.). This work is part of a larger trend toward focusing on these issues by many researchers in reinforcement learning (e.g. Singh, 1992a,b; Kaelbling, 1993; Lin, 1993; Dayan & Hinton, 1993; Thrun & Schwartz, 1995; Huber & Grupen, 1997; Dietterich, 1998; Parr & Russell, 1998).

Markov Decision Processes

In this section we briefly describe the conventional reinforcement learning framework of discrete-time, finite *Markov decision processes*, or *MDPs*, which forms the basis for our extensions to temporally extended courses of action. In this framework, a learning *agent* interacts with an *environment* at some discrete, lowest-level time scale $t = 0, 1, 2, \ldots$. On each time step the agent perceives the state of the environment, $s_t \in \mathcal{S}$, and on that basis chooses a primitive action, $a_t \in \mathcal{A}$. In response to each action, a_t, the environment produces one step later a numerical reward, r_{t+1}, and a next state, s_{t+1}. The environment's transition dynamics are modeled by one-step state-transition probabilities,

$$p^a_{ss'} = \Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\},$$

and one-step expected rewards,

$$r^a_s = E\{r_{t+1} \mid s_t = s, a_t = a\},$$

for all $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$. These two sets of quantities together constitute the *one-step model* of the environment.

The agent's objective is to learn an *optimal Markov policy*, a mapping from states to probabilities of taking each available primitive action, $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$, that maximizes the expected discounted future reward from each state s:

$$V^\pi(s) = E\Big\{r_{t+1} + \gamma r_{t+1} + \gamma^2 r_{t+1} + \cdots \Bigm| s_t = s, \pi\Big\}$$

where $\gamma \in [0, 1]$ is a *discount-rate* parameter. $V^\pi(s)$ is called the *value* of state s under policy π, and V^π is called the *state-value function* for π. The unique *optimal* state-value function, $V^*(s) = \max_\pi V^\pi(s), \forall s \in \mathcal{S}$, gives the value of a state under an optimal policy. Any policy that achieves V^* is by definition an optimal policy. There are also action value functions, $Q^\pi : \mathcal{S} \times \mathcal{A} \mapsto \Re$ and $Q^* : \mathcal{S} \times \mathcal{A} \mapsto \Re$, that give the value of a state given that a particular action is initially taken in it, and a given policy is followed afterwards.

Options

We use the term *options* for our generalization of primitive actions to include temporally extended courses of action. Formally, an option consists of three components: an input set $\mathcal{I} \subseteq \mathcal{S}$, a policy $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0,1]$, and a termination condition $\beta : \mathcal{S} \mapsto [0,1]$. An option $\langle \mathcal{I}, \pi, \beta \rangle$ is available in state s if and only if $s \in \mathcal{I}$. If the option is taken, then actions are selected according to π until the option terminates stochastically according to β. In particular, the next action a_t is selected according to the probability distribution $\pi(s_t, \cdot)$. The environment then makes a transition to state s_{t+1}, where the option either terminates, with probability $\beta(s_{t+1})$, or else continues, determining a_{t+1} according to $\pi(s_{t+1}, \cdot)$, possibly terminating in s_{t+2} according to $\beta(s_{t+2})$, and so on. When the option terminates, the agent has the opportunity to select another option.

The input set and termination condition of an option together restrict its range of application in a potentially useful way. In particular, they limit the range over which the option's policy needs to be defined. For example, a handcrafted policy π for a mobile robot to dock with its battery charger might be defined only for states $\mathcal{I}$ in which the battery charger is within sight. The termination condition β would be defined to be 1 outside of $\mathcal{I}$ and when the robot is successfully docked. It is natural to assume that all states where an option might continue are also states where the option might be taken (i.e., that $\{s : \beta(s) < 1\} \subseteq \mathcal{I}$). In this case, π needs to be defined only over $\mathcal{I}$ rather than over all of $\mathcal{S}$.

The definition of options is crafted to make them as much like actions as possible, except temporally extended. Because options terminate in a well defined way, we can consider policies that select options instead of primitive actions. Let the set of options available in state s be denoted $\mathcal{O}_s$; the set of all options is denoted $\mathcal{O} = \bigcup_{s \in \mathcal{S}} \mathcal{O}_s$. When initiated in a state s_t, the Markov policy over options $\mu : \mathcal{S} \times \mathcal{O} \mapsto [0,1]$ selects an option $o \in \mathcal{O}_{s_t}$ according to probability distribution $\mu(s_t, \cdot)$. The option o is then taken in s_t, determining actions until it terminates in s_{t+k}, at which point a new option is selected, according to $\mu(s_{t+k}, \cdot)$, and so on. In this way a policy over options, μ, determines a policy over actions, or *flat policy*, $\pi = f(\mu)$. Henceforth we use the unqualified term *policy* for Markov policies over options, which include Markov flat policies as a special case. Note, however, that $f(\mu)$ is typically not Markov because the action taken in a state depends on which option is being taken at the time, not just on the state. We define the value of a state s under a general flat policy π as the expected return if the policy is started in s:

$$V^\pi(s) \stackrel{\text{def}}{=} E\Big\{ r_{t+1} + \gamma r_{t+2} + \cdots \;\Big|\; \mathcal{E}(\pi, s, t) \Big\},$$

where $\mathcal{E}(\pi, s, t)$ denotes the event of π being initiated in s at time t. The value of a state under a general policy (i.e., a policy over options) μ can then be defined as the value of the state under the corresponding flat policy: $V^\mu(s) \stackrel{\text{def}}{=} V^{f(\mu)}(s)$.

MDP + Options = SMDP

Options are closely related to the actions in a special kind of decision problem known as a *semi-Markov decision process*, or *SMDP* (e.g., see Puterman, 1994). In fact, a fixed set of options defines a new discrete-time SMDP embedded within the original MDP, as suggested by Figure 1. The top panel shows the state trajectory over discrete time of an MDP, the middle panel shows the larger state changes over continuous time of an SMDP, and the last panel shows how these two levels of analysis can be superimposed through the use of options. In this case the underlying base system is an MDP, with regular, single-step transitions, while the options define larger transitions, like those of an SMDP, that last for a number of discrete steps. All the usual SMDP theory applies to the superimposed SMDP defined by the options but, in addition, we have an explicit interpretation of them in terms of the underlying MDP. We will now outline the way in which some of the SMDP results can be interpreted and used in the context of MDPs and options.

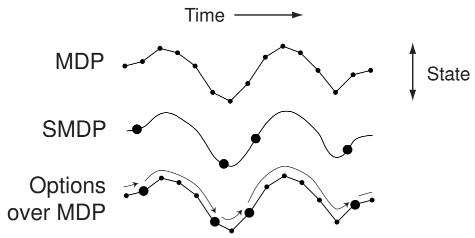

Figure 1: The state trajectory of an MDP is made up of small, discrete-time transitions, whereas that of an SMDP comprises larger, continuous-time transitions. Options enable an MDP trajectory to be analyzed at either level.

Planning with options requires a model of their consequences. Fortunately, the appropriate form of model for options, analogous to the r^a_s and $p^a_{ss'}$ defined earlier for actions, is known from existing SMDP theory. For each state in which an option may be started, this kind of model predicts the state in which the option will terminate and the total reward received along the way. These quantities are discounted in a particular way. For any option o, let $\mathcal{E}(o, s, t)$ denote the event of o being initiated in state s at time t. Then the reward part of the model of o for state $s \in \mathcal{S}$ is

$$r_s^o = E\left\{ r_{t+1} + \gamma r_{t+2} + ... + \gamma^{k-1} r_{t+k} \;\middle|\; \mathcal{E}(o,s,t) \right\}, \quad (1)$$

where $t+k$ is the random time at which o terminates. The state-prediction part of the model of o for state s is

$$\begin{aligned} p_{ss'}^o &= \sum_{j=1}^{\infty} \gamma^j \Pr\{s_{t+k} = s', k = j \mid \mathcal{E}(o,s,t)\} \\ &= E\{\gamma^k \delta_{s' s_{t+k}} \mid \mathcal{E}(o,s,t)\}, \quad (2) \end{aligned}$$

for all $s' \in \mathcal{S}$, under the same conditions, where $\delta_{ss'}$ is an identity indicator, equal to 1 if $s = s'$, and equal to 0 otherwise. Thus, $p_{ss'}^o$ is a combination of the likelihood that s' is the state in which o terminates together with a measure of how delayed that outcome is relative to γ. We call this kind of model a *multi-time model* (Precup and Sutton, 1998) because it describes the outcome of an option not at a single time, but at potentially many different times, appropriately combined.

Using multi-time models we can write Bellman equations for general policies and options. For instance, let us denote a restricted set of options by $\mathcal{O}$ and the set of all policies selecting only from options in $\mathcal{O}$ by $\Pi(\mathcal{O})$. Then the optimal value function given that we can select only from $\mathcal{O}$ is

$$V_{\mathcal{O}}^*(s) = \max_{o \in \mathcal{O}_s} \left[r_s^o + \sum_{s'} p_{ss'}^o V_{\mathcal{O}}^*(s') \right].$$

A corresponding *optimal policy*, denoted $\mu_{\mathcal{O}}^*$, is any policy that achieves $V_{\mathcal{O}}^*$, i.e., for which $V^{\mu_{\mathcal{O}}^*}(s) = V_{\mathcal{O}}^*(s)$ in all states $s \in \mathcal{S}$. If $V_{\mathcal{O}}^*$ and models of the options are known, then $\mu_{\mathcal{O}}^*$ can be formed by choosing in any proportion among the maximizing options in the equation above.

It is straightforward to extend MDP planning methods to SMDPs. The policies found using temporally abstract options are approximate in the sense that they achieve only $V_{\mathcal{O}}^*$, which is typically less than V^*.

The Rooms Example

As a simple illustration of planning with options, consider the *rooms example*, a gridworld environment of four rooms shown in Figure 2. The cells of the grid correspond to the states of the environment. From any state the agent can perform one of four actions, `up`, `down`, `left` or `right`. These actions usually move the agent in the corresponding direction, but with 1/3 probability they instead move the agent in another, random direction. In each of the four rooms, the system is also provided with two built-in *hallway options* that take the agent from anywhere within the room to one of the hallway cells leading out of the room.

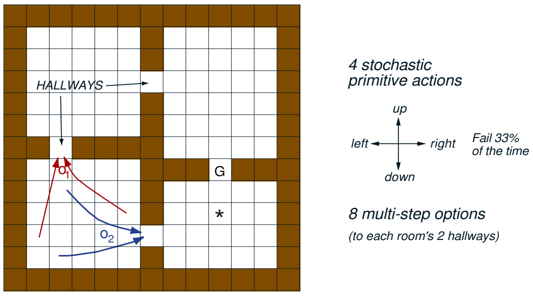

Figure 2: The rooms example is a gridworld environment with stochastic cell-to-cell actions and room-to-room hallway options. Two of the hallway options are suggested by the arrows labeled o_1 and o_2. The label G indicates a location used as goal.

To complete the specification of the planning problem we designate one state as a goal, say the state labeled G, by providing a reward of +1 on arrival there. Figure 3 shows the results of applying synchronous value iteration (SVI) to this problem with and without options. The upper part of the figure shows the value function after the first two iterations of SVI using just primitive actions. The region of accurately valued states moved out by one cell on each iteration, but after two iterations most states still had their initial arbitrary value of zero. In the lower part of the figure are shown the corresponding value functions for SVI with the hallway options. In the first iteration all states in the rooms adjacent to the goal state became accurately valued, and in the second iteration all the states became accurately valued. Rather than planning step-by-step, the hallway options enable the planning to proceed at a higher

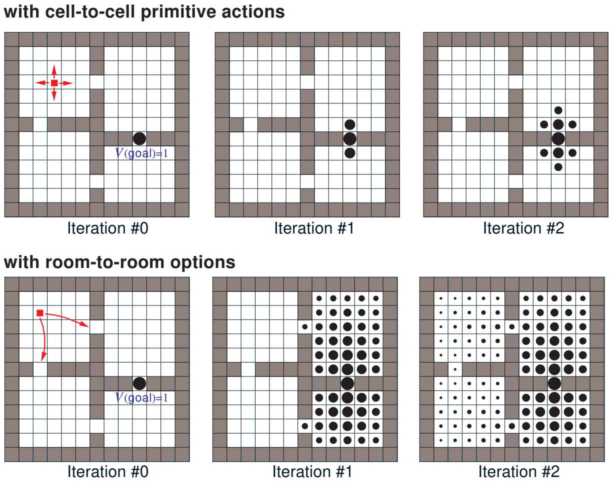

Figure 3: Value functions formed over iterations of planning by synchronous value iteration with primitive actions and with hallway options. The hallway options enable planning to proceed room-by-room rather than cell-by-cell. The area of the disk in each cell is proportional to the estimated value of the state, where a disk that just fills a cell represents a value of 1.0. We use discounting with $\gamma = 1$ for this task.

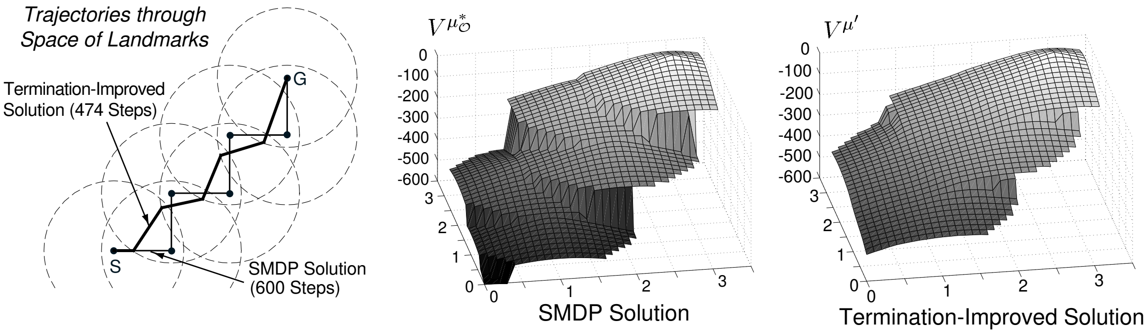

Figure 4: Termination improvement in navigating with landmark-directed controllers. The task (left) is to navigate from S to G in minimum time using options based on controllers that run each to one of seven landmarks (the black dots). The circles show the region around each landmark within which the controllers operate. The thin line shows the optimal behavior that uses only these controllers run to termination, and the thick line shows the corresponding termination improved behavior, which cuts the corners. The right panels show the state-value functions for the SMDP-optimal and termination-improved policies. Note that the latter is greater

level, room-by-room, and thus be much faster.

Termination Improvement

So far we have assumed that an option, once started, must be followed until it terminates. This assumption is necessary to apply the theoretical machinery of SMDPs. On the other hand, the whole point of the options framework is that one also has an interpretation in terms of the underlying MDP. This enables us to consider interrupting options before they would terminate normally. For example, suppose we have determined the option-value function $Q^\mu(s, o)$ for some policy μ and for all state–options pairs s, o that could be encountered while following μ. This function tells us how well we do while following μ and committing irrevocably to each option, but it can also be used to re-evaluate our commitment on each step. Suppose at time t we are in the midst of executing option o. If o is Markov in s, then we can compare the value of continuing with o, which is $Q^\mu(s_t, o)$, to the value of terminating o and selecting a new option according to μ, which is $V^\mu(s) = \sum_{o'} \mu(s, o') Q^\mu(s, o')$. If the latter is more highly valued, then why not terminate o and allow the switch? Indeed, we have shown that this new way of behaving is guaranteed to be better. We characterize this as an improvement in the termination condition of the original option, i.e., as a *termination improvement*.

Figure 4 shows a simple example of termination improvement. Here the task is to navigate from a start location to a goal location within a continuous two-dimensional state space. The actions are movements of 0.01 in any direction from the current state. Rather than working with these low-level actions, infinite in number, we introduce seven landmark locations. For each landmark we define a controller that takes us to the landmark in a direct path. Each controller is only applicable within a limited range of states, in this case within a certain distance of the corresponding landmark. Each controller then defines an option: the circular region around the controller's landmark is the option's input set, the controller itself is the policy, and arrival at the target landmark is the termination condition. We denote the set of seven landmark options by $\mathcal{O}$. Any action within 0.01 of the goal location transitions to the terminal state, $\gamma = 1$, and the reward is -1 on all transitions, which makes this a minimum-time task.

One of the landmarks coincides with the goal, so it is possible to reach the goal while picking only from $\mathcal{O}$. The optimal policy within $\mathcal{O}$ runs from landmark to landmark, as shown by the thin line in Figure 4. This is the optimal solution to the SMDP defined by $\mathcal{O}$ and is indeed the best that one can do while picking only from these options. But of course one can do better if the options are not followed all the way to each landmark. The trajectory shown by the thick line in Figure 4 cuts the corners and is shorter. This is the termination-improved policy with respect to the SMDP-optimal policy. The termination improvement policy takes 474 steps from start to goal which, while not as good as the optimal policy in primitive actions (425 steps), is much better, for no additional cost, than the SMDP-optimal policy, which takes 600 steps. The state-value functions, $V^{\mu^*_{\mathcal{O}}}$ and $V^{\mu'}$ for the two policies are also shown on the right in Figure 4.

Another illustration of termination improvement in a more complex task is shown in Figure 5. The task here is to fly a reconnaissance plane from base, to observe as many sites as possible, from a given set of sites, and return to base without running out of fuel. The local weather at each site flips between cloudy and clear according to independent Poisson processes. If the sky at a given site is cloudy when the plane gets there, no observation is made and the reward is 0. If the sky is clear, the plane gets a reward, according to the importance of the site. The plane has a limited amount of fuel, and it consumes one unit of fuel during each time tick. If the fuel runs out before reaching the base, the plane crashes and receives a reward of -100.

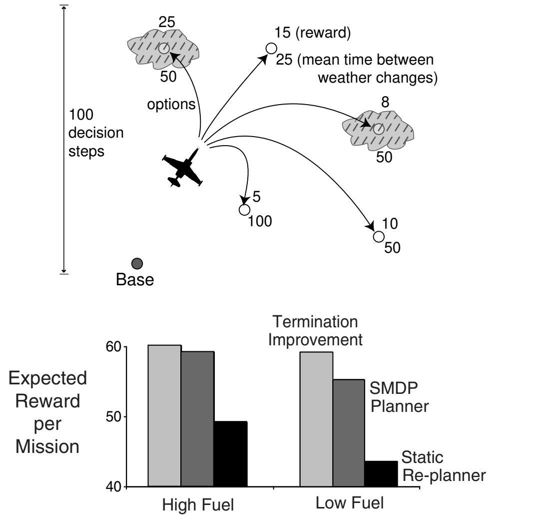

Figure 5: The mission planning task and the performance of policies constructed by SMDP methods, termination improvement of the SMDP policy, and an optimal static re-planner that does not take into account possible changes in weather conditions.

The primitive actions are tiny movements in any direction (there is no inertia). The state of the system is described by several variables: the current position of the plane, the fuel level, the sites that have been observed so far, and the current weather at each of the remaining sites. This state-action space has approximately 24 billion elements (assuming 100 discretization levels of the continuous variables), making the problem intractable by normal dynamic programming methods. We introduced options that can take the plane to each of the sites (including the base), from any position in the input space. The resulting SMDP has only 874,800 elements and it is feasible to determine $V^*_{\mathcal{O}}(s')$ exactly for all sites s'. From this solution and the model of the options, we can determine $Q^*_{\mathcal{O}}(s,o) = r^o_s + \sum_{s'} p^o_{ss'} V^*_{\mathcal{O}}(s')$ for any option o and any state s in the whole space.

The data in figure 5 compares the SMDP and termination improvement policies found for the problem with the performance of a static planner, which exhaustively searches for the best tour assuming the weather does not change, and then re-plans whenever the weather does change. The policy obtained by the termination improvement approach performs significantly better than the SMDP policy, which in turn is significantly better than the static planner.

Intra-Option Learning

Optimal value functions can be determined by learning as well as by planning. One natural approach is to use SMDP learning methods (Bradtke & Duff (1995), Parr & Russell (1998), Mahadevan et al. (1997), and McGovern, Sutton & Fagg (1997)), which treat complete option executions just as primitive actions are treated in conventional reinforcement learning methods. One drawback to these methods is that they need to execute an option to termination before they can learn about it. Because of this, they can only be applied to one option at a time—the option that is executing at that time. More interesting and potentially more powerful methods are possible by taking advantage of the structure inside each option. In particular, if the options are Markov and we are willing to look *inside* them, then we can use special temporal-difference methods to learn usefully about an option before the option terminates. This is the main idea behind *intra-option* methods .

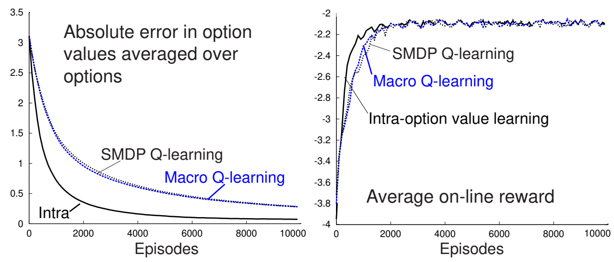

Figure 6: Comparison of SMDP, intra-option and Macro Q-learning. Intra-option methods converge faster to correct values.

Figure 6 shows an illustration of the advantages of intra-option learning in the rooms example. In this case small negative rewards were introduced at all the states, and the goal was located at G. We experimented with two SMDP methods: one-step SMDP Q-learning (Bradtke & Duff, 1995) and a hierarchical form of Q-learning called *Macro Q-learning* (McGovern, Sutton & Fagg, 1997). Although the SMDP methods can be used here, they were much slower than the intra-option method.

Analyzing the Effects of Options

Adding options can either accelerate or retard learning depending on their appropriateness to the particular task. In another aspect of our work, we are trying to break down the effect of options into components that can be measured and studied independently. The two components that we have studied so far are: the effect of options on initial exploratory behavior, independent of learning, and the effect of learning with options on the speed at which correct value information is propagated, independent of the behavior. We have found that both of these effects are significant.

We have measured these effects in gridworld tasks and in the larger, simulated robotics task shown on the left in Figure 7. This is a foraging task in a two-dimensional space. The circular robot inhabits a world with two rooms, one door connecting them, and one food object. The robot

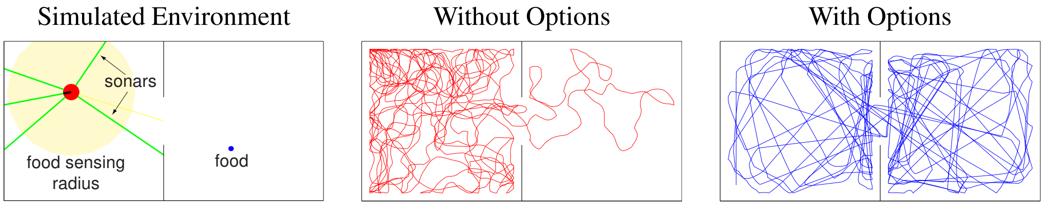

Figure 7: The simulated robotic foraging task. The picture on of the environment shows the five sonars, the doorway sensors, and the food sensor. The graphs on the right hand side represent the position of the robot during a random walk.

has simulated sonars to sense the distance to the nearest wall in each of five fixed directions and simple inertial dynamics, with friction and inelastic collisions with the walls. We provide two options, one which orients the robot towards the door, and the other which drives the robot forward until it encounters a wall. Because the state space is continuous and large, we used a tile-coding function approximators is necessary.

To assess the effect of options, we examined the behavior for 100,000 steps when the actions were selected randomly from the primitive actions only and from both the primitive actions and the options. The two right panels in Figure 7 show a projection of one such trajectory onto the two spatial dimensions. The options have a large influence on this exploratory behavior. With options, the robot crosses more often between the two rooms and travels more often with high velocity. In preliminary results we have also been able to show faster learning of efficient foraging strategies through the use of options.

Closing

In this paper we have briefly surveyed a number of ways in which temporal abstraction can contribute to the hierarchical control of MDPs. We have presented some of the basic theory and several suggestive examples, but many of the most interesting questions remain open.

Acknowledgments

The authors gratefully acknowledge the contributions to these ideas of many colleagues, especially Andrew Barto, Ron Parr, Tom Dietterich, Andrew Fagg, Leo Zelevinsky and Manfred Huber. We also thank Paul Cohen, Robbie Moll, Mance Harmon, Sascha Engelbrecht, and Ted Perkins for helpful reactions and constructive criticism. This work was supported by NSF grant ECS-9511805 and grant AFOSR-F49620-96-1-0254, both to Andrew Barto and Richard Sutton. Satinder Singh was supported by NSF grant IIS-9711753.

References

Bradtke, S. J., and Duff, M. O. 1995. Reinforcement learning methods for continuous-time Markov decision problems. In *Advances in Neural Information Processing Systems 7*, 393–400. MIT Press.

Dayan, P., and Hinton, G. E. 1993. Feudal reinforcement learning. In *Advances in Neural Information Processing Systems 5*, 271–278. Morgan Kaufmann.

Dietterich, T. G. 1998. The MAXQ method for hierarchical reinforcement learning. In *Proc. of the 15th Intl. Conf. on Machine Learning*. Morgan Kaufmann.

Huber, M., and Grupen, R. A. 1997. A feedback control structure for on-line learning tasks. *Robotics and Autonomous Systems* 22(3-4):303–315.

Kaelbling, L. P. 1993. Hierarchical learning in stochastic domains: Preliminary results. In *Proc. of the 10th Intl. Conf. on Machine Learning*, 167–173. Morgan Kaufmann.

Lin, L.-J. 1993. *Reinforcement Learning for Robots Using Neural Networks*. Ph.D. Dissertation, Carnegie Mellon University.

Mahadevan, S.; Marchallek, N.; Das, T. K.; and Gosavi, A. 1997. Self-improving factory simulation using continuous-time average-reward reinforcement learning. In *Proc. of the 14th Intl. Conf. on Machine Learning*, 202–210. Morgan Kaufmann.

McGovern, A.; Sutton, R. S.; and Fagg, A. H. 1997. Roles of macro-actions in accelerating reinforcement learning. In *Grace Hopper Celebration of Women in Computing*, 13–17.

Parr, R., and Russell, S. 1998. Reinforcement learning with hierarchies of machines. In *Advances in Neural Information Processing Systems 10*. MIT Press.

Precup, D.; Sutton, R. S.; and Singh, S. 1998a. Multi-time models for temporally abstract planning. In *Advances in Neural Information Processing Systems 10*. MIT Press.

Precup, D.; Sutton, R. S.; and Singh, S. 1998b. Theoretical results on reinforcement learning with temporally abstract options. In *Machine Learning: ECML98. 10th European Conference on Machine Learning. Proceedings*, 382–393. Springer.

Singh, S. P. 1992a. Reinforcement learning with a hierarchy of abstract models. In *Proc. of the 10th National Conf. on Artificial Intelligence*, 202–207. MIT/AAAI Press.

Singh, S. P. 1992b. Scaling reinforcement learning by learning variable temporal resolution models. In *Proc. of the 9th Intl. Conf. on Machine Learning*, 406–415. Morgan Kaufmann.

Sutton, R. S.; Precup, D.; and Singh, S. 1998. Intra-option learning about temporally abstract actions. In *Proc. of the 15th Intl. Conf. on Machine Learning*. Morgan Kaufman.

Sutton, R. S.; Precup, D.; and Singh, S. in preparation. Between MDPs and Semi-MDPs: learning, planning, and representing knowledge at multiple temporal scales.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley.

Thrun, S., and Schwartz, A. 1995. Finding structure in reinforcement learning. In *Advances in Neural Information Processing Systems 7*, 385–392. MIT Press.