Technical Report 98-74, Dept. of Computer Science, University of Massachusetts, Amherst, MA 01003. April, 1998.

Between MDPs and Semi-MDPs: Learning, Planning, and Representing Knowledge at Multiple Temporal Scales

Richard S. Sutton RICH@CS.UMASS.EDU
Doina Precup DPRECUP@CS.UMASS.EDU
University of Massachusetts, Amherst, MA 01003 USA

Satinder Singh BAVEJA@CS.COLORADO.EDU
University of Colorado, Boulder, CO 80309 USA

Abstract

Learning, planning, and representing knowledge at multiple levels of temporal abstraction are key challenges for AI. In this paper we develop an approach to these problems based on the mathematical framework of reinforcement learning and Markov decision processes (MDPs). We extend the usual notion of action to include *options*—whole courses of behavior that may be temporally extended, stochastic, and contingent on events. Examples of options include picking up an object, going to lunch, and traveling to a distant city, as well as primitive actions such as muscle twitches and joint torques. Options may be given a priori, learned by experience, or both. They may be used interchangeably with actions in a variety of planning and learning methods. The theory of semi-Markov decision processes (SMDPs) can be applied to model the consequences of options and as a basis for planning and learning methods using them. In this paper we develop these connections, building on prior work by Bradtke and Duff (1995), Parr (in prep.) and others. Our main novel results concern the interface between the MDP and SMDP levels of analysis. We show how a set of options can be altered by changing only their termination conditions to improve over SMDP methods with no additional cost. We also introduce *intra-option* temporal-difference methods that are able to learn from fragments of an option's execution. Finally, we propose a notion of subgoal which can be used to improve the options themselves. Overall, we argue that options and their models provide hitherto missing aspects of a powerful, clear, and expressive framework for representing and organizing knowledge.

1. Temporal Abstraction

To make everyday decisions, people must foresee the consequences of their possible courses of action at multiple levels of temporal abstraction. Consider a traveler deciding to undertake a journey to a distant city. To decide whether or not to go, the benefits of the trip must be weighed against the expense. Having decided to go, choices must be made at each leg, e.g., whether to fly or to drive, whether to take a taxi or to arrange a ride. Each of these steps involves foresight and decision, all the way down to the smallest of actions. For example, just to call a taxi may involve finding a telephone, dialing each digit, and the individual muscle contractions to lift the receiver to the ear. Human decision making routinely involves planning and foresight—choice among temporally-extended options—over a broad range of time scales.

In this paper we examine the nature of the knowledge needed to plan and learn at multiple levels of temporal abstraction. The principal knowledge needed is the ability to predict the consequences of different courses of action. This may seem straightforward, but it is not. It is not at all clear what we mean either by a "course of action" or, particularly, by "its consequences". One problem is that most courses of action have many consequences, with the immediate consequences different from the longer-term ones. For example, the course of action `go-to-the-library` may have the near-term consequence of being outdoors and walking, and the long-term consequence of being indoors and reading. In addition, we usually only consider courses of action for a limited but indefinite time period. An action like `wash-the-car` is most usefully executed up until the car is clean, but without specifying a particular time at which it is to stop. We seek a way of representing predictive knowledge that is:

Expressive The representation must be able to include basic kinds of commonsense knowledge such as the examples we have mentioned. In particular, it should be able to predict consequences that are temporally extended and uncertain. This criterion rules out many conventional engineering representations, such as differential equations and transition probabilities. The representation should also be able to predict the consequences of courses of action that are stochastic and contingent on subsequent observations. This rules out simple sequences of action with a deterministically known outcome, such as conventional macro-operators.

Clear The representation should be clear, explicit, and grounded in primitive observations and actions. Ideally it would be expressed in a formal mathematical language. Any predictions made should be testable simply by comparing them against data: no human interpretation should be necessary. This criterion rules out conventional AI representations with ungrounded symbols. For example, "Tweety is a bird" relies on people to understand "Tweety," "Bird," and "is-a"; none of these has a clear interpretation in terms of observables. A related criterion is that the representation should be learnable. Only a representation that is clear and directly testable from observables is likely to be learnable. A clear representation need not be unambiguous. For example, it could predict that one of two events will occur at a particular time, but not specify which of them will occur.

Suitable for Planning A representation of knowledge must be suitable for how it will be used as part of planning and decision-making. In particular, the representation should enable interrelating and intermixing knowledge at different levels of temporal abstraction.

It should be clear that we are addressing a fundamental question of AI: how should an intelligent agent represent its knowledge of the world? We are interested here in the underlying *semantics* of the knowledge, not with its surface form. In particular, we are not concerned with the data structures of the knowledge representation, e.g., whether the knowledge is represented by neural networks or symbolic rules. Whatever data structures are used to generate the predictions, our concern is with their *meaning*, i.e., with the interpretation that we or that other parts of the system can make of the predictions. Is the meaning clear and grounded enough to be tested and learned? Do the representable meanings include the commonsense predictions we seem to use in everyday planning? Are these meanings sufficient to support effective planning?

Planning with temporally extended actions has been extensively explored in several fields. Early AI research focused on it from the point of view of abstraction in planning (e.g., Fikes, Hart, and Nilsson, 1972; Newell and Simon, 1972; Nilsson, 1973; Sacerdoti, 1974). More recently, macro-operators, qualitative modeling, and other ways of chunking action selections into units have been extensively developed (e.g., Kuipers, 1979; de Kleer and Brown, 1984; Korf, 1985, 1987; Laird, Rosenbloom and Newell, 1986; Minton, 1988; Iba, 1989; Drescher, 1991; Ruby and Kibler, 1992; Dejong, 1994; Levinson and Fuchs, 1994; Nilsson, 1994; Say and Selahattin, 1996; Brafman and Moshe, 1997; Haigh, Shewchuk, and Veloso, 1997). Roboticists and control engineers have long considered methodologies for combining and switching between independently designed controllers (e.g., Brooks, 1986; Maes, 1991; Koza and Rice, 1992; Brockett, 1993; Grossman et al., 1993; Millán, 1994; Araujo and Grupen, 1996; Colombetti, Dorigo, and Borghi, 1996; Dorigo and Colombetti, 1994; Tóth, Kovács, and Lörincz, 1995; Sastry, 1997; Rosenstein and Cohen, 1998). More recently, the topic has been taken up within the framework of MDPs and reinforcement learning (Watkins, 1989; Ring, 1991; Wixson, 1991; Schmidhuber, 1991; Mahadevan and Connell, 1992; Tenenberg, Karlsson, and Whitehead, 1992; Lin, 1993; Dayan and Hinton, 1993; Dayan, 1993; Kaelbling, 1993; Singh et al., 1994; Chrisman, 1994; Hansen, 1994; Uchibe, Asada and Hosada, 1996; Asada et al., 1996; Thrun and Schwartz, 1995; Kalmár, Szepesvári, and Lörincz, 1997, in prep.; Dietterich, 1997; Matarić, 1997; Huber and Grupen, 1997; Wiering and Schmidhuber, 1997; Parr and Russell, 1998; Drummond, 1998; Hauskrecht et al., in prep.; Meuleau, in prep.), within which we also work here. Our recent work in this area (Precup and Sutton, 1997, 1998; Precup, Sutton, and Singh, 1997, 1998; see also McGovern, Sutton, and Fagg, 1997; McGovern and Sutton, in prep.) can be viewed as a combination and generalization of Singh's hierarchical Dyna (1992a,b,c,d) and Sutton's mixture models (1995; Sutton and Pinette, 1985). In the current paper we simplify our treatment of the ideas by linking temporal abstraction to the theory of Semi-MDPs, as in Parr (in prep.), and as we discuss next.

2. Between MDPs and SMDPs

In this paper we explore the extent to which Markov decision processes (MDPs) can provide a mathematical foundation for the study of temporal abstraction and temporally extended action. MDPs have been widely used in AI in recent years to study planning and learning in stochastic environments (e.g., Barto, Bradtke, and Singh, 1995; Dean et al., 1995; Boutilier, Brafman, and Gelb, 1997; Simmons and Koenig, 1995; Geffner and Bonet, in prep.). They provide a simple formulation of the AI problem including sensation, action, stochastic cause-and-effect, and general goals formulated as reward signals. Effective learning and planning methods for MDPs have been proven in a number of significant applications (e.g., Mahade-

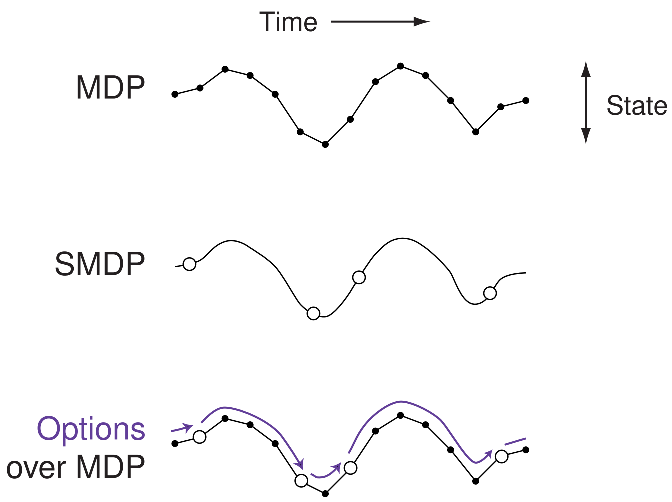

Figure 1: The state trajectory of an MDP is made up of small, discrete-time transitions, whereas that of an SMDP comprises larger, continuous-time transitions. Options enable an MDP trajectory to be analyzed at either level.

van et al., 1997; Marbach et al., 1998; Nie and Haykin, to appear; Singh and Bertsekas, 1997; Tesauro, 1995; Crites and Barto, 1996). However, conventional MDPs include only a single temporal scale of action. They are based on a discrete time step: the unitary action taken at time t affects the state and reward at time $t+1$. There is no notion of a course of action persisting over a variable period of time. As a consequence, MDP methods are unable to take advantage of the simplicities and efficiencies sometimes available at higher levels of temporal abstraction.

An alternative is to use semi-Markov decision processes (SMDPs), a special kind of MDP appropriate for modeling continuous-time discrete-event systems (e.g., see Puterman, 1994; Mahadevan et al., 1997). The actions in SMDPs are permitted to take variable amounts of time and are intended to model temporally-extended courses of action. The existing theory of SMDPs also specifies how to model the results of these actions and how to plan with them. However, existing SMDP work is limited because the temporally extended actions are treated as indivisible and unknown units. There is no attempt in SMDP theory to look *inside* the temporally extended actions, to examine or modify how they are comprised of lower-level actions. As we have tried to suggest above, this is the essence of analyzing temporally abstract actions in AI applications: goal directed behavior involves multiple overlapping scales at which decisions are made and modified.

In this paper we explore what might be viewed as a middle ground between MDPs and SMDPs. The base problem we consider is that of a conventional discrete-time MDP, but we also consider courses of action within the MDP whose results are state transitions of extended and variable duration. We use the term *options* for these courses of action, which include primitive actions as a special case. 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. The SMDP actions (the options) are no longer black boxes, but policies in the base MDP which can be examined, changed, learned, and planned in their own right. This is what we see as the essential insight of the current work, as the key that enables new results of relevance to AI.

The first part of this paper (Sections 3-5) develops the formal machinery for options as temporally extended actions equivalent to SMDP actions within a base MDP. We define new value functions and Bellman equations for this case, but most of the results are simple applications of existing SMDP theory or of existing reinforcement learning methods for SMDPs. The primary appeal of our formalization is that it enables multi-step options to be treated identically to primitive actions in existing planning and learning methods. In particular, the consequences of multi-step options can be modeled just as SMDP actions are modeled, and the models can be used in existing MDP planning methods interchangeably with models of primitive MDP actions.

The second part of the paper introduces several ways of going beyond an SMDP analysis of options to change or learn their internal structure in terms of the MDP. The first issue we consider is that of effectively combining a given set of policies into a single overall policy. For example, a robot may have pre-designed controllers for servoing joints to positions, picking up objects, and visual search, but still face a difficult problem of how to coordinate and switch between these behaviors (e.g., Mahadevan and Connell, 1992; Matarić, 1997; Uchibe et al., 1996; Sastry, 1997; Maes and Brooks, 1990; Koza and Rice, 1992; Dorigo and Colombetti, 1994; Kalmár et al., 1997, in prep). The second issue we consider is that of *intra-option learning*—looking inside options to learn simultaneously about all options consistent with each fragment of experience. Finally, we define a notion of subgoal that can be used to shape options and create new ones.

3. Reinforcement Learning (MDP) Framework

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, $a_t \in \mathcal{A}_{s_t}$. 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}. It is notationaly convenient to suppress the differences in available actions across states whenever possible; we let $\mathcal{A} = \bigcup_{s \in \mathcal{S}} \mathcal{A}_s$ denote the union of the action sets. If $\mathcal{S}$ and $\mathcal{A}$, are finite, then 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 $a \in \mathcal{A}$ (it is understood here that $p^a_{ss'} = 0$ for $a \notin \mathcal{A}_s$). 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:

$$\begin{aligned} V^\pi(s) &= E\left\{r_{t+1} + \gamma r_{t+1} + \gamma^2 r_{t+1} + \cdots \;\middle|\; s_t = s, \pi\right\} && (1)\\ &= E\left\{r_{t+1} + \gamma V^\pi(s_{t+1}) \;\middle|\; s_t = s, \pi\right\} \\ &= \sum_{a \in \mathcal{A}_s} \pi(s, a) \left[r^a_s + \gamma \sum_{s'} p^a_{ss'} V^\pi(s')\right], && (2) \end{aligned}$$

where $\pi(s, a)$ is the probability with which the policy π chooses action $a \in \mathcal{A}_s$ in state s, and $\gamma \in [0, 1]$ is a *discount-rate* parameter. This quantity, $V^\pi(s)$, is called the *value* of state s under policy π, and V^π is called the *state-value function* for π. The *optimal* state-value function gives the value of a state under an optimal policy:

$$\begin{aligned} V^*(s) &= \max_\pi V^\pi(s) && (3)\\ &= \max_{a \in \mathcal{A}_s} E\left\{r_{t+1} + \gamma V^*(s_{t+1}) \;\middle|\; s_t = s, a_t = a\right\} \\ &= \max_{a \in \mathcal{A}_s} \left[r^a_s + \gamma \sum_{s'} p^a_{ss'} V^*(s')\right]. && (4) \end{aligned}$$

Any policy that achieves the maximum in (3) is by definition an optimal policy. Thus, given V^*, an optimal policy is easily formed by choosing in each state s any action that achieves the maximum in (4). Planning in reinforcement learning refers to the use of models of the environment to compute value functions and thereby to optimize or improve policies. Particularly useful in this regard are Bellman equations, such as (2) and (4), which recursively relate value functions to themselves. If we treat the values, $V^\pi(s)$ or $V^*(s)$, as unknowns, then a set of Bellman equations, for all $s \in \mathcal{S}$, forms a system of equations whose unique solution is in fact V^π or V^* as given by (1) or (3). This fact is key to the way in which all temporal-difference and dynamic programming methods estimate value functions.

Particularly important for learning methods is a parallel set of value functions and Bellman equations for state–action pairs rather than for states. The value of taking action a in state s under policy π, denoted $Q^\pi(s, a)$, is the expected discounted future reward starting in s, taking a, and henceforth following π:

$$Q^\pi(s, a) = E\left\{r_{t+1} + \gamma r_{t+1} + \gamma^2 r_{t+1} + \cdots \;\middle|\; s_t = s, a_t = a, \pi\right\}$$

$$= r_s^a + \gamma \sum_{s'} p_{ss'}^a V^\pi(s')$$
$$= r_s^a + \gamma \sum_{s'} p_{ss'}^a \sum_{a'} \pi(s, a') Q^\pi(s', a').$$

This is known as the *action-value function* for policy π. The *optimal* action-value function is

$$\begin{aligned} Q^*(s,a) &= \max_\pi Q^\pi(s,a) \\ &= r_s^a + \gamma \sum_{s'} p_{ss'}^a \max_{a'} Q^*(s', a'). \end{aligned}$$

Finally, many tasks are episodic in nature, involving repeated trials, or *episodes*, each ending with a reset to a standard state or state distribution. In these *episodic tasks*, we include a single special *terminal state*, arrival in which terminates the current episode. The set of regular states plus the terminal state (if there is one) is denoted $\mathcal{S}^+$. Thus, the s' in $p_{ss'}^a$ in general ranges over the set $\mathcal{S}^+$ rather than just $\mathcal{S}$ as stated earlier. In an episodic task, values are defined by the expected cumulative reward up until termination rather than over the infinite future (or, equivalently, we can consider the terminal state to transition to itself forever with a reward of zero).

4. Options

We use the term *options* for our generalization of primitive actions to include temporally extended courses of action. Options consist of three components: a policy $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0,1]$, a termination condition $\beta : \mathcal{S}^+ \mapsto [0,1]$, and an input set $\mathcal{I} \subseteq \mathcal{S}$. 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, if the option taken in state s_t is *Markov*, then the next action a_t is selected according to the probability distribution $\pi(s, \cdot)$. 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.[1] When the option terminates, then the agent has the opportunity to select another option. For example, an option named `open-the-door` might consist of a policy for reaching, grasping and turning the door knob, a termination condition for recognizing that the door has been opened, and an input set restricting consideration of `open-the-door` to states in which a door is present. In episodic tasks, termination of an episode also terminates the current option (i.e., β maps the terminal state to 1 in all options).

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 need 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

1. The termination condition β plays a role similar to the β in β-models (Sutton, 1995), but with an opposite sense. That is, $\beta(s)$ in this paper corresponds to $1 - \beta(s)$ in that earlier paper.

is within sight. The termination condition β could be defined to be 1 outside of $\mathcal{I}$ and when the robot is successfully docked. A subpolicy for servoing a robot arm to a particular joint configuration could similarly have a set of allowed starting states, a controller to be applied to them, and a termination condition indicating that either the target configuration had been reached within some tolerance or that some unexpected event had taken the subpolicy outside its domain of application. For Markov options 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 need only be defined over $\mathcal{I}$ rather than over all of $\mathcal{S}$.

Sometimes it is useful for options to "timeout," to terminate after some period of time has elapsed even if they have failed to reach any particular state. Unfortunately, this is not possible with Markov options because their termination decisions are made solely on the basis of the current state, not on how long the option has been executing. To handle this and other cases of interest we consider a generalization to *semi-Markov* options, in which policies and termination conditions may make their choices dependent on all prior events since the option was initiated. In general, an option is initiated at some time, say t, determines the actions selected for some number of steps, say k, and then terminates in s_{t+k}. At each intermediate time T, $t \leq T < t + k$, the decisions of a Markov option may depend only on s_T, whereas the decisions of a semi-Markov option may depend on the entire sequence $s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}, \ldots, r_T, s_T$, but not on events prior to s_t (or after s_T). We call this sequence the *history* from t to T and denote it by h_{tT}. We denote the set of all histories by Ω. In semi-Markov options, the policy and termination condition are functions of possible histories, that is, they are $\pi : \Omega \times \mathcal{A} \mapsto [0, 1]$ and $\beta : \Omega \mapsto [0, 1]$. The semi-Markov case is also useful for cases in which options use a more detailed state representation than is available to the policy that selects the options.

Given a set of options, their input sets implicitly define a set of available options $\mathcal{O}_s$ for each state $s \in \mathcal{S}$. These $\mathcal{O}_s$ are much like the sets of available actions, $\mathcal{A}_s$. We can unify these two kinds of sets by noting that actions can be considered a special case of options. Each action a corresponds to an option that is available whenever a is available ($\mathcal{I} = \{s : a \in \mathcal{A}_s\}$), that always lasts exactly one step ($\beta(s) = 1,\ \forall s \in \mathcal{S}$), and that selects a everywhere ($\pi(s, a) = 1,\ \forall s \in \mathcal{I}$). Thus, we can consider the agent's choice at each time to be entirely among options, some of which persist for a single time step, others which are more temporally extended. The former we refer to as *one-step* or *primitive* options and the latter as *multi-step* options. Just as in the case of actions, it is convenient to notationaly suppress the differences in available options across states. We let $\mathcal{O} = \bigcup_{s \in \mathcal{S}} \mathcal{O}_s$ denote the set of all available options.

Our 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 sequences of them in much the same way as we consider sequences of actions. We can consider policies that select options instead of primitive actions, and we can model the consequences of selecting an option much as we model the results of an action. Let us consider each of these in turn.

Given any two options a and b, we can consider taking them in sequence, that is, we can consider first taking a until it terminates, and then b until it terminates (or omitting b

altogether if a terminates in a state outside of b's input set). We say that the two options are *composed* to yield a new option, denoted ab, corresponding to this way of behaving. The composition of two Markov options will in general be semi-Markov, not Markov, because actions are chosen differently before and after the first option terminates. The composition of two semi-Markov options is always another semi-Markov option. Because actions are special cases of options, we can also compose them, producing a deterministic action sequence, in other words, a classical macro-operator.

More interesting are *policies over options*. When initiated in a state s_t, the Markov policy over options $\mu : \mathcal{S} \times \mathcal{O} \mapsto [0,1]$ selects an option $o \in \mathcal{O}_s$ 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 conventional policy over actions, or *flat policy*, $\pi = flat(\mu)$. Henceforth we use the unqualified term *policy* for policies over options, which include flat policies as a special case. Note that even if a policy is Markov and all of the options it selects are Markov, the corresponding flat policy is unlikely to be Markov if any of the options are multi-step (temporally extended). The action selected by the flat policy in state s_T depends not just on s_T but on the option being followed at that time, and this depends stochastically on the entire history h_{tT} since the policy was initiated at time t. By analogy to semi-Markov options, we call policies that depend on histories in this way *semi-Markov* policies.[2]

Our definitions of state values and action values can be generalized to apply to general policies and options. First we define the value of a state $s \in \mathcal{S}$ under a semi-Markov flat policy π as the expected return if the policy is started in s:

$$V^\pi(s) \stackrel{\text{def}}{=} E\left\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \;\middle|\; \mathcal{E}(\pi, s, t)\right\},$$

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 μ can then be defined as the value of the state under the corresponding flat policy: $V^\mu(s) \stackrel{\text{def}}{=} V^{flat(\mu)}(s)$, for all $s \in \mathcal{S}$.

It is natural to generalize action-value functions to *option*-value functions. We define $Q^\mu(s, o)$, the value of taking option o in state $s \in \mathcal{I}$ under policy μ, as

$$Q^\mu(s, o) \stackrel{\text{def}}{=} E\left\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \;\middle|\; \mathcal{E}(o\mu, s, t)\right\},$$

where $o\mu$, the *composition* of o and μ, denotes the semi-Markov policy that first follows o until it terminates and then initiates μ in the resultant state.

5. SMDP (Option-to-Option) Methods

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, any MDP with

2. This and other similarities suggest that the concepts of policy and option can be unified. In such a unification, options would select other options, and thus arbitrary hierarchical structures would be permitted. Although this appears straightforward, for simplicity we restrict ourselves in this paper to just two levels: policies that select options, and options that select actions.

a fixed set of options *is* an SMDP, as we state formally below. This theorem is not really a result, but a simple observation that follows more or less immediately from definitions. We present it as a theorem to highlight it and state explicitly its conditions and consequences:

Theorem 1 (MDP + Options = SMDP) *For any MDP, and any set of options defined on that MDP, the decision process that selects among those options, executing each to termination, is an SMDP.*

Proof: (Sketch) An SMDP consists of 1) a set of states, 2) a set of actions, 3) for each pair of state and action, an expected cumulative discounted reward, and 4) a well-defined joint distribution of the next state and transit time. In our case, the set of states is $\mathcal{S}$, and the set of actions is just the set of options. The expected reward and the next-state and transit-time distributions are defined for each state and option by the MDP and by the option's policy and termination condition, π and β. These expectations and distributions are well defined because the MDP is Markov and the options are semi-Markov; thus the next state, reward, and time are dependent only on the option and the state in which it was initiated. The transit times of options are always discrete, but this is simply a special case of the arbitrary real intervals permitted in SMDPs. $\diamond$

The relationship between MDPs, options, and SMDPs provides a basis for the theory of planning and learning methods with options. In later sections we discuss the limitations of this theory due to its treatment of options as indivisible units without internal structure, but in this section we focus on establishing the benefits and assurances that it provides. We establish theoretical foundations and then survey SMDP methods for planning and learning with options. Although our formalism is slightly different, these results are in essence taken or adapted from prior work (including classical SMDP work and Singh, 1992a,b,c,d; Bradtke and Duff, 1995; Sutton, 1995; Precup and Sutton, 1997, 1998; Precup, Sutton, and Singh, 1997, 1998; Parr and Russell, 1998; McGovern, Sutton, and Fagg, 1997; Parr, in prep.). A result very similar to Theorem 1 was proved in detail by Parr (in prep.). In the sections following this one we present new methods that improve over SMDP methods.

Planning with options requires a model of their consequences. Fortunately, the appropriate form of model for options, analogous to the r_s^a and $p_{ss'}^a$ 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 any state $s \in \mathcal{S}$ is

$$r_s^o = E\left\{r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{k-1} r_{t+k} \;\middle|\; \mathcal{E}(o,s,t)\right\}, \tag{5}$$

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

$$\begin{aligned} p_{ss'}^o &= \sum_{j=1}^{\infty} \gamma^j \Pr\{s_{t+k} = s', k = j \mid \mathcal{E}(o,s,t)\} \\ &= E\Big\{\gamma^k \delta_{s' s_{t+k}} \mid \mathcal{E}(o,s,t)\Big\}, \end{aligned} \tag{6}$$

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, $p^o_{ss'}$ 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, 1997, 1998) because it describes the outcome of an option not at a single time but at potentially many different times, appropriately combined.[3]

Using multi-time models we can write Bellman equations for general policies and options. For any Markov policy μ, the state-value function can be written

$$V^\mu(s) = E\left\{r_{t+1} + \cdots + \gamma^{k-1} r_{t+k} + \gamma^k V^\mu(s_{t+k}) \;\middle|\; \mathcal{E}(\mu, s, t)\right\},$$

where k is the duration of the first option selected by μ,

$$= \sum_{o \in \mathcal{O}_s} \mu(s,o) \left[r^o_s + \sum_{s'} p^o_{ss'} V^\mu(s') \right], \tag{7}$$

which is a Bellman equation analogous to (2). The corresponding Bellman equation for the value of an option o in state $s \in \mathcal{I}$ is

$$\begin{aligned} Q^\mu(s,o) &= E\left\{r_{t+1} + \cdots + \gamma^{k-1} r_{t+k} + \gamma^k V^\mu(s_{t+k}) \;\middle|\; \mathcal{E}(o, s, t)\right\} \\ &= E\left\{r_{t+1} + \cdots + \gamma^{k-1} r_{t+k} + \gamma^k \sum_{o' \in \mathcal{O}_s} \mu(s_{t+k}, o') Q^\mu(s_{t+k}, o') \;\middle|\; \mathcal{E}(o, s, t)\right\} \\ &= r^o_s + \sum_{s'} p^o_{ss'} \sum_{o' \in \mathcal{O}_s} \mu(s', o') Q^\mu(s', o'). \end{aligned} \tag{8}$$

Note that all these equations specialize to those given earlier in the special case in which μ is a conventional policy and o is a conventional action. Also note that $Q^\mu(s,o) = V^{o\mu}(s)$.

Finally, there are generalizations of *optimal* value functions and *optimal* Bellman equations to options and to policies over options. Of course the conventional optimal value functions V^* and Q^* are not affected by the introduction of options; one can ultimately do just as well with primitive actions as one can with options. Nevertheless, it is interesting to know how well one can do with a restricted set of options that does not include all the actions. For example, in planning one might first consider only high-level options in order to find an approximate plan quickly. Let us denote the 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

$$\begin{aligned} V^*_{\mathcal{O}}(s) &\stackrel{\text{def}}{=} \max_{\mu \in \Pi(\mathcal{O})} V^\mu(s) \\ &= \max_{o \in \mathcal{O}_s} E\left\{r_{t+1} + \cdots + \gamma^{k-1} r_{t+k} + \gamma^k V^*_{\mathcal{O}}(s_{t+k}) \;\middle|\; \mathcal{E}(o, s, t)\right\}, \end{aligned}$$

where k is the duration of o when taken in s_t,

3. Note that the definition of state predictions in multi-time models differs slightly from that given earlier for primitive actions. Under the new definition, the model of transition from state s to s' for primitive action a is not simply the corresponding transition probability, but the transition probability *times* γ. Henceforth we use the new definition given by (6).

$$= \max_{o \in \mathcal{O}_s} \left[r_s^o + \sum_{s'} p_{ss'}^o V_{\mathcal{O}}^*(s') \right] \tag{9}$$

$$= \max_{o \in \mathcal{O}_s} E\left\{ r + \gamma^k V_{\mathcal{O}}^*(s') \;\Big|\; \mathcal{E}(o,s) \right\}, \tag{10}$$

where $\mathcal{E}(o,s)$ denotes option o being initiated in state s. Conditional on this event are the usual random variables: s' is the state in which o terminates, r is the cumulative discounted reward along the way, and k is the number of time steps elapsing between s and s'. The value functions and Bellman equations for optimal option values are

$$\begin{aligned}
Q_{\mathcal{O}}^*(s,o) &\stackrel{\text{def}}{=} \max_{\mu \in \Pi(\mathcal{O})} Q^\mu(s,o) \\
&= E\left\{ r_{t+1} + \cdots + \gamma^{k-1} r_{t+k} + \gamma^k V_{\mathcal{O}}^*(s_{t+k}) \;\Big|\; \mathcal{E}(o,s,t) \right\},
\end{aligned}$$

where k is the duration of o from s_t,

$$\begin{aligned}
&= E\left\{ r_{t+1} + \cdots + \gamma^{k-1} r_{t+k} + \gamma^k \max_{o' \in \mathcal{O}_{s_{t+k}}} Q_{\mathcal{O}}^*(s_{t+k}, o') \;\Big|\; \mathcal{E}(o,s,t) \right\}, \\
&= r_s^o + \sum_{s'} p_{ss'}^o \max_{o' \in \mathcal{O}_{s_{t+k}}} Q_{\mathcal{O}}^*(s', o') \qquad (11) \\
&= E\left\{ r + \gamma^k \max_{o' \in \mathcal{O}_{s_{t+k}}} Q_{\mathcal{O}}^*(s', o') \;\Big|\; \mathcal{E}(o,s) \right\},
\end{aligned}$$

where r, k, and s' are again the reward, number of steps, and next state due to taking $o \in \mathcal{O}_s$.

Given a set of options, $\mathcal{O}$, 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 optimal policies can be formed by choosing in any proportion among the maximizing options in (9) or (10). Or, if $Q_{\mathcal{O}}^*$ is known, then optimal policies can be found without a model by choosing in each state s in any proportion among the options o for which $Q_{\mathcal{O}}^*(s,o) = \max_{o'} Q_{\mathcal{O}}^*(s,o')$. In this way, computing approximations to $V_{\mathcal{O}}^*$ or $Q_{\mathcal{O}}^*$ become key goals of planning and learning methods with options.

5.1 SMDP Planning

With these definitions, an MDP together with the set of options $\mathcal{O}$ formally comprises an SMDP, and standard SMDP methods and results apply. Each of the Bellman equations for options, (7), (8), (9), and (11), defines a system of equations whose unique solution is the corresponding value function. These Bellman equations can be used as update rules in dynamic-programming-like planning methods for finding the value functions. Typically, solution methods for this problem maintain an approximation of $V_{\mathcal{O}}^*(s)$ or $Q_{\mathcal{O}}^*(s,o)$ for all states $s \in \mathcal{S}$ and all options $o \in \mathcal{O}_s$. For example, *synchronous value iteration* (SVI) with options initializes an approximate value function $V_0(s)$ arbitrarily and then updates it by

$$V_{k+1}(s) \leftarrow \max_{o \in \mathcal{O}_s} \left[r_s^o + \sum_{s' \in \mathcal{S}^+} p_{ss'}^o V_k(s') \right] \tag{12}$$

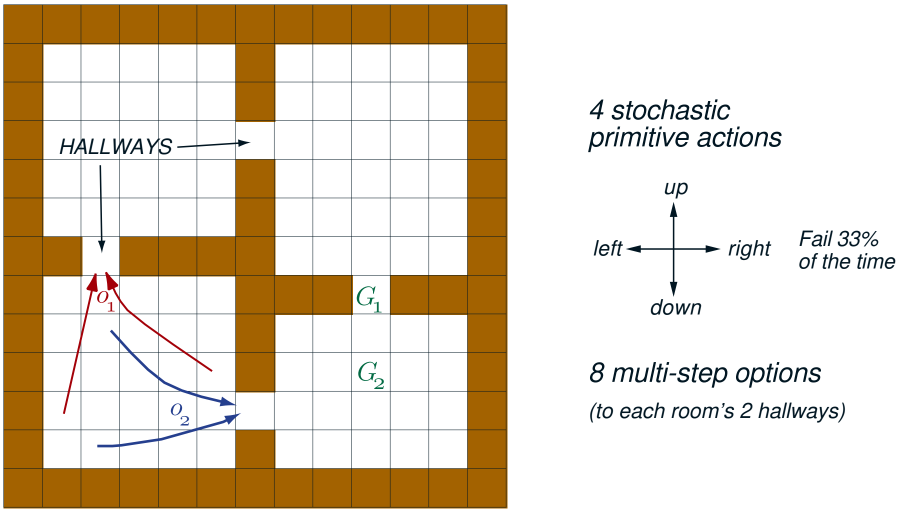

Figure 2: The rooms example is a gridworld environment with stochastic cell-to-cell actions and room-to-room hallway options. Two of the hallway options are suggested by the arrows labeled o_1 and o_2. The labels G_1 and G_2 indicate two locations used as goals in experiments described in the text.

for all $s \in \mathcal{S}$. The action-value form of SVI initializes $Q_0(s, o)$ arbitrarily and then updates it by

$$Q_{k+1}(s, o) \leftarrow r_s^o + \sum_{s' \in \mathcal{S}^+} p_{ss'}^o \max_{o' \in \mathcal{O}_{s'}} Q_k(s', o')$$

for all $s \in \mathcal{S}$ and $o \in \mathcal{O}_s$. Note that these algorithms reduce to the conventional value iteration algorithms in the special case that $\mathcal{O} = \mathcal{A}$. Standard results from SMDP theory guarantee that these processes converge for general semi-Markov options: $\lim_{k\to\infty} V_k(s) = V_{\mathcal{O}}^*(s)$ and $\lim_{k\to\infty} Q_k(s, o) = Q_{\mathcal{O}}^*(s, o)$ for all $s \in \mathcal{S}$, $o \in \mathcal{O}$, and for all sets of options $\mathcal{O}$.

The plans (policies) found using temporally abstract options are approximate in the sense that they achieve only $V_{\mathcal{O}}^*$, which is less than the maximum possible, V^*. On the other hand, if the models used to find them are correct, then they are guaranteed to achieve $V_{\mathcal{O}}^*$. We call this the *value achievement* property of planning with options. This contrasts with planning methods that abstract over state space, which generally cannot be guaranteed to achieve their planned values even if their models are correct (e.g., Dean and Lin, 1995).

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`, which have a stochastic effect. With probability 2/3, the actions cause the agent to move one cell in the corresponding direction, and with probability 1/3, the agent moves instead in one of the other three directions, each with 1/9 probability. In either case, if the movement would take the agent into a wall then the agent remains in the same cell. For now we consider a case in which rewards are zero on all state transitions.

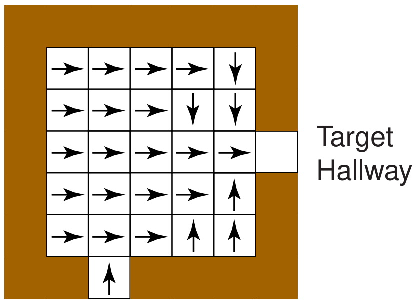

Figure 3: The policy underlying one of the eight hallway options.

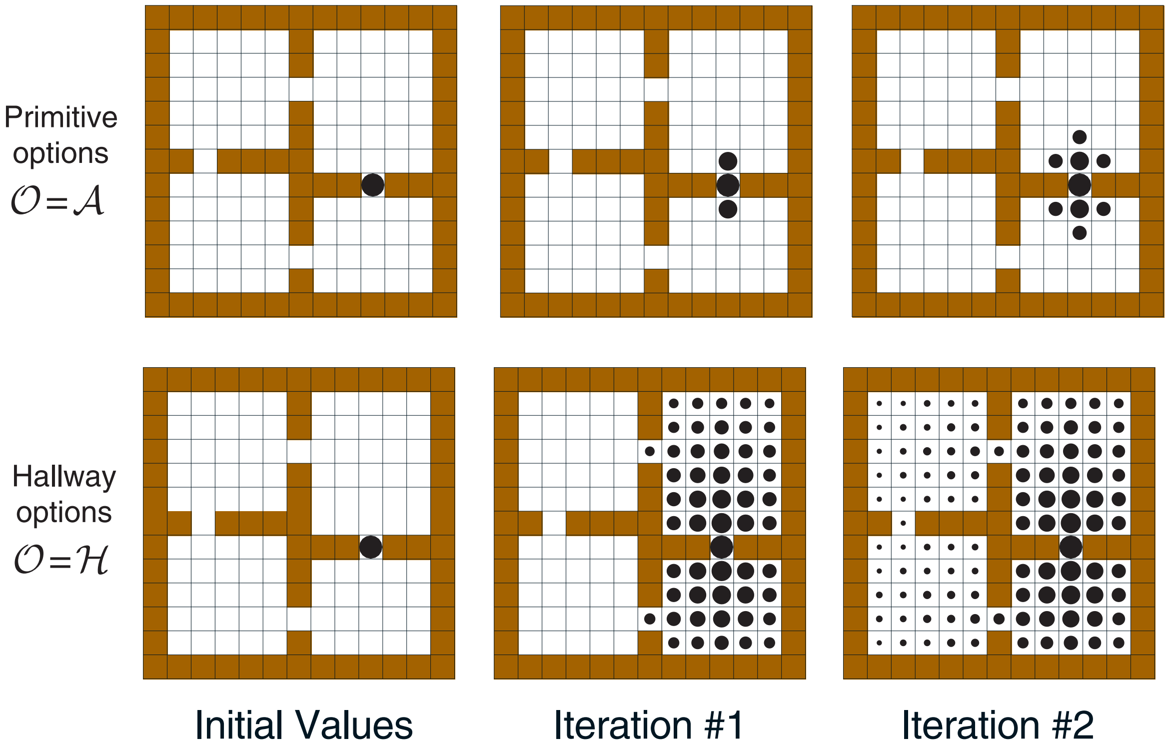

Figure 4: Value functions formed over iterations of planning by synchronous value iteration with primitive actions and with hallway options. The hallway options enabled planning to proceed room-by-room rather than cell-by-cell. The area of the disk in each cell is proportional to the estimated value of the state, where a disk that just fills a cell represents a value of 1.0.

In each of the four rooms we provide two built-in *hallway options* designed to take the agent from anywhere within the room to one of the two hallway cells leading out of the room. A hallway option's policy π follows a shortest path within the room to its target hallway while minimizing the chance of stumbling into the other hallway. For example, the policy for one hallway option is shown in Figure 3. The termination condition $\beta(s)$ for each hallway option is zero for states s within the room and 1 for states outside the room, including the hallway states. The input set $\mathcal{I}$ comprises the states within the room plus the non-target hallway state leading into the room. Note that these options are deterministic and Markov, and that an option's policy is not defined outside of its input set. We denote the set of eight hallway options by $\mathcal{H}$. For each option $o \in \mathcal{H}$, we also provide a priori its accurate model r_s^o and $p_{ss'}^o$, for all $s \in \mathcal{I}$ and $s' \in \mathcal{S}^+$, assuming there are no goal states. Note that although the transition models $p_{ss'}^o$ are nominally large (order $|\mathcal{I}| \times |\mathcal{S}^+|$), in fact they are sparse, and relatively little memory (order $|\mathcal{I}| \times 2$) is actually needed to hold the nonzero transitions from each state to the two adjacent hallway states.[4]

Now consider a sequence of planning tasks for navigating within the grid to a designated goal state, in particular, to the hallway state labeled G_1 in Figure 2. Formally, the goal state is a state from which all actions lead to the terminal state with a reward of +1. Throughout this paper we use discounting ($\gamma = 0.9$) with this task.

As a planning method, we used SVI as given by (12), with various sets of options $\mathcal{O}$. The initial value function V_0 was 0 everywhere except the goal state, which was initialized to its correct value, $V_0(G_1) = 1$, as shown in the leftmost panels of Figure 4. This figure contrasts planning with the original actions ($\mathcal{O} = \mathcal{A}$) and planning with the hallway options and not the original actions ($\mathcal{O} = \mathcal{H}$). 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 become accurately valued. Although the values continued to change by small amounts over subsequent iterations, a complete and optimal policy was known by this time. Rather than planning step-by-step, the hallway options enabled the planning to proceed at a higher level, room-by-room, and thus be much faster.

The example above is a particularly favorable case for the use of multi-step options because the goal state is a hallway, the target state of some of the options. Next we consider a case in which there is no such coincidence, in which the goal lies in the middle of a room, in the state labeled G_2 in Figure 2. The hallway options and their models were just as in the previous experiment. In this case, planning with (models of) the hallway options alone could never completely solve the task, because these take the agent only to hallways and thus never to the goal state. Figure 5 shows the value functions found over five iterations of SVI using *both* the hallway options and options corresponding to the primitive actions (i.e., using $\mathcal{O} = \mathcal{A} \cup \mathcal{H}$). In the first two iterations, accurate values were

4. The off-target hallway states are exceptions in that they have three possible outcomes: the target hallway, themselves, and the neighboring state in the off-target room.

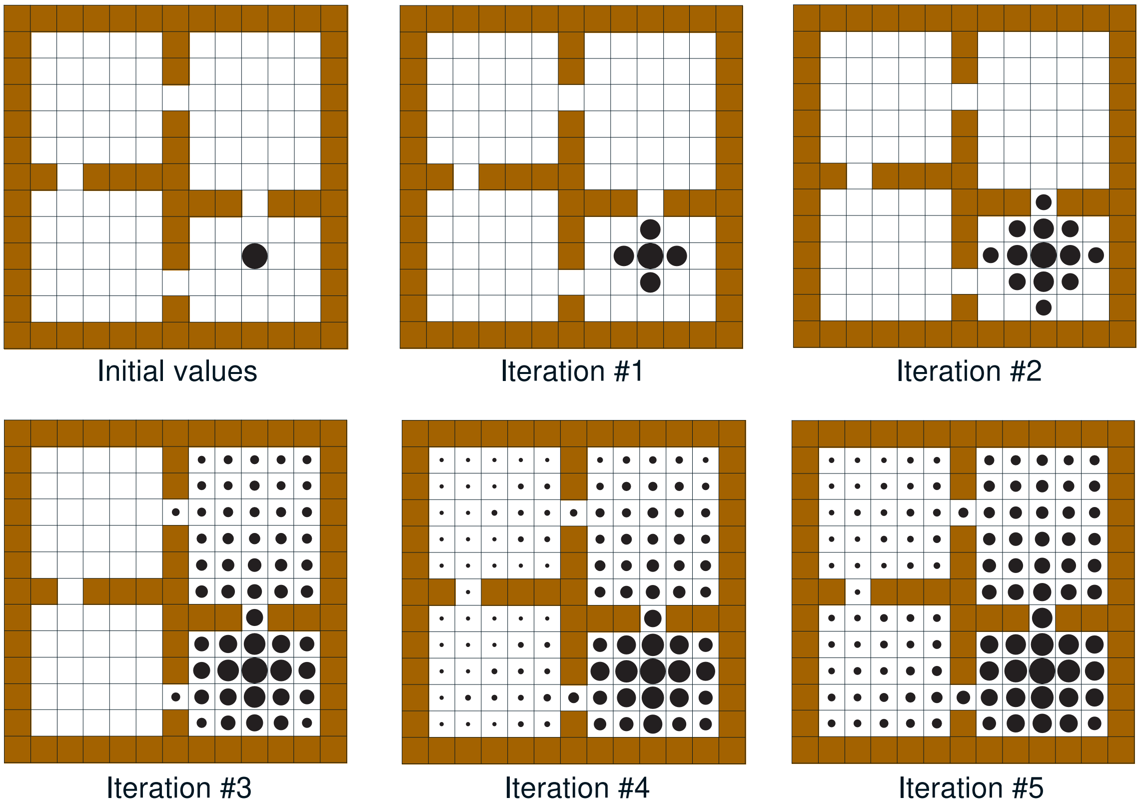

Figure 5: An example in which the goal is different from the subgoal of the hallway options. Planning here was by SVI with options $\mathcal{O} = \mathcal{A} \cup \mathcal{H}$. Initial progress was due to the models of the primitive actions, but by the third iteration room-to-room planning dominated and greatly accelerated planning.

propagated from G_2 by one cell per iteration by the models corresponding to the primitive actions. After two iterations, however, the first hallway state was reached, and subsequently room-to-room planning using the temporally extended hallway options dominated. Note how the lower-left most state was given a nonzero value during iteration three. This value corresponds to the plan of first going to the hallway state above and then down to the goal; it was overwritten by a larger value corresponding to a more direct route to the goal in the next iteration. Because of the options, a close approximation to the correct value function was found everywhere by the fourth iteration; without them only the states within three steps of the goal would have been given non-zero values by this time.

We have used SVI in this example because it is a particularly simple planning method which makes the potential advantage of multi-step options particularly clear. In large problems, SVI is impractical because the number of states is too large to complete many iterations, often not even one. In practice it is often necessary to be very selective about the states updated, the options considered, and even the next states considered. These issues are not resolved by multi-step options, but neither are they greatly aggravated. Options provide a tool for dealing with them more flexibly. Planning with options need be no more complex than planning with actions. In the SVI experiments above there were four primitive options and eight hallway options, but in each state only two hallway options needed to be considered. In addition, the models of the primitive actions generate four possible successors with non-zero probability whereas the multi-step options generate only two. Thus planning with the multi-step options was actually computationally cheaper than conventional SVI in this case. In the second experiment this was not the case, but still the use of multi-step options did not greatly increase the computational costs.

5.2 SMDP Value Learning

The problem of finding an optimal policy over a set of options $\mathcal{O}$ can also be addressed by learning methods. Because the MDP augmented by the options is an SMDP, we can apply SMDP learning methods as developed by Bradtke and Duff (1995), Parr and Russell (1998; Parr, in prep.), Mahadevan et al. (1997), or McGovern, Sutton and Fagg (1997). Much as in the planning methods discussed above, each option is viewed as an indivisible, opaque unit. When the execution of option o is started in state s, we next jump to the state s' in which o terminates. Based on this experience, an approximate option-value function $Q(s,o)$ is updated. For example, the SMDP version of one-step Q-learning (Bradtke and Duff, 1995), which we call *SMDP Q-learning*, updates after each option termination by

$$Q(s,o) \leftarrow Q(s,o) + \alpha \left[r + \gamma^k \max_{a \in \mathcal{O}} Q(s',a) - Q(s,o) \right],$$

where k denotes the number of time steps elapsing between s and s', r denotes the cumulative discounted reward over this time, and it is implicit that the step-size parameter α may depend arbitrarily on the states, option, and time steps. The estimate $Q(s,o)$ converges to $Q^*_{\mathcal{O}}(s,o)$ for all $s \in \mathcal{S}$ and $o \in \mathcal{O}$ under conditions similar to those for conventional Q-learning (Parr, in prep.), from which it is easy to determine an optimal policy as described earlier.

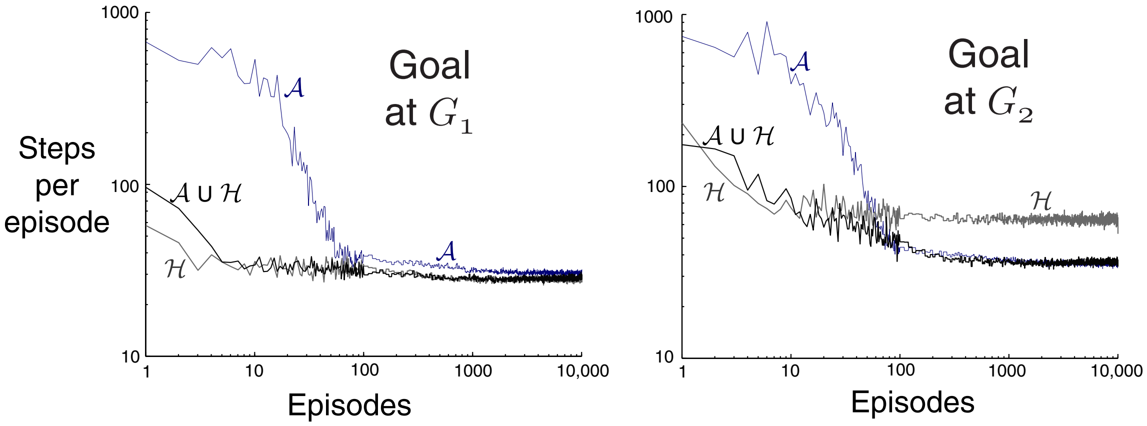

Figure 6: Learning curves for SMDP Q-learning in the rooms example with various goals and sets of options. After 100 episodes, the data points are averages over bins of 10 episodes to make the trends clearer. The step size parameter was optimized to the nearest power of 2 for each goal and set of options. The results shown used $\alpha = \frac{1}{8}$ in all cases except that with $\mathcal{O} = \mathcal{H}$ and G_1 ($\alpha = \frac{1}{16}$) and that with $\mathcal{O} = \mathcal{A} \cup \mathcal{H}$ and G_2 ($\alpha = \frac{1}{4}$).

As an illustration, we applied SMDP Q-learning to the rooms example (Figure 2) with the goal at G_1 and at G_2. As in the case of planning, we used three different sets of options, $\mathcal{A}$, $\mathcal{H}$, and $\mathcal{A} \cup \mathcal{H}$. In all cases, options were selected from the set according to an ϵ*-greedy* method. That is, given the current estimates $Q(s, o)$, let $o^* = \arg\max_{o \in \mathcal{O}_s} Q(s, o)$ denote the best valued action (with ties broken randomly). Then the policy used to select options was

$$\mu(s, o) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|\mathcal{O}_s|} & \text{if } o = o^* \\ \frac{\epsilon}{|\mathcal{O}_s|} & \text{otherwise,} \end{cases}$$

for all $s \in \mathcal{S}$ and $o \in \mathcal{O}$, where $\mathcal{O}$ is one of $\mathcal{A}$, $\mathcal{H}$, or $\mathcal{A} \cup \mathcal{H}$. The probability of random action, ϵ, was set at 0.1 in all cases. The initial state of each trial was in the upper-left corner. Figure 6 shows learning curves for both goals and all sets of options. In all cases, multi-step options caused the goal to be reached much more quickly, even on the very first trial. With the goal at G_1, these methods maintained an advantage over conventional Q-learning throughout the experiment, presumably because they did less exploration. The results were similar with the goal at G_2, except that the $\mathcal{H}$ method performed worse than the others in the long term. This is because the best solution requires several steps of primitive actions (the hallway options alone find the best solution running between hallways that sometimes stumbles upon G_2). For the same reason, the advantages of the $\mathcal{A} \cup \mathcal{H}$ method over the $\mathcal{A}$ method were also reduced.

6. Termination Improvement

SMDP methods apply to options, but only when they are treated as opaque indivisible units. More interesting and potentially more powerful methods are possible by looking inside options and by altering their internal structure. In this section we take a first step in altering options to make them more useful. This is the area where working simultaneously in terms of MDPs and SMDPs is most relevant. We can analyze options in terms of the SMDP and then use their MDP interpretation to change them and produce a new SMDP.

In particular, in this section we consider altering the termination conditions of options. Note that treating options as indivisible units, as SMDP methods do, is limiting in an unnecessary way. Once an option has been selected, such methods require that its policy be followed until the option terminates. 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, 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 $V^\mu(s)=\sum_q \mu(s,q)Q^\mu(s,q)$. If the latter is more highly valued, then why not terminate o and allow the switch? We prove below that this new way of behaving will indeed be better.

In the following theorem we characterize the new way of behaving as following a policy μ' that is the same as the original policy, μ, but over a new set of options: $\mu'(s,o')=\mu(s,o)$, for all $s\in\mathcal{S}$. Each new option o' is the same as the corresponding old option o except that it terminates whenever termination seems better than continuing according to Q^μ. In other words, the termination condition β' of o' is the same as that of o except that $\beta'(s)=1$ if $Q^\mu(s,o)<V^\mu(s)$. We call such a μ' a *termination improved policy* of μ. The theorem below generalizes on the case described above in that termination improvement is optional, not required, at each state where it could be done; this weakens the requirement that $Q^\mu(s,o)$ be completely known. A more important generalization is that the theorem applies to semi-Markov options rather than just Markov options. This is an important generalization, but can make the result seem less intuitively accessible on first reading. Fortunately, the result can be read as restricted to the Markov case simply by replacing every occurrence of "history" with "state", set of histories, Ω, with set of states, $\mathcal{S}$, etc.

Theorem 2 (Termination Improvement) *For any MDP, any set of options $\mathcal{O}$, and any Markov policy $\mu:\mathcal{S}\times\mathcal{O}\mapsto[0,1]$, define a new set of options, $\mathcal{O}'$, with a one-to-one mapping between the two option sets as follows: for every $o=\langle\mathcal{I},\pi,\beta\rangle\in\mathcal{O}$ we define a corresponding $o'=\langle\mathcal{I},\pi,\beta'\rangle\in\mathcal{O}'$, where $\beta'=\beta$ except that for any history h that ends in state s and in which $Q^\mu(h,o)<V^\mu(s)$, we may choose to set $\beta'(h)=1$. Any histories whose termination conditions are changed in this way are called* termination-improved histories. *Let policy μ' be such that for all $s\in\mathcal{S}$, and for all $o'\in\mathcal{O}'$, $\mu'(s,o')=\mu(s,o)$, where o is the option in $\mathcal{O}$ corresponding to o'. Then*

1. $V^{\mu'}(s)\geq V^\mu(s)$ *for all* $s\in\mathcal{S}$.

2. *If from state $s \in \mathcal{S}$ there is a non-zero probability of encountering a termination-improved history upon initiating μ' in s, then $V^{\mu'}(s) > V^{\mu}(s)$.*

Proof: Shortly we show that, for an arbitrary start state s, executing the option given by the termination improved policy μ' and then following policy μ thereafter is no worse than always following policy μ. In other words, we show that the following inequality holds:

$$\sum_{o'} \mu'(s,o')[r_s^{o'} + \sum_{s'} p_{ss'}^{o'} V^{\mu}(s')] \geq V^{\mu}(s) = \sum_{o} \mu(s,o)[r_s^{o} + \sum_{s'} p_{ss'}^{o} V^{\mu}(s')]. \qquad (13)$$

If this is true, then we can use it to expand the left-hand side, repeatedly replacing every occurrence of $V^{\mu}(x)$ on the left by the corresponding $\sum_{o'} \mu'(x,o')[r_x^{o'} + \sum_{x'} p_{xx'}^{o'} V^{\mu}(x')]$. In the limit, the left-hand side becomes $V^{\mu'}$, proving that $V^{\mu'} \geq V^{\mu}$.

To prove the inequality in (13), we note that for all s, $\mu'(s,o') = \mu(s,o)$, and show that

$$r_s^{o'} + \sum_{s'} p_{ss'}^{o'} V^{\mu}(s') \quad \geq \quad r_s^{o} + \sum_{s'} p_{ss'}^{o} V^{\mu}(s') \qquad (14)$$

as follows. Let Γ denote the set of all termination improved histories: $\Gamma = \{h \in \Omega : \beta(h) \neq \beta'(h)\}$. Then,

$$r_s^{o'} + \sum_{s'} p_{ss'}^{o'} V^{\mu}(s') = E\left\{r + \gamma^k V^{\mu}(s') \;\middle|\; \mathcal{E}(o',s), h_{ss'} \notin \Gamma\right\} + E\left\{r + \gamma^k V^{\mu}(s') \;\middle|\; \mathcal{E}(o',s), h_{ss'} \in \Gamma\right\},$$

where s', r, and k are the next state, cumulative reward, and number of elapsed steps following option o from s ($h_{ss'}$ is the history from s to s'). Trajectories that end because of encountering a history not in Γ never encounter a history in Γ, and therefore also occur with the same probability and expected reward upon executing option o in state s. Therefore, if we continue the trajectories that end because of encountering a history in Γ with option o until termination and thereafter follow policy μ, we get

$$\begin{aligned} & E\left\{r + \gamma^k V^{\mu}(s') \;\middle|\; \mathcal{E}(o',s), h_{ss'} \notin \Gamma\right\} \\ & \quad + \; E\left\{\beta(s')[r + \gamma^k V^{\mu}(s')] + (1-\beta(s'))[r + \gamma^k Q^{\mu}(h_{ss'},o)] \;\middle|\; \mathcal{E}(o',s), h_{ss'} \in \Gamma\right\} \\ & \qquad = \; r_s^{o} + \sum_{s'} p_{ss'}^{o} V^{\mu}(s'), \end{aligned}$$

because option o is semi-Markov. This proves (13) because for all $h_{ss'} \in \Gamma$, $Q_{\mathcal{O}}^{\mu}(h_{ss'},o) \leq V^{\mu}(s')$. Note that strict inequality holds in (14) if $Q_{\mathcal{O}}^{\mu}(h_{ss'},o) < V^{\mu}(s')$ for at least one history $h_{ss'} \in \Gamma$ that ends a trajectory generated by o' with non-zero probability. $\diamond$

As one application of this result, consider the case in which μ is an optimal policy for some given set of Markov options $\mathcal{O}$. We have already discussed how we can, by planning or learning, determine the optimal value functions $V_{\mathcal{O}}^*$ and $Q_{\mathcal{O}}^*$ and from them the optimal policy $\mu_{\mathcal{O}}^*$ that achieves them. This is indeed the best that can be done without changing $\mathcal{O}$, that is, in the SMDP defined by $\mathcal{O}$, but less than the best possible achievable in the MDP, which is $V^* = V_{\mathcal{A}}^*$. But of course we typically do not wish to work directly in the primitive options $\mathcal{A}$ because of the computational expense. The termination improvement theorem gives us a way of improving over $\mu^*_{\mathcal{O}}$ with little additional computation by stepping outside $\mathcal{O}$. That is, at each step we interrupt the current option and switch to any new option that is valued more highly according to $Q^*_{\mathcal{O}}$. Checking for such options can typically be done at vastly less expense per time step than is involved in the combinatorial process of computing $Q^*_{\mathcal{O}}$. In this sense, termination improvement gives us a nearly free improvement over any SMDP planning or learning method that computes $Q^*_{\mathcal{O}}$ as an intermediate step. Kaelbling (1993) was the first to demonstrate this effect—improved performance by interrupting a temporally extended substep based on a value function found by planning at a higher level—albeit in a more restricted setting than we consider here.

In the extreme case, we might interrupt *on every step* and switch to the greedy option—the option in that state that is most highly valued according to $Q^*_{\mathcal{O}}$. In this case, options are never followed for more than one step, and they might seem superfluous. However, the options still play a role in determining $Q^*_{\mathcal{O}}$, the basis on which the greedy switches are made, and recall that multi-step options enable $Q^*_{\mathcal{O}}$ to be found much more quickly than Q^* could (Section 5). Thus, even if multi-step options are never actually followed for more than one step, they still provide substantial advantages in computation and in our theoretical understanding.

Figure 7 shows a simple example. 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 work with these low-level actions, infinite in number, we introduce seven landmark locations in the space. For each landmark we define a controller that takes us to the landmark in a direct path (cf. Moore, 1994). 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, the discount rate γ is 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 the upper panel of Figure 7. 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 7 cuts the corners and is shorter. This is the termination-improvement 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 shown in the lower part of Figure 7.

Figure 8 shows results for an example using controllers/options with dynamics. The task here is to move a mass along one dimension from rest at position 0.0 to rest at position 2.0, again in minimum time. There is no option that takes the system all the way from

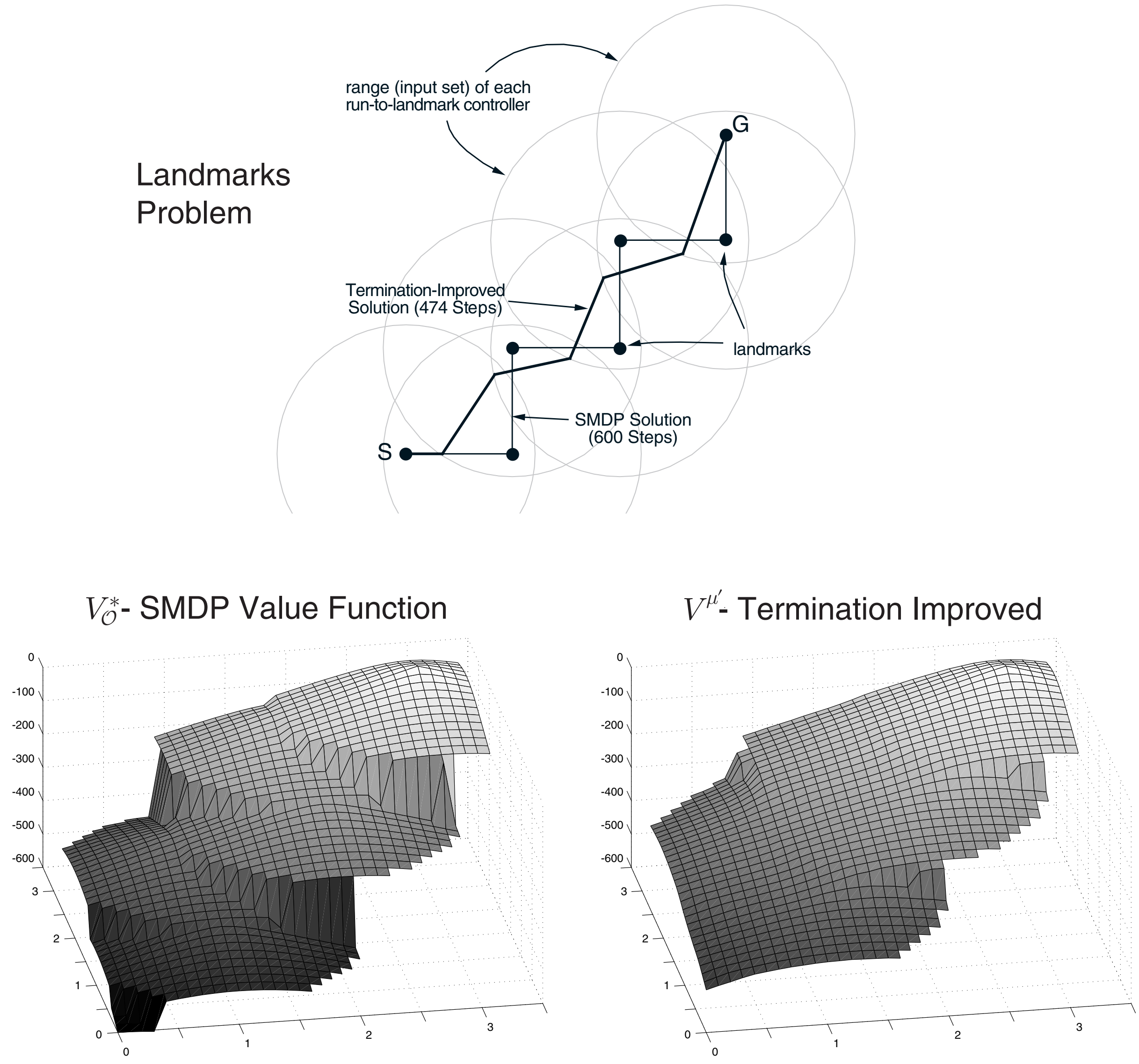

Figure 7: Termination improvement in navigating with landmark-directed controllers. The task (top) is to navigate from S to G in minimum time using options based on controllers that run each to one of seven landmarks (the black dots). The circles show the region around each landmark within which the controllers operate. The thin line shows the SMDP solution, the optimal behavior that uses only these controllers without interrupting them, and the thick line shows the corresponding termination improved solution, which cuts the corners. The lower two panels show the state-value functions for the SMDP and termination-improved solutions.

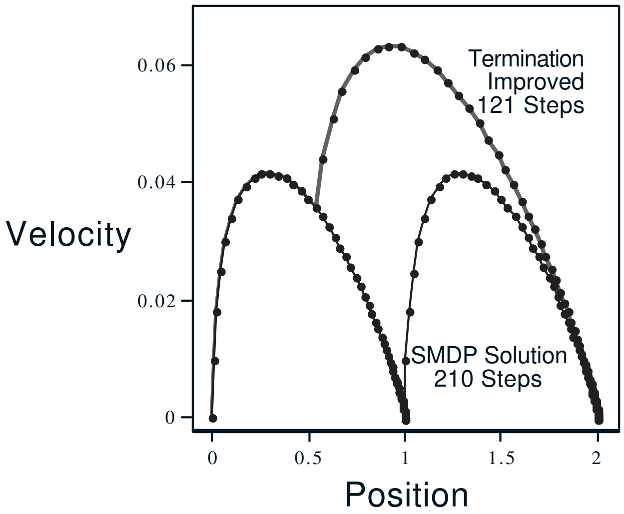

Figure 8: Phase-space plot of the SMDP and termination improved policies in a simple dynamical task. The system is a mass moving in one dimension: $x_{t+1} = x_t + \dot{x}_{t+1}$, $\dot{x}_{t+1} = \dot{x}_t + a_t - 0.175\dot{x}_t$ where x_t is the position, $\dot{x}_t$ the velocity, 0.175 a coefficient of friction, and the action a_t an applied force. Two controllers are provided as options, one that drives the position to zero velocity at $x^* = 1.0$ and the other to $x^* = 2.0$. Whichever option is being followed at time t, its target position x^* determines the action taken, according to $a_t = 0.01(x^* - x_t)$.

0.0 to 2.0, but we do have an option that takes it from 0.0 to 1.0 and another option that takes it from any position greater than 0.5 to 2.0. Both options control the system precisely to its target position and to zero velocity, terminating only when both of these are correct to within $\epsilon = 0.0001$. Using just these options, the best that can be done is to first move precisely to rest at 1.0, using the first option, then re-accelerate and move to 2.0 using the second option. This SMDP-optimal solution is much slower than the corresponding termination improved policy, as shown in Figure 8. Because of the need to slow down to near-zero velocity at 1.0, it takes over 200 time steps, whereas the improved policy takes only 121 steps.

7. Intra-Option Model Learning

The models of an option, r_s^o and $p_{ss'}^o$, can be learned from experience given knowledge of the option (i.e., of its $\mathcal{I}$, π, and β). For a semi-Markov option, the only general approach is to execute the option to termination many times in each state s, recording in each case the resultant next state s', cumulative discounted reward r, and elapsed time k. These outcomes are then averaged to approximate the expected values for r_s^o and $p_{ss'}^o$ given by (5) and (6). For example, an incremental learning rule for this could update its estimates $\hat{r}_s^o$ and $\hat{p}_{sx}^o$, for all $x \in \mathcal{S}$, after each execution of o in state s, by

$$\hat{r}_s^o = \hat{r}_s^o + \alpha[r - \hat{r}_s^o], \tag{15}$$

and

$$\hat{p}_{sx}^o = \hat{p}_{sx}^o + \alpha[\gamma^k \delta_{xs'} - \hat{p}_{sx}^o], \tag{16}$$

where the step-size parameter, α, may be constant or may depend on the state, option, and time. For example, if α is 1 divided by the number of times that o has been experienced in s, then these updates maintain the estimates as sample averages of the experienced outcomes. However the averaging is done, we call these *SMDP model-learning methods* because, like SMDP value-learning methods, they are based on jumping from initiation to termination of each option, ignoring what happens along the way. In the special case in which o is a primitive action, SMDP model-learning methods reduce to those used to learn conventional one-step models of actions.

One drawback to SMDP model-learning methods is that they improve the model of an option only when the option terminates. Because of this, they cannot be used for nonterminating options and can only be applied to one option at a time—the one option that is executing at that time. For Markov options, special temporal-difference methods can be used to learn usefully about the model of an option before the option terminates. We call these *intra-option* methods because they learn from experience within a single option. Intra-option methods can even be used to learn about the model of an option without ever executing the option, as long as some selections are made that are consistent with the option. Intra-option methods are examples of *off-policy* learning methods (Sutton and Barto, 1998) because they learn about the consequences of one policy while actually behaving according to another, potentially different policy. Intra-option methods can be used to simultaneously learn models of many different options from the same experience.

Intra-option methods were introduced by Sutton (1995), but only for a prediction problem with a single unchanging policy, not the full control case we consider here.

Just as there are Bellman equations for value functions, there are also Bellman equations for models of options. Consider the intra-option learning of the model of a Markov option $o = \langle \mathcal{I}, \pi, \beta \rangle$. The correct model of o is related to itself by

$$r_s^o = \sum_{a \in \mathcal{A}_s} \pi(s,a) E\Big\{ r + \gamma(1-\beta(s'))r_{s'}^o \Big\}$$

where r and s' are the reward and next state given that action a is taken in state s,

$$= \sum_{a \in \mathcal{A}_s} \pi(s,a) \left[r_s^a + \sum_{s'} p_{ss'}^a (1-\beta(s'))r_{s'}^o \right],$$

and

$$\begin{aligned} p_{sx}^o &= \sum_{a \in \mathcal{A}_s} \pi(s,a) \gamma E\Big\{ (1-\beta(s'))p_{s'x}^o + \beta(s')\delta_{s'x} \Big\} \\ &= \sum_{a \in \mathcal{A}_s} \pi(s,a) \sum_{s'} p_{ss'}^a (1-\beta(s'))p_{s'x}^o + \beta(s')\delta_{s'x} \end{aligned}$$

for all $s, x \in \mathcal{S}$. How can we turn these Bellman-like equations into update rules for learning the model? First consider that action a_t is taken in s_t, and that the way it was selected is consistent with $o = \langle \mathcal{I}, \pi, \beta \rangle$, that is, that a_t was selected with the distribution $\pi(s_t, \cdot)$. Then the Bellman equations above suggest the temporal-difference update rules

$$\hat{r}_{s_t}^o \leftarrow \hat{r}_{s_t}^o + \alpha \left[r_{t+1} + \gamma(1-\beta(s_{t+1}))\hat{r}_{s_{t+1}}^o - \hat{r}_{s_t}^o \right] \tag{17}$$

and

$$\hat{p}_{s_t x}^o \leftarrow \hat{p}_{s_t x}^o + \alpha \left[\gamma(1-\beta(s_{t+1}))\hat{p}_{s_{t+1}x}^o + \gamma\beta(s_{t+1})\delta_{s_{t+1}x} - \hat{p}_{s_t x}^o \right], \tag{18}$$

where $\hat{p}_{ss'}^o$ and $\hat{r}_s^o$ are the estimates of $p_{ss'}^o$ and r_s^o, respectively, and α is a positive step-size parameter. The method we call *one-step intra-option model learning* applies these updates to every option consistent with every action taken, a_t. Of course, this is just the simplest intra-option model-learning method. Others may be possible using eligibility traces and standard tricks for off-policy learning (as in Sutton, 1995).

As an illustration, consider the use of SMDP and intra-option model learning in the rooms example. As before, we assume that the eight hallway options are given, but now we assume that their models are not given and must be learned. In this experiment, the rewards were selected according to a normal probability distribution with a standard deviation of 0.1 and a mean that was different for each state–action pair. The means were selected randomly at the beginning of each run uniformly from the $[-1, 0]$ interval. Experience was generated by selecting randomly in each state among the two possible options and four possible actions, with no goal state. In the SMDP model-learning method, equations (15) and (16) were applied whenever an option terminated, whereas, in the intra-option model-learning method, equations (17) and (18) were applied on every step to all options that were consistent with the action taken on that step. In this example, all options are deterministic,

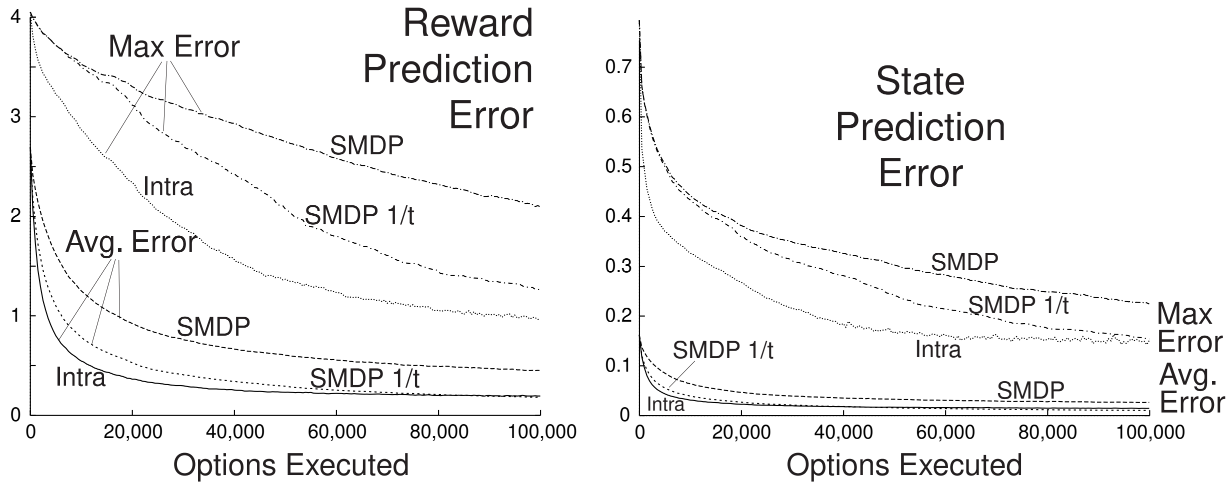

Figure 9: Learning curves for model learning by SMDP and intra-option methods. Shown are the average and maximum over $\mathcal{I}$ of the absolute errors between the learned and true models, averaged over the eight hallway options and 30 repetitions of the whole experiment. The lines labeled 'SMDP 1/t' are for the SMDP method using sample averages; the others all used $\alpha = 1/4$.

so consistency with the action selected means simply that the option would have selected that action.

For each method, we tried a range of values for the step-size parameter, $\alpha = \frac{1}{2}, \frac{1}{4}, \frac{1}{8}$, and $\frac{1}{16}$. Results are shown in Figure 9 for the value that seemed to be best for each method, which happened to be $\alpha = \frac{1}{4}$ in all cases. For the SMDP method, we also show results with the step-size parameter set such that the model estimates were sample averages, which should give the best possible performance of this method (these lines are labeled $1/t$). The figure shows the average and maximum errors over the state–option space for each method, averaged over the eight options and 30 repetitions of the experiment. As expected, the intra-option method was able to learn significantly faster than the SMDP methods.

8. Intra-Option Value Learning

We turn now to the intra-option learning of option values and thus of optimal policies over options. If the options are semi-Markov, then again the SMDP methods described in Section 5.2 are probably the only feasible methods; a semi-Markov option must be completed before it can be evaluated in any way. But if the options are Markov and we are willing to look *inside* them, then we can consider intra-option methods. Just as in the case of model learning, intra-option methods for value learning are potentially more efficient than SMDP methods because they extract more training examples from the same experience.

For example, suppose we are learning to approximate $Q^*_{\mathcal{O}}(s,o)$ and that o is Markov. Based on an execution of o from t to $t+k$, SMDP methods extract a single training example for $Q^*_{\mathcal{O}}(s,o)$. But because o is Markov, it is, in a sense, also initiated at each of the steps between t and $t+k$. The jumps from each intermediate s_i to s_{t+k} are also valid experiences with o, experiences that can be used to improve estimates of $Q^*_{\mathcal{O}}(s_i,o)$. Or consider an option that is very similar to o and which would have selected the same actions, but which would have terminated one step later, at $t+k+1$ rather than at $t+k$. Formally this is a different option, and formally it *was not executed*, yet all this experience could be used for learning relevant to it. In fact, an option can often learn something from experience that is only slightly related (occasionally selecting the same actions) to what would be generated by executing the option. This is the idea of off-policy training—to make full use of whatever experience occurs to learn as much as possible about all options irrespective of their role in generating the experience. To make the best use of experience we would like an off-policy and intra-option version of Q-learning.

It is convenient to introduce new notation for the value of a state–option pair given that the option is Markov and executing upon *arrival* in the state:

$$U^*_{\mathcal{O}}(s,o) = (1-\beta(s))Q^*_{\mathcal{O}}(s,o) + \beta(s)\max_{o'\in\mathcal{O}} Q^*_{\mathcal{O}}(s,o'),$$

Then we can write Bellman-like equations that relate $Q^*_{\mathcal{O}}(s,o)$ to expected values of $U^*_{\mathcal{O}}(s',o)$, where s' is the immediate successor to s after initiating Markov option $o = \langle \mathcal{I},\pi,\beta\rangle$ in s:

$$\begin{aligned}
Q^*_{\mathcal{O}}(s,o) &= \sum_{a\in\mathcal{A}_s} \pi(s,a) E\left\{ r + \gamma U^*_{\mathcal{O}}(s',o) \;\Big|\; s,a \right\} \\
&= \sum_{a\in\mathcal{A}_s} \pi(s,a)\left[r^a_s + \sum_{s'} p^a_{ss'} U^*_{\mathcal{O}}(s',o)\right], \qquad (19)
\end{aligned}$$

where r is the immediate reward upon arrival in s'. Now consider learning methods based on this Bellman equation. Suppose action a_t is taken in state s_t to produce next state s_{t+1} and reward r_{t+1}, and that a_t was selected in a way consistent with the Markov policy π of an option $o = \langle \mathcal{I},\pi,\beta\rangle$. That is, suppose that a_t was selected according to the distribution $\pi(s_t,\cdot)$. Then the Bellman equation above suggests applying the off-policy one-step temporal-difference update:

$$Q(s_t,o) \leftarrow Q(s_t,o) + \alpha\Big[(r_{t+1} + \gamma U(s_{t+1},o)) - Q(s_t,o)\Big], \qquad (20)$$

where

$$U(s,o) = (1-\beta(s))Q(s,o) + \beta(s)\max_{o'\in\mathcal{O}} Q(s,o').$$

The method we call *one-step intra-option Q-learning* applies this update rule to every option o consistent with every action taken, a_t. Note that the algorithm is potentially dependent on the order in which options are updated.

Theorem 3 (Convergence of intra-option Q-learning) *For any set of deterministic Markov options $\mathcal{O}$, one-step intra-option Q-learning converges w.p.1 to the optimal Q-values, $Q^*_{\mathcal{O}}$, for every option regardless of what options are executed during learning provided every primitive action gets executed in every state infinitely often.*

Proof: (Sketch) On experiencing the transition, (s, a, r', s'), for every option o that picks action a in state s, intra-option Q-learning performs the following update:

$$Q(s, o) \leftarrow Q(s, o) + \alpha(s, o)[r' + \gamma U(s', o) - Q(s, o)].$$

Let a be the action selection by deterministic Markov option $o = \langle \mathcal{I}, \pi, \beta \rangle$. Our result follows directly from Theorem 1 of Jaakkola, Jordan, and Singh (1994) and the observation that the expected value of the update operator $r' + \gamma U(s', o)$ yields a contraction, proved below:

$$\begin{aligned}
|E\{r' + \gamma U(s', o)\} - Q^*_{\mathcal{O}}(s, o)| &= |r^a_s + \sum_{s'} p^a_{ss'} U(s', o) - Q^*_{\mathcal{O}}(s, o)| \\
&= |r^a_s + \sum_{s'} p^a_{ss'} U(s', o) - r^a_s - \sum_{s'} p^a_{ss'} U^*_{\mathcal{O}}(s', o)| \\
&\leq |\sum_{s'} p^a_{ss'} \Big[(1 - \beta(s'))(Q(s', o) - Q^*_{\mathcal{O}}(s', o)) \\
&\qquad + \beta(s')(\max_{o' \in \mathcal{O}} Q(s', o') - \max_{o' \in \mathcal{O}} Q^*_{\mathcal{O}}(s', o')\Big]| \\
&\leq \sum_{s'} p^a_{ss'} \max_{s'', o''} |Q(s'', o'') - Q^*_{\mathcal{O}}(s'', o'')| \\
&\leq \gamma \max_{s'', o''} |Q(s'', o'') - Q^*_{\mathcal{O}}(s'', o'')|
\end{aligned}$$

$\diamond$

As an illustration, we applied this intra-option method to the rooms example, this time with the goal in the rightmost hallway, cell G_1 in Figure 2. Actions were selected randomly with equal probability from the four primitives. The update (20) was applied first to the primitive options, then to any of the hallway options that were consistent with the action. The hallway options were updated in clockwise order, starting from any hallways that faced up from the current state. The rewards were the same as in the experiment in the previous section. Figure 10 shows learning curves demonstrating the effective learning of option values without ever selecting the corresponding options.

Intra-option versions of other reinforcement learning methods such as Sarsa, TD(λ), and eligibility-trace versions of Sarsa and Q-learning should be straightforward, although there has been no experience with them. The intra-option Bellman equation (19) could also be used for intra-option sample-based planning.

9. Learning Options

Perhaps the most important aspect of working between MDPs and SMDPs is that the options making up the SMDP actions may be changed. We have seen one way in which this can be done by changing their termination conditions. Perhaps more fundamental than that is changing their *policies*, which we consider briefly in this section. It is natural to think of options as achieving subgoals of some kind, and to adapt each option's policy to better achieve its subgoal. For example, if the option is `open-the-door`, then it is natural

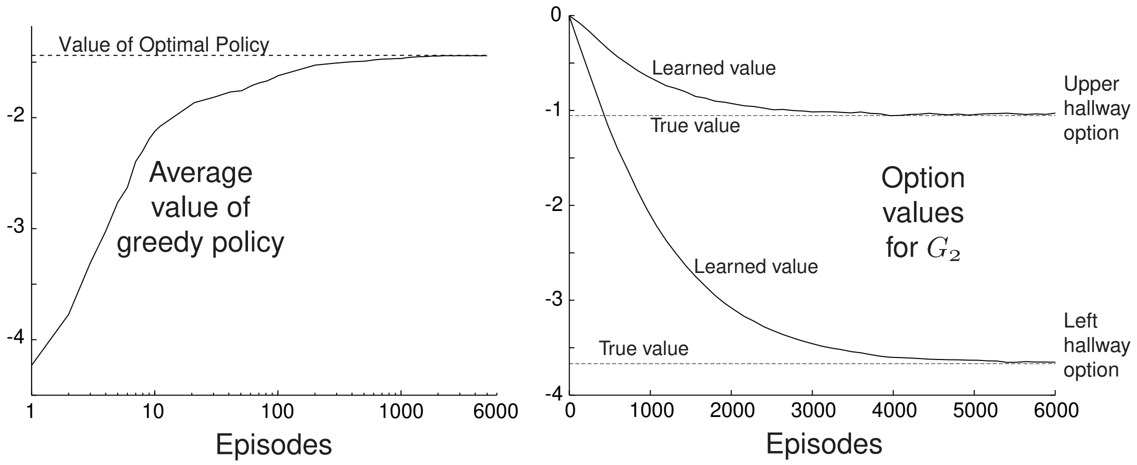

Figure 10: The learning of option values by intra-option methods without ever selecting the options. Experience was generated by selecting randomly among primitive actions, with the goal at G_1. Shown on the left is the value of the greedy policy, averaged over all states and 30 repetitions of the experiment, as compared with the value of the optimal policy. The right panel shows the learned option values from state G_2 approaching their correct values.

to adapt its policy over time to make it more effective and efficient in opening the door, which should make it more generally useful. Given subgoals for options, it is relatively straightforward to design off-policy intra-option learning methods to adapt the policies to better achieve those subgoals. For example, it may be possible to simply apply Q-learning to learn independently about each subgoal and option (as in Singh, 1992b; Thrun and Schwartz, 1995; Lin, 1993; Dorigo and Colombetti, 1994).

On the other hand, it is not clear which of the several possible ways of formulating subgoals to associate with options is the best, or even what the basis for evaluation should be. One of the important considerations is the extent to which models of options constructed to achieve one subgoal can be transferred to aid in planning the solution to another. We would like a long-lived learning agent to face a continuing series of subtasks that result in its being more and more capable. A full treatment of the transfer across subgoals probably involves developing the ideas of general hierarchical options (options that select other options), which we have avoided in this paper. Nevertheless, in this section we briefly present a simple approach to associating subgoals with options. We do this without going to the full hierarchical case, that is, we continue to consider only options that select only primitive actions. The formalization of subgoals we present here is probably not the best, but it suffices to illustrate some of the possibilities and problems that arise. A larger issue which we do not address is the source of the subgoals. We assume that the subgoals are given and focus on how options can be learned and tuned to achieve them, and on how learning toward different subgoals can aid each other.

A simple way to formulate a subgoal is by assigning a *subgoal value*, $g(s)$, to each state s in a subset of states $\mathcal{G} \subseteq \mathcal{S}$. These values indicate how desirable it is to terminate an option in each state in $\mathcal{G}$. For example, to learn a hallway option in the rooms task, the target hallway might be assigned a subgoal value of +1 while the other hallway and all states outside the room might be assigned a subgoal value of 0. Let $\mathcal{O}_g$ denote the set of options that terminate only and always in the states $\mathcal{G}$ in which g is defined (i.e., for which $\beta(s) = 0$ for $s \notin \mathcal{G}$ and $\beta(s) = 1$ for $s \in \mathcal{G}$). Given a subgoal-value function $g : \mathcal{G} \mapsto \Re$, one can define a new state-value function, denoted $V_g^o(s)$, for options $o \in \mathcal{O}_g$, as the expected value of the cumulative reward if option o is initiated in state s, plus the subgoal value $g(s')$ of the state s' in which it terminates. Similarly, we can define a new action-value function $Q_g^o(s,a) = V_g^{ao}(s)$ for actions $a \in \mathcal{A}_s$ and options $o \in \mathcal{O}_g$.

Finally, we can define *optimal* value functions for any subgoal g: $V_g^*(s) = \max_{o \in \mathcal{O}_g} V_g^o(s)$ and $Q_g^*(s,a) = \max_{o \in \mathcal{O}_g} Q_g^o(s,a)$. Finding an option that achieves these maximums (an *optimal option* for the subgoal) is then a well defined subtask. For Markov options, this subtask has Bellman equations and methods for learning and planning just as in the original task. For example, the one-step tabular Q-learning method for updating an estimate $Q_g(s_t, a_t)$ of $Q_g^*(s_t, a_t)$ is

$$Q_g(s_t, a_t) \leftarrow Q_g(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q_g(s_{t+1}, a_{t+1}) - Q_g(s_t, a_t) \right],$$

if $s_{t+1} \notin \mathcal{G}$, and

$$Q_g(s_t, a_t) \leftarrow Q_g(s_t, a_t) + \alpha \left[r_{t+1} + \gamma g(s_{t+1}) - Q_g(s_t, a_t) \right],$$

if $s_{t+1} \in \mathcal{G}$.

As a simple example, we applied this method to learn the policies of the eight hallway options in the rooms example. Each option was assigned subgoal values of +1 for the target hallway and 0 for all states outside the option's room, including the off-target hallway. The initial state was that in the upper left corner, actions were selected randomly with equal probability, and there was no goal state. The parameters were $\gamma = 0.9$ and $\alpha = 0.1$. All rewards were zero. Figure 11 shows the learned action values $Q_g(s,a)$ for each of the eight subgoals/options reliably approaching their ideal values, $Q_g^*(s,a)$.

It is interesting to note that, in general, the policies learned to achieve subgoals will depend in detail on the precise values assigned by g to the subgoal states. For example, suppose nonzero expected rewards were introduced into the rooms task, distributed uniformly between 0 and −1. Then a subgoal value of +10 (at the target hallway) results in an optimal policy that goes directly to the target hallway and away from the other hallway, as shown on the left in Figure 12, whereas a subgoal value of +1 may result in an optimal policy that goes only indirectly to the target hallway, as shown on the right in Figure 12. A roundabout path may be preferable in the latter case to avoid unusually large penalties. In the extreme it may even be optimal to head for the off-target hallway, or even to spend an infinite amount of time running into a corner and never reach any subgoal state. This is not a problem, but merely illustrates the flexibility of this subgoal formulation. For example, we may want to have two options for `open-the-door`, one of which opens the door only if it is easy to do so, for example, if is unlocked, and one which opens the door no matter

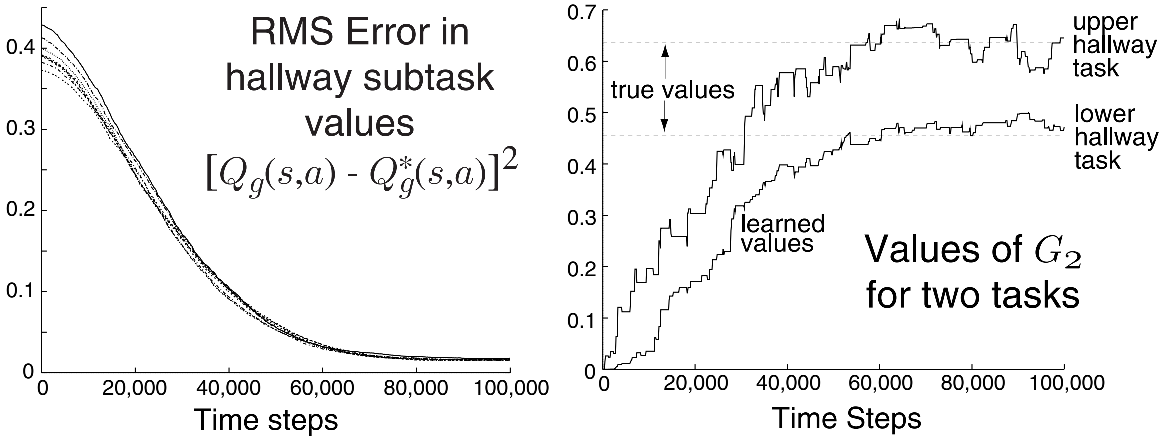

Figure 11: Learning curves for the action values of each hallway option under random behavior. Shown on the left is the error between $Q_g(s,a)$ and $Q^*_g(s,a)$ averaged over $s \in \mathcal{I}$, $a \in \mathcal{A}$, and 30 repetitions of the whole experiment. The right panel shows the individual learned values for two options at one state (maximum over the learned action values) approaching their correct values.

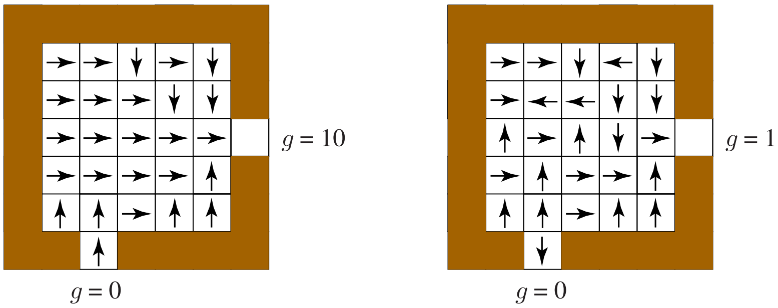

Figure 12: Two different optimal policies for options given two different subgoal values at the target hallway. A subgoal value of +10 (left) results in a more direct policy than a subgoal of +1.

what, for example, by breaking it down if need be. If we had only the first option, then we would not be able to break down the door if need be, but if we had only the second, then we would not be able to choose to open the door without committing to breaking it down if it was locked, which would greatly diminish the option's usefulness. The ability to learn and represent options for different intensities of subgoals, or different balances of outcome values, is an important flexibility.

Subgoals, options, and models of options enable interesting new possibilities for reinforcement learning agents. For example, we could present the agent with a series of tasks as subgoals, perhaps graded in difficulty. For each, the agent would be directed to find an option that achieves the subgoal and to learn a model of the option. Although the option

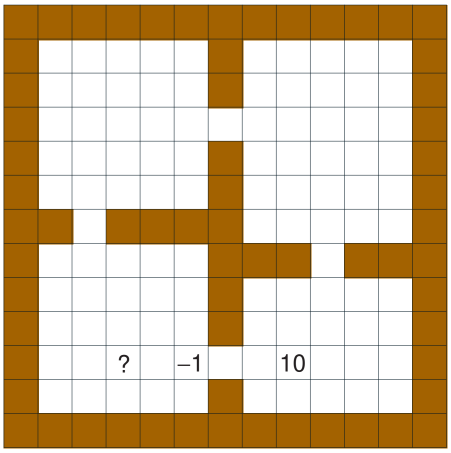

Figure 13: A subgoal to which a hallway option does *not* transfer. The option for passing from the lower-left room through to the state with subgoal value 10 no longer works because of the state with subgoal value -1. The original model of this option is overpromising with respect to the subgoal.

and model are constructed based on the task, note that they can be transferred to any other task. The option just says what to do; if behaving that way is a useful substep on another task, then it will help on that task. Similarly, the model just predicts the consequences of behaving that way; if that way of behaving is a useful substep on another task, then the model will help in planning to use that substep. As long as the model is accurate for its option it may be useful in planning the solution to another task. Singh (1992a,b,c) and Lin (1993) provide some simple examples of learning solutions to subtasks and then transferring them to help solve a new task.

On the other hand, assuring that the models of options remain accurate across changes in tasks or subgoals is far from immediate. The most severe problem arises when the new subgoal prevents the successful completion of an option whose model has previously been learned. Figure 13 illustrates the problem in a rooms example. Here we assume the options and models have already been learned, then a new subgoal is considered that assigns a high value, 10 to a state in the lower-right room but a low value, -1, to a state that must be passed through to enter that room from the lower-left room. The -1 subgoal state makes it impossible to pass between the two rooms—the subgoal considers only options that terminate in its subgoal states—and the low value of this state makes it undesirable to try. Yet the prior model indicates that it is still possible to travel from the lower-left room "through" the -1 state to the hallway state and thereby to the 10-valued state. Thus, planning with this model will lead inevitably to a highly-valued but poor policy. Such problems can arise whenever the new subgoal involves states that which may be passed through when an option is executed.

On the other hand, such problems can be detected and prevented in a number of ways. One idea is keep track of which states an option passes through and invalidate options and models that pass through subgoal states. Another idea is to alter the subgoal formulation such that subgoal states *can* be passed through: stopping in them and collecting the subgoal value is optional rather than required. Finally, note that we do not require models to be accurate, just *non-overpromising*—that is, they do not have to predict the correct outcome, just an outcome that is less than or equal to, in expected value, the correct outcome. This finesse may enable important special cases to be handled simply. For example, any new subgoal involving states $\mathcal{G}$ that all have the same subgoal value, e.g., any singleton $\mathcal{G}$, can probably be safely transferred to. The sort of problem shown in Figure 13 can never occur in such cases.

10. Conclusion

Representing knowledge flexibly at multiple levels of temporal abstraction has the potential to greatly speed planning and learning on large problems. Options and their models offer a new set of tools for realizing this potential. They offer new capabilities in each of three critical areas that we identified at the beginning of this paper. They are *clear* enough to be interpreted entirely mechanically, as we have shown by exhibiting simple procedures for executing options, learning models of them, testing the models against real events, modifying options, and creating new options given subgoals. They are more *expressive* than previous methods based on MDPs and SMDPs in that they permit multiple levels of temporal abstraction to simultaneously apply to the same system. Finally, they are explicitly designed to be *suitable for planning* using methods based on Bellman equations. Compared to conventional MDP and SMDP formulations, options provide a substantial increase in expressiveness with no loss of clarity or suitability for planning. Compared with classical AI representations, they are a substantial increase in clarity and in some aspects of expressiveness. In particular, they apply to stochastic environments, closed-loop policies, and to a more general class of goals.

The foundation for the theory of options is provided by the existing theory of Semi-MDPs. The fact that each set of options defines an SMDP provides a rich set of planning and learning methods, convergence theory, and an immediate, natural, and general way of analyzing mixtures of actions at different time scales. This theory offers a lot, but still the most interesting cases are beyond it because they involve interrupting, constructing, or otherwise decomposing options into their constituent parts. It is the intermediate ground between MDPs and SMDPs that seems richest in possibilities for new algorithms and results. In this paper we have broken this ground and touched on many of the issues, but there is far more left to be done. Key issues such as transfer between subtasks, the source of subgoals, and integration with state abstraction remain open and unclear. The connection between options and SMDPs provides only a foundation for addressing these and other issues.

Finally, although this paper has emphasized temporally extended *action*, it is interesting to note that there may be implications for temporally extended *perception* as well. It is now common to recognize that action and perception are intimately related. To see the objects in a room is not so much to label or locate them as it is to know what opportunities they afford for action: a door to open, a chair to sit on, a book to read, a person to talk to. If the temporally extended actions are modeled as options, then perhaps the model of the option corresponds well to these perceptions. Consider a robot learning to recognize its battery charger. The most useful concept for it is the set of states from which it can successfully dock with the charger. This is exactly the concept that would be produced by the model of a docking option. These kinds of action-oriented concepts are appealing because they can be tested and learned by the robot without external supervision, as we have shown in this paper.

Acknowledgements

The authors gratefully acknowledge the substantial help they have received from the colleagues who have shared their related results and ideas with us over the long period during which this paper was in preparation, especially Amy McGovern, Andy Barto, Ron Parr, Tom Dietterich, Andrew Fagg, and Manfred Huber. We also thank Leo Zelevinsky, Zsolt Kalmár, Csaba Szepesvári, András Lörincz, Paul Cohen, Robbie Moll, Mance Harmon, Sascha Engelbrecht, and Ted Perkins. This work was supported by NSF grant ECS-9511805 and grant AFOSR-F49620-96-1-0254, both to Andrew Barto and Richard Sutton. Doina Precup also acknowledges the support of the Fulbright foundation. Satinder Singh was supported by NSF grant IIS-9711753.

References

Araujo, E.G., Grupen, R.A. (1996). Learning control composition in a complex environment. *Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, pp. 333-342.

Asada, M., Noda, S., Tawaratsumida, S., Hosada, K. (1996). Purposive behavior acquisition for a real robot by vision-based reinforcement learning. *Machine Learning 23*:279–303.

Barto, A.G., Bradtke, S.J., Singh, S.P. (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence 72*:81–138.

Boutilier, C., Brafman, R.I., Geib, C. (1997). Prioritized goal Decomposition of Markov decision processes: Toward a synthesis of classical and decision theoretic planning. *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pp. 1165–1162.

Bradtke, S.J., and Duff, M.O. (1995). Reinforcement learning methods for continuous-time Markov decision problems. *Advances in Neural Information Processing Systems 8*:393–400. MIT Press, Cambridge, MA.

Brafman, R.I., Moshe, T. (1997). Modeling agents as qualitative decision makers. *Artificial Intelligence 94*(1):217-268.

Brockett, R.W. (1993). Hybrid models for motion control systems. In *Essays in Control: Perspectives in the Theory and and its Applications*, pp. 29–53. Birkhäuser, Boston.

Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE journal of Robotics and Automation*, 14–23.

Chrisman, L. (1994). Reasoning about probabilistic actions at multiple levels of granularity, *AAAI Spring Symposium: Decision-Theoretic Planning*, Stanford University.

Colombetti, M., Dorigo, M., Borghi, G. (1996). Behavior analysis and training: A methodology for behavior engineering. *IEEE Transactions on Systems, Man, and Cybernetics-Part B 26*(3):365–380

Crites, R.H., and Barto, A.G. (1996). Improving elevator performance using reinforcement learning. *Advances in Neural Information Processing Systems 9*:1017–1023. MIT Press, Cambridge, MA.

Dayan, P. (1993). Improving generalization for temporal difference learning: The successor representation. *Neural Computation 5*:613–624.

Dayan, P., Hinton, G.E. (1993). Feudal reinforcement learning. *Advances in Neural Information Processing Systems 5*:271–278. San Mateo, CA: Morgan Kaufmann.

de Kleer, J., Brown, J.S. (1984). A qualitative physics based on confluences. *Artificial Intelligence 24*(1–3):7–83.

Dean, T., Kaelbling, L.P., Kirman, J., Nicholson, A. (1995). Planning under time constraints in stochastic domains. *Artificial Intelligence 76*(1–2): 35–74.

Dean, T., Lin, S.-H. (1995). Decomposition techniques for planning in stochastic domains. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 1121–1127. Morgan Kaufmann. See also Technical Report CS-95-10, Brown University, Department of Computer Science, 1995.

Dejong, G.F. (1994). Learning to plan in continuous domains. *Artificial Intelligence 65*:71–141.

Dietterich, T.G. (1997). Hierarchical reinforcement learning with the MAXQ value function decomposition. Technical Report, Department of Computer Science, Oregon State University.

Dorigo, M., Colombetti, M. (1994). Robot shaping: Developing autonomous agents through learning. *Artificial Intelligence 71*:321–370.

Drescher, G.L. (1991). *Made Up Minds: A Constructivist Approach to Artificial Intelligence.* MIT Press.

Drummond, C. (1998). Composing functions to speed up reinforcement learning in a changing world. *Proceedings of the Tenth European Conference on Machine Learning.* Springer-Verlag.

Fikes, R.E., Hart, P.E., Nilsson, N.J. (1972). Learning and executing generalized robot plans. *Artificial Intelligence 3*:251–288.

Geffner, H., Bonet, B. (in preparation). High-level planning and control with incomplete information using POMDPs.

Grossman, R.L., Nerode, A., Ravn, A.P., Rischel, H. (1993). *Hybrid Systems*. Springer-Verlag, New York.

Haigh, K.Z., Shewchuk, J., Veloso, M.M. (1997). Exploring geometry in analogical route planning. *Journal of Experimental and Theoretical Artificial Intelligence 9*:509–541.

Hansen, E. (1994). Cost-effective sensing during plan execution. *Proc. AAAI-94*, pp. 1029–1035.

Hauskrecht, M., Meuleau, N., Boutilier, C., Kaelbling, L.P., Dean, T. (in preparation). Hierarchical solution of Markov decision processes using macro-actions.

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

Iba, G.A. (1989). A heuristic approach to the discovery of macro-operators. *Machine Learning 3*:285–317.

Jaakkola, T., Jordan, M.I., and Singh, S.P. (1994). On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation 6*(6):1185–1201.

Kaelbling, L.P. (1993). Hierarchical learning in stochastic domains: Preliminary results. *Proc. of the Tenth Int. Conf. on Machine Learning*, pp. 167–173, Morgan Kaufmann.

Kalmár, Z., Szepesvári, C., Lörincz, A. (1997). Module based reinforcement learning for a real robot. *Proceedings of the Sixth European Workshop on Learning Robots*, pp. 22–32.

Kalmár, Z., Szepesvári, C., Lörincz, A. (in preparation). Module based reinforcement learning: Experiments with a real robot.

Korf, R.E. (1985). *Learning to Solve Problems by Searching for Macro-Operators*. Boston: Pitman Publishers.

Korf, R.E. (1987). Planning as search: A quantitative approach. *Artificial Intelligence 33*:65–88.

Koza, J.R., Rice, J.P. (1992). Automatic programming of robots using genetic programming. *Proceedings of the Tenth National Conference on Artificial Intelligence*, pp. 194–201.

Kuipers, B.J. (1979). Commonsense knowledge of space: Learning from experience. *Proc. IJCAI-79*, pp. 499–501.

Laird, J.E., Rosenbloom, P.S., Newell, A. (1986). Chunking in SOAR: The anatomy of a general learning mechanism. *Machine Learning 1*:11–46.

Levinson, R., Fuchs, G. (1994). A pattern-weight formulation of search knowledge. Technical Report UCSC-CRL-94-10, University of California at Santa Cruz.

Lin, L.-J. (1993). *Reinforcement Learning for Robots Using Neural Networks*. PhD thesis, Carnegie Mellon University. Technical Report CMU-CS-93-103.

Maes, P. (1991). A bottom-up mechanism for behavior selection in an artificial creature. *Proceedings of the First International Conference on Simulation of Adaptive Behavior.* MIT Press.

Maes, P., Brooks, R. (1990). Learning to coordinate behaviors. *Proceedings of AAAI-90*, pp. 796–802.

Mahadevan, S., Connell, J. (1992). Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence 55*(2-3):311–365.

Mahadevan, S., Marchalleck, N., Das, T., Gosavi, A. (1997). Self-improving factory simulation using continuous-time average-reward reinforcement learning. *Proceedings of the 14th International Conference on Machine Learning.*

Marbach, P., Mihatsch, O., Schulte, M., Tsitsiklis, J.N. (1998). Reinforcement learning for call admission control in routing in integrated service networks. *Advances in Neural Information Processing Systems 10.* San Mateo: Morgan Kaufmann.

Mataric, M.J. (1997). Behavior-based control: Examples from navigation, learning, and group behavior. *Journal of Experimental and Theoretical Artificial Intelligence 9*(2–3).

McGovern, A., Sutton, R.S., Fagg, A.H. (1997). Roles of macro-actions in accelerating reinforcement learning. *Proceedings of the 1997 Grace Hopper Celebration of Women in Computing.*

McGovern, A., Sutton, R.S., (in prep.). Roles of temporally extended actions in accelerating reinforcement learning.

Meuleau, N., Hauskrecht, M., Kim, K.-E., Peshkin, L., Kaelbling, L.P., Dean, T., Boutilier, C. (in preparation). Solving very large weakly coupled Markov decision processes.

Millán, J. del R. (1994). Learning reactive sequences from basic reflexes. *Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, pp. 266–274.

Minton, S. (1988). *Learning Search Control Knowledge: An Explanation-based Approach.* Kluwer Academic.

Moore, A.W. (1994). The parti-game algorithm for variable resolution reinforcement learning in multidimensional spaces, *Advances in Neural Information Processing Systems 7*:711–718, MIT Press, Cambridge, MA.

Newell, A., Simon, H.A. (1972). *Human Problem Solving.* Prentice-Hall, Englewood Cliffs, NJ.

Nie, J., and Haykin, S. (to appear). A Q-learning based dynamic channel assignment technique for mobile communication systems. *IEEE Transactions on Vehicular Technology.*

Nilsson, N.J. (1973). Hierarchical robot planning and execution system. SRI AI Center Technical Note 76, SRI International, Inc., Menlo Park, CA.

Nilsson, N. (1994). Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research, 1*:139–158.

Parr, R., Russell, S. (1998). Reinforcement learning with hierarchies of machines. *Advances in Neural Information Processing Systems 11.* MIT Press, Cambridge, MA.

Parr, R. (in preparation). Hierarchical control and learning for Markov decision processes, chapter 3.

Precup, D., Sutton, R.S. (1997). Multi-time models for reinforcement learning. *Proceedings of the ICML'97 Workshop on Modelling in Reinforcement Learning.*

Precup, D., Sutton, R.S. (1998). Multi-time models for temporally abstract planning. *Advances in Neural Information Processing Systems 11.* MIT Press, Cambridge, MA.

Precup, D., Sutton, R.S., Singh, S.P. (1997). Planning with closed-loop macro actions. *Working notes of the 1997 AAAI Fall Symposium on Model-directed Autonomous Systems.*

Precup, D., Sutton, R.S., Singh, S.P. (1998). Theoretical results on reinforcement learning with temporally abstract options. *Proceedings of the Tenth European Conference on Machine Learning.* Springer-Verlag.

Puterman, M. L. (1994). *Markov Decision Problems.* Wiley, New York.

Rosenstein, M.T., Cohen, P.R. (1998). Concepts from time series. *Proceedings of the Fifteenth National Conference on Artificial Intelligence.*

Ring, M. (1991). Incremental development of complex behaviors through automatic construction of sensory-motor hierarchies. *Proceedings of the Eighth International Conference on Machine Learning,* pp. 343–347, Morgan Kaufmann.

Rudy, D., Kibler, D. (1992). Learning episodes for optimization. *Proceedings of the Ninth International Conference on Machine Learning,* Morgan Kaufmann.

Sacerdoti, E.D. (1974). Planning in a hierarchy of abstraction spaces. *Artificial Intelligence 5*:115–135.

Sastry, S. (1997). Algorithms for design of hybrid systems. *Proceedings of the International Conference of Information Sciences.*

Say, A.C.C., Selahattin, K. (1996). Qualitative system identification: Deriving structure from behavior. *Artificial Intelligence 83*(1):75–141.

Schmidhuber, J. (1991). Neural Sequence Chunkers. Technische Universitat Munchen TR FKI-148-91.

Simmons, R., Koenig, S. (1995). Probabilistic robot navigation in partially observable environments. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence,* pp. 1080–1087. Morgan Kaufmann.

Singh, S.P. (1992a). Reinforcement learning with a hierarchy of abstract models. *Proceedings of the Tenth National Conference on Artificial Intelligence*, pp. 202–207. MIT/AAAI Press.

Singh, S.P. (1992b). Scaling reinforcement learning by learning variable temporal resolution models. *Proceedings of the Ninth International Conference on Machine Learning*, pp. 406–415, Morgan Kaufmann.

Singh, S.P. (1992c). Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning 8*(3/4):323–340.

Singh, S.P. (1992d). The efficient learning of multiple task sequences. In *Advances in Neural Information Processing Systems 4*:251–258, Morgan Kaufmann.

Singh S.P., Barto A.G., Grupen R.A., Connolly C.I. (1994). Robust reinforcement learning in motion planning. *Advances in Neural Information Processing Systems 6*:655–662, Morgan Kaufmann.

Singh, S.P., Bertsekas, D. (1997). Reinforcement learning for dynamic channel allocation in cellular telephone systems. *Advances in Neural Information Processing Systems 9*:974–980. MIT Press.

Sutton, R.S. (1995). TD models: Modeling the world at a mixture of time scales. *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 531–539, Morgan Kaufmann.

Sutton, R.S., Barto, A.G. (1998). *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA.

Sutton, R.S., Pinette, B. (1985). The learning of world models by connectionist networks. *Proc. of the Seventh Annual Conf. of the Cognitive Science Society*, pp. 54-64.

Tenenberg, J. Karlsson, J., Whitehead, S. (1992). Learning via task decomposition. *Proc. Second Int. Conf. on the Simulation of Adaptive Behavior.* MIT Press.

Tesauro, G.J. (1995). Temporal difference learning and TD-Gammon. *Communications of the ACM 38*:58–68.

Thrun, T., Schwartz, A. (1995). Finding structure in reinforcement learning. *Advances in Neural Information Processing Systems 7.* San Mateo: Morgan Kaufmann.

Tóth, G.J., Kovács, S., Lörincz, A. (1995). Genetic algorithm with alphabet optimization. *Biological Cybernetics 73*:61–68.

Uchibe, M., Asada, M., Hosada, K. (1996). Behavior coordination for a mobile robot using modular reinforcement learning. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1329–1336.

Watkins, C.J.C.H. (1989). *Learning with Delayed Rewards.* PhD thesis, Cambridge University.

Wiering, M., Schmidhuber, J. (1997). HQ-learning. *Adaptive Behavior 6*(2).

Wixson, L.E. (1991). Scaling reinforcement learning techniques via modularity, *Proc. Eighth Int. Conf. on Machine Learning*, pp. 368–372, Morgan Kaufmann.