In *Machine Learning: ECML'98. 10th European conference on Machine Learning, Chemnitz, Germany, April 1998. Proceedings*, pp.382–393. Springer Verlag.

Theoretical Results on Reinforcement Learning with Temporally Abstract Options

Doina Precup[1], Richard S. Sutton[1], and Satinder Singh[2]

[1] Department of Computer Science
University of Massachusetts
Amherst, MA 01003-4610
http://www.cs.umass.edu/{~dprecup|~rich}

[2] Department of Computer Science
University of Colorado
Boulder, CO 80309-0430
http://www.cs.colorado.edu/~baveja

Abstract. We present new theoretical results on planning within the framework of temporally abstract reinforcement learning (Precup & Sutton, 1997; Sutton, 1995). Temporal abstraction is a key step in any decision making system that involves planning and prediction. In temporally abstract reinforcement learning, the agent is allowed to choose among "options", whole courses of action that may be temporally extended, stochastic, and contingent on previous events. Examples of options include closed-loop policies such as picking up an object, as well as primitive actions such as joint torques. Knowledge about the consequences of options is represented by special structures called multi-time models. In this paper we focus on the theory of planning with multi-time models. We define new Bellman equations that are satisfied for sets of multi-time models. As a consequence, multi-time models can be used interchangeably with models of primitive actions in a variety of well-known planning methods including value iteration, policy improvement and policy iteration.

1 Introduction

Model-based reinforcement learning offers a possible solution to the problem of integrating planning with real-time learning and decision-making [20]. However, conventional model-based reinforcement learning uses one-step models [11, 13, 18], that cannot represent common-sense, higher-level actions, such as picking an object or traveling to a specified location.

Several researchers have proposed extending reinforcement learning to a higher level by treating entire closed-loop policies as actions [3–6, 9, 10, 12, 17]. In order to use such actions in planning, an agent needs the ability to create and handle models at a variety of different, interrelated levels of temporal abstraction. Sutton [19] introduced an approach to modeling at different time scales, based on prior work by Singh [17], Dayan [2] and by Sutton and Pinette [21]. This approach enables models of the environment at different temporal scales to be intermixed, producing temporally abstract models. In previous work [14], we generalized this approach from the prediction case, to the full control case. In this paper, we summarize the framework of temporally abstract reinforcement learning and present new theoretical results on planning with general *options* and temporally abstract models of options.

Options are similar to AI's classical "macro operators" [7,8,16], in that they can take control for some period of time, determining the actions during that time, and in that one can choose among options much as one originally chose among primitive actions. However, classical macro operators are only a fixed sequence of actions, whereas options incorporate a general (possibly non-Markov) closed-loop policy and completion criterion. These generalizations are required when the environment is stochastic and uncertain with general goals, as in reinforcement learning and Markov decision processes (MDP).

The predictive knowledge needed in order to plan using options can be represented through *multi-time models* [14]. Such models summarize several time scales and have the ability to predict events that can happen at various unknown moments. In this paper, we focus on the theoretical properties of multi-time models. We show formally that such models can be used interchangeably with models of primitive actions in a variety of well-known dynamic programming methods, while preserving the same guarantees of convergence to correct solutions. The benefit of using such temporally extended models is a significant improvement in the convergence rates of these algorithms.

2 Reinforcement Learning (MDP) Framework

First we briefly summarize the mathematical framework of the reinforcement learning problem that we use in the paper. In this framework, a learning *agent* interacts with an *environment* at some discrete, lowest-level time scale $t = 0, 1, 2, \ldots$. At each time step, the agent perceives the state of the environment, s_t, and on that basis chooses a primitive action, a_t. In response to each primitive action, a_t, the environment produces one step later a numerical reward, r_{t+1}, and a next state, s_{t+1}.

The agent's objective is to learn a policy π, which is a mapping from states to probabilities of taking each action, that maximizes the expected discounted future reward from each state s:

$$V^\pi(s) = E\Big\{r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots \;\Big|\; s_0 = s, \pi\Big\},$$

where $\gamma \in [0, 1)$ is a *discount-rate* parameter. The quantity $V^\pi(s)$ is called the *value* of state s under policy π, and V^π is called the value function for policy π. The optimal value of a state is denoted

$$V^*(s) = \max_\pi V^\pi(s)$$

The environment is henceforth assumed to be a stationary, finite Markov decision process. We assume that the states are discrete and form a finite set, $s_t \in \{1, 2, \ldots, n\}$. The latter assumption is a temporary theoretical convenience; it is not a limitation of the ideas we present.

3 Options

In order to achieve faster planning, the agent should be able to predict what happens if it follows a certain course of action over a period of time. By a "course of action" we mean any way of behaving, i.e. any way of mapping representations of states to primitive actions. An option is a way of choosing actions that is initiated, takes control for some period of time, and then eventually ends. Options are defined by three elements:

– the set of states $\mathcal{I}$ in which the option applies
– a decision rule μ which specifies what actions are executed by the option
– a completion function β which specifies the probability of completing the option on every time step.

μ is of a slightly more general form than the policies used in conventional reinforcement learning. The first generalization is based on the observation that primitive actions qualify as options: they are initiated in a state, take control for a while (one time step), and then end. Therefore, we allow μ to choose among options, rather than only among primitive actions.

The conventional reinforcement learning setting also requires that a policy's decision probabilities at time t should be a function only of the current state s_t. This type of policy is called *Markov*. For the policy of a option, μ, we relax this assumption, and we allow the probabilities of selecting a sub-option to depend on all the states and actions from time t_0, when the option began executing, up through the current time, t. We call policies of this more general class *semi-Markov*.

The completion function β can also be semi-Markov. This property enables us to describe various kinds of completion. Perhaps the simplest case is that of a option that completes after some fixed number of time steps (e.g. after 1 step, as in the case of primitive actions, or after n steps, as in the case of a classical macro operator consisting of a sequence of n actions). A slightly more general case is that in which the option completes during a certain time period, e.g. 10 to 15 time steps later. In this case, the completion function β should specify the probability of completion for each of these time steps. The case which is probably the most useful in practice is the completion of a option with the occurrence of a critical state, often a state that we think of as a subgoal. For instance, the option `pick-up-the-object` could complete when the object is in the hand. This event occurs at a very specific moment in time, but this moment is indefinite, not known in advance. In this case, the completion function depends on the state history of the system, rather than explicitly on time. Lastly, if β is always 0, the option does not complete. This is the case of usual policies used in reinforcement learning.

Options are typically executed in a *call-and-return* fashion. Each option o can be viewed as a "subroutine" which calls sub-options, according to its internal decision rule. When a sub-option o' is selected, it takes control of the action choices until it completes. Upon completion, o' transfers the control back to o. o also inherits the current time t' and the current state $s_{t'}$, and has to decide if it should terminate or pick a new sub-option.

Two options, a and b, can be *composed* to yield a new option, denoted ab, that first follows a until it terminates and then follows b until it terminates, also terminating the composed option ab.

4 Models of Options

Planning in reinforcement learning refers to the use of models of the effects of actions to compute value functions, particularly V^*. We use the term *model* for any structure that generates predictions based on the representation of the state of the system and on the course of action that the system is following.

In order to plan at the level of options, we need to have a knowledge representation form that predicts their consequences. The *multi-time model* of a option characterizes the states that result upon the option's completion and the truncated return received along the way when the option is executed in various states [14]. Let $\mathbf{p}$ be an n-vector and r a scalar. The pair $\mathbf{p}, r$ is an *accurate prediction* for the execution of semi-Markov option o in state s if and only if

$$r = E\Big\{r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{T-1} r_T \;\Big|\; s_t = s, o\Big\} \tag{1}$$

and

$$\mathbf{p} = E\Big\{\gamma^T \mathbf{s}_T \;\Big|\; s_t = s, o\Big\}, \tag{2}$$

where T is the random variable denoting the time at which o terminates when executed in $s_t = s$, and $\mathbf{s}_T$ denotes the unit basis n-vector corresponding to s_T. $\mathbf{p}$ is called the *state prediction vector* and o is called the *reward prediction* for state s and option o.

We will use the notation "$\cdot$", as in $\mathbf{x} \cdot \mathbf{y}$, to represents the inner or dot product between vectors. We will refer to the i-th element of a vector $\mathbf{x}$ as $x(i)$. The transpose of a vector $\mathbf{x}$ will be denoted by $\mathbf{x}^T$.

The predictions corresponding to the states $s \notin \mathcal{I}$ are always 0. If the option never terminates, the reward prediction r is equal to the value function of the internal policy of the option, and the elements of $\mathbf{p}$ are all 0.

The state prediction vectors for the same option in all the states are often grouped as the rows of an $n \times n$ *state prediction matrix*, $\mathbf{P}$, and the reward predictions are often grouped as the components of an n-component *reward prediction vector*, $\mathbf{r}$. $\mathbf{r}, \mathbf{P}$ form the *accurate model* of the option.

A key theoretical result refers to the way in which the model of a composed option can be obtained from the models of its components.

Theorem 1 (Composition or Sequencing). *Given an accurate prediction $r_a, \mathbf{p}_a$ for some option a applied in state s and an accurate model $\mathbf{r}_b, \mathbf{P}_b$ for some option b, the prediction:*

$$g = r_a + \mathbf{p}_a \cdot \mathbf{r}_b \quad \textit{and} \quad \mathbf{p} = \mathbf{p}_a{}^T \mathbf{P}_b \tag{3}$$

is an accurate prediction for the composed option ab when it is executed in s.

Proof. Let k be the random variable denoting the time at which a completes, and T be the random variable denoting the time at which ab completes. Then we have:

$$\begin{aligned}
r &= E\{r_{t+1} + \cdots + \gamma^{T-1} r_T \,|\, s_t = s, ab\} \\
&= E\{r_{t+1} + \cdots + \gamma^{k-1} r_k \,|\, s_t = s, a\} + E\{\gamma^k r_{k+1} + \cdots + \gamma^{T-1} r_T \,|\, s_t = s, ab\} \\
&= r_a + \sum_{i=1}^{\infty} P\{k = i \mid s_t = s, a\} \gamma^i \sum_{s'} P\{s_k = s' \mid s_t = s, k = i\} \\
&\qquad E\{r_{k+1} + \cdots + \gamma^{T-k-1} r_T \,|\, s_k = s', b\} \\
&= r_a + \sum_{s'} \sum_{i=1}^{\infty} P\{k = i \mid s_t = s, a\} \gamma^i P\{s_k = s' \mid s_t = s, k = i\} r_b(s')
\end{aligned}$$

$$
\begin{aligned}
&= r_a + \sum_{s'} p_a(s') r_b(s') \\
&= r_a + \mathbf{p}_a \cdot \mathbf{r}_b
\end{aligned}
$$

The equation for $\mathbf{p}$ follows similarly. □

A simple combination of models can also be used to predict the effect of probabilistic choice among options:

Theorem 2 (Averaging or Choice). *Let $r_i, \mathbf{p}_i$ be a set of accurate predictions for the options o_i executed in state s, and let $w_i > 0$ be a set of numbers such that $\sum_i w_i = 1$. Then the prediction defined by:*

$$
r = \sum_i w_i r_i \quad \text{and} \quad \mathbf{p} = \sum_i w_i \mathbf{p}_i \tag{4}
$$

is accurate for the option o , which chooses in state s among sub-options o_i with probabilities w_i and then follows the chosen o_i until completion.

Proof. Based on the definition of o, $r = \sum_i w_i E\{r_{t+1} + \cdots + \gamma^{T-1} r_T \,|\, s_t = s, o_i\} = \sum_i w_i r_i$ The equation for $\mathbf{p}$ follows similarly. □

These results represent the basis for developing the theory of planning at the level of options. The options map the low-level MDP in a higher-level *semi-Markov decision process (SMDP)* [15], which we can solve using dynamic programming methods. We will now go into the details of this mapping.

5 Planning with Models of Options

In this section, we extend the theoretical results of dynamic programming for the case in which the agent is allowed to use an arbitrary set of options, $\mathcal{O}$. If $\mathcal{O}$ is exactly the set of primitive actions, then our results degenerate to the conventional case. We assume, for the sake of simplicity, that for any state s, the set of options that apply in s, denoted $\mathcal{O}_s$, is always non-empty. However, the theory that we present extends, with some additional complexity, to the case in which no options apply in certain states.

Given a set of options $\mathcal{O}$, we define a policy π to be a mapping that, for any state s specifies the probability of taking each option from $\mathcal{O}_s$. The value of π is defined similarly to the conventional case, as the expected discounted reward if the policy is applied starting in s:

$$
V^\pi(s) = E\Big\{ r_{t+1} + \gamma r_{t+2} + \cdots \;\Big|\; s_t = s, \pi \Big\}. \tag{5}
$$

The *optimal value function, given the set $\mathcal{O}$,* can be defined as

$$
V^*_{\mathcal{O}}(s) = \sup_{\pi \in \Pi_{\mathcal{O}}} V^\pi(s), \tag{6}
$$

for all s, where $\Pi_{\mathcal{O}}$ is the set of policies that can be defined using the options from $\mathcal{O}$. The value functions are sometimes represented as n-vectors, $\mathbf{V}^\pi$ and $\mathbf{V}^*_{\mathcal{O}}$, with each component representing the value of a different state.

The value function of any Markov policy $\pi \in \Pi_{\mathcal{O}}$ satisfies the Bellman evaluation equations:

$$V^{\pi}(s) = \sum_{o \in \mathcal{O}_s} \pi(s,o)(r_o + \mathbf{p}_o \cdot \mathbf{V}^{\pi}), \qquad \text{for all } s \in S, \tag{7}$$

where $\pi(s,o)$ is the probability of choosing option o in state s when acting according to π and $r_o, \mathbf{p}_o$ is the accurate prediction for sub-option o in state s. We will show that, similarly to the conventional case, $\mathbf{V}^{\pi}$ is the unique solution to this system of equations.

Similarly, the optimal value function given a set of options $\mathcal{O}$ satisfies the Bellman optimality equations:

$$V^*_{\mathcal{O}}(s) = \max_{o \in \mathcal{O}_s} (r_o + \mathbf{p}_o \cdot \mathbf{V}^*_{\mathcal{O}}), \qquad \text{for all } s \in S, \tag{8}$$

As in the conventional case, $\mathbf{V}^*_{\mathcal{O}}$ is the unique solution of this set of equations, and there exists at least one deterministic Markov policy $\pi^*_{\mathcal{O}} \in \Pi_{\mathcal{O}}$ that is optimal, i.e., for which $\mathbf{V}^{\pi^*_{\mathcal{O}}} = \mathbf{V}^*_{\mathcal{O}}$.

We will now present in detail the proofs leading to these theoretical results. The practical consequence is that all the usual update rules used in reinforcement learning can be used to compute $\mathbf{V}^{\pi}$ and $\mathbf{V}^*_{\mathcal{O}}$ when the agent makes choices among options. We focus first on the Bellman policy evaluation equations.

Theorem 3 (Value Functions for Composed Policies). *Let π be a policy that, when starting in state s, follows option o and, when the option completes, follows policy π'. Then*

$$V^{\pi}(s) = r_o + \mathbf{p}_o \cdot \mathbf{V}^{\pi'}. \tag{9}$$

Proof. According to the theorem statement, $\pi = o\pi'$. The conclusion follows immediately from the composition theorem. □

Theorem 4 (Bellman Policy Evaluation Equation). *The value function of any Markov policy satisfies the Bellman policy evaluation equations (7).*

Proof. Using the averaging rule and the fact that π is Markov, we have:

$$V^{\pi}(s) = \sum_{o \in \mathcal{O}_s} \pi(s,o) V^{o\pi}(s)$$

We can expand $v^{o\pi}(s)$ using (9), to obtain the desired result (7). □

We now prove that the Bellman evaluation equations have a unique solution, and that this solution can be computed using the well-known algorithm of *policy evaluation with successive approximations*: start with arbitrary initial values $V_0(s)$ for all states and iterate for all s: $V_{k+1}(s) \leftarrow r_{\pi(s)} + \mathbf{p}_{\pi(s)} \cdot \mathbf{V}_k$, where $\pi(s)$ is the option suggested by π in state s.

Proof. For any option o and any starting state s, there exists a constant ϵ_o such that $\| \mathbf{p}_o \|_1 \leq \epsilon_o < 1$, where $\| \cdot \|_1$ denotes the $l_1 - norm$ of a vector: $\| \mathbf{x} \|_1 = \sum_i x(i)$. The value of ϵ_o depends on γ and on the expected duration of o when started in s. For any option o, we show that the operator $T_o(\mathbf{V}) = r_o + \mathbf{p}_o \cdot \mathbf{V}$ is a contraction with constant ϵ_o. For arbitrary vectors $\mathbf{V}$ and $\mathbf{V}'$, we have:

$$T_o(\mathbf{V}) - T_o(\mathbf{V}') = \sum_{s'} p_o(s')(V(s') - V'(s')) \leq \sum_{s'} p_o(s') \| \mathbf{V} - \mathbf{V}' \|_\infty,$$

where $\| \cdot \|_\infty$ denotes the $l_\infty - norm$ of a vector: $\| \mathbf{x} \|_\infty = \max_i x(i)$. Therefore, $T_o(\mathbf{V}) - T_o(\mathbf{V}') \leq \epsilon_o \| \mathbf{V} - \mathbf{V}' \|_\infty$ The result follows from the contraction mapping theorem [1]. □

So far we have established that the value functions of Markov option policies have similar properties with the Markov policies that use only primitive actions. Now we establish similar results for the optimal value function that can be obtained when planning with a set of options.

Theorem 5 (Bellman Optimality Equation). *For any set of options $\mathcal{O}$, the optimal value function $\mathbf{V}^*_{\mathcal{O}}$ satisfies the Bellman optimality equations (8).*

Proof. Let π be an arbitrary policy which, at time step 0, in state $s_0 = s$, chooses among the available options with probabilities w_o.

$$V^\pi(s) = \sum_{o \in \mathcal{O}_s} w_o (r_o + \sum_{s'} p_o(s') W_\pi(s'))$$

where $W_\pi(s')$ is the expected discounted return from state s' on. Since $p_o(s') \geq 0, \forall s'$, using the definition of $V^*_{\mathcal{O}}(s')$, we have:

$$V^\pi(s) \leq \sum_{o \in \mathcal{O}_s} w_o (r_o + \mathbf{p}_o \cdot \mathbf{V}^*_{\mathcal{O}}) \leq \max_{o \in \mathcal{O}_s} (r_o + \mathbf{p}_o \cdot \mathbf{V}^*_{\mathcal{O}})$$

Since π is arbitrary, due to the definition of $V^*_{\mathcal{O}}(s)$,

$$V^*_{\mathcal{O}}(s) \leq \max_{o \in \mathcal{O}_s} (r_o + \mathbf{p}_o \cdot \mathbf{V}^*_{\mathcal{O}}).$$

On the other hand, let $o_0 = \arg\max_{o \in \mathcal{O}_s} (r_o + \mathbf{p}_o \cdot \mathbf{V}^*_{\mathcal{O}})$. Let π be the policy that chooses o_0 at time step 0 and, after o_0 ends, in state s', switches to a policy $\pi_{s'}$ such that $v^{\pi_{s'}}(s') \geq V^*_{\mathcal{O}}(s') - \epsilon$. $\pi_{s'}$ exists because of the way in which $V^*_{\mathcal{O}}$ was defined. From (9), we have:

$$V^\pi(s) = r_{o_0} + \sum_{s'} p_{o_0}(s') v^{\pi_{s'}}(s') \geq r_{o_0} + \sum_{s'} p_{o_0}(s')(V^*_{\mathcal{O}}(s') - \epsilon) \geq r_{o_0} + \mathbf{p}_{o_0} \cdot \mathbf{V}^*_{\mathcal{O}} - \epsilon$$

Since $V^*_{\mathcal{O}}(s) \geq v^\pi(s)$, we have:

$$V^*_{\mathcal{O}}(s) \geq \max_{o \in \mathcal{O}_s} (r_o + \mathbf{p}_o \cdot \mathbf{V}^*_{\mathcal{O}}) - \epsilon$$

Since ϵ is arbitrary, it follows that

$$V^*_{\mathcal{O}}(s) = \max_{o \in \mathcal{O}_s} (r_o + \mathbf{p}_o \cdot \mathbf{V}^*_{\mathcal{O}}) \quad \square$$

The following step is to show that the solution of the Bellman optimality equations (8) is bounded and unique, and that it can be computed through a *value iteration algorithm*: start with arbitrary initial values $V_0(s)$ and iterate the update: $V_{k+1}(s) \leftarrow \max_{o \in \mathcal{O}_s} r_o + \mathbf{p}_o \cdot \mathbf{V}_k, \forall s$.

Proof. For any set of actions $\mathcal{O}$ let us consider the operator $T_{\mathcal{O}}$ which, in any state s, performs the following transformation: $T_{\mathcal{O}_s}(\mathbf{V}) = \max_{o \in \mathcal{O}_s}(r_o + \mathbf{p}_o \cdot \mathbf{V})$ Let $\mathbf{V}$ and $\mathbf{V}'$ be two arbitrary vectors. Then we have:

$$\begin{aligned} T_{\mathcal{O}_s}(\mathbf{V}) &= \max_{o \in \mathcal{O}_s}(r_o + \mathbf{p}_o(\mathbf{V}' + \mathbf{V} - \mathbf{V}')) = \max_{o \in \mathcal{O}_s}(T_o(\mathbf{V}') + \mathbf{p}_o \cdot (\mathbf{V} - \mathbf{V}')) \\ &\leq T_{\mathcal{O}_s}(\mathbf{V}') + \max_{o \in \mathcal{O}_s} \sum_{s'} p_o(s') \| \mathbf{V} - \mathbf{V}' \|_\infty = T_{\mathcal{O}_s}(\mathbf{V}') + \epsilon_{\mathcal{O}_s} \| \mathbf{V} - \mathbf{V}' \|_\infty, \end{aligned}$$

where $\epsilon_{\mathcal{O}_s} = \max_{o \in \mathcal{O}_s} \epsilon_o$. Similarly,

$$T_{\mathcal{O}_s}(\mathbf{V}') \leq T_{\mathcal{O}_s}(\mathbf{V}) + \epsilon_{\mathcal{O}_s} \| \mathbf{V} - \mathbf{V}' \|_\infty$$

Therefore, $\forall s$,

$$\| T_{\mathcal{O}}(\mathbf{V}) - T_{\mathcal{O}}(\mathbf{V}') \|_\infty \leq \epsilon_{\mathcal{O}} \| \mathbf{V} - \mathbf{V}' \|_\infty,$$

where $\epsilon_{\mathcal{O}} = \max_s \epsilon_{\mathcal{O}_s}$. Therefore $T_{\mathcal{O}}$ is a contraction with constant $\epsilon_{\mathcal{O}}$. The results follow from the contraction mapping theorem [1].

So far we have shown that the optimal value function $\mathbf{V}^*_{\mathcal{O}}$ is the unique bounded solution of the Bellman optimality equations. Now we will show that this value function can be achieved by a deterministic Markov policy:

Theorem 6 (Value Achievement). *The policy $\pi^*_{\mathcal{O}}$ defined as*

$$\pi^*_{\mathcal{O}}(s) = \arg\max_{o \in \mathcal{O}_s} r_o + \mathbf{p}_o \cdot \mathbf{V}^*_{\mathcal{O}}$$

achieves $\mathbf{V}^*_{\mathcal{O}}$.

Proof. For any arbitrary state s, we have:

$$0 \leq V^*_{\mathcal{O}}(s) - V^{\pi^*_{\mathcal{O}}}(s) = \mathbf{p}_{\pi^*_{\mathcal{O}}(s)} \cdot (\mathbf{V}^*_{\mathcal{O}} - \mathbf{V}^{\pi^*_{\mathcal{O}}}) \leq \epsilon_{\mathcal{O}_s} \| \mathbf{V}^*_{\mathcal{O}} - \mathbf{V}^{\pi^*_{\mathcal{O}}} \|_\infty.$$

Since $\epsilon_{\mathcal{O}} < 1, V^*_{\mathcal{O}}(s) = V^{\pi^*_{\mathcal{O}}}(s)$. □

In order to find the optimal policy given a set of options, one can simply compute $\mathbf{V}^*_{\mathcal{O}}$ and then use it to pick actions greedily. Another popular planning method for computing optimal policies is *policy iteration*, which alternates steps of policy evaluation and policy improvement. We have already investigated policy evaluation, so we turn now to policy improvement:

Theorem 7 (Policy Improvement Theorem). *For any Markov policy π defined using options from a set $\mathcal{O}$, let π' be a new policy which, for some state s, chooses greedily among the available options, and then follows π*

$$\pi'(s) = \arg\max_{o \in \mathcal{O}_s} r_o + \mathbf{p}_o \cdot \mathbf{V}^\pi,$$

Then $V^{\pi'}(s) \geq V^\pi(s)$.

Proof. Let o_0 be the option chosen by π in state s. Then, from (7),

$$V^{\pi}(s) = r_{o_0} + \mathbf{p}_{o_0} \cdot \mathbf{V}^{\pi} \leq \max_{o \in \mathcal{O}_s} r_o + \mathbf{p}_o \cdot \mathbf{V}^{\pi} = V^{\pi'}(s). \square$$

Given a set of options $\mathcal{O}$ such that $\mathcal{O}_s$ is finite for all s, the policy iteration algorithm, which interleaves policy evaluation and policy improvement, converges to $\pi^*_{\mathcal{O}}$ in a finite number of steps.

The final result relates the models of options to the optimal value function of the environment, V^*.

Theorem 8. *If $r_o, \mathbf{p}_o$ is an accurate prediction for some option b in state s, then*

$$r_o + \mathbf{p}_o \cdot \mathbf{V}^* \leq V^*(s)$$

We say that accurate models are non-overpromising, *i.e. they never promise more that the agent can actually achieve.*

Proof. Assume that the agent can use o and all the primitive actions in the environment. Let $\pi = o\pi^*$, where π^* is the optimal policy of the environment. Then we have:

$$V^*(s) \geq V^{\pi}(s) = r_o + \mathbf{p}_o \cdot \mathbf{V}^* \quad \square$$

6 Illustrations

The theoretical results presented so far show that accurate models of options can be used in all the planning algorithms typically employed for solving MDPs, with the same guarantees of convergence to correct plans as in the case of primitive actions. We will now illustrate the speedup that can be obtained when using these methods in two simple gridworld learning tasks.

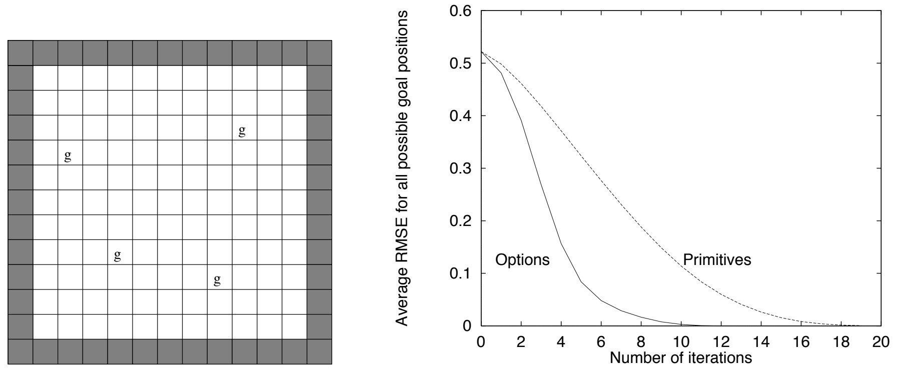

Fig. 1. Empty gridworld task. Options allow the error in the value function estimation to decrease more quickly

The first task is depicted on the left panel of figure 1. The cells of the grid correspond to the states of the environment. From any state the agent can perform one of four primitive actions, `up`, `down`, `left` or `right`. With probability 2/3, the actions cause the agent to move one cell in the corresponding direction (unless this would take the agent into a wall, in which case it stays in the same state). With probability 1/3, the agent moves instead in one of the other three directions (unless this takes it into a wall, of course). There is no penalty for bumping into walls. In addition to these primitive actions, the agent can use four additional higher-level options, to travel to each of the marked locations. These locations have been chosen randomly inside the environment. Accurate models for all the options are also available. Both the options and their models have been learned during a prior random walk in the environment, using Q-learning [22] and the β-model learning algorithm [19].

The agent is repeatedly given new goal positions and it needs to compute optimal paths to these positions as quickly as possible. In this experiment, we considered all possible goal positions. In each case, the value of the goal state is 1, there are no rewards along the way, and the discounting factor is $\gamma = 0.9$. We performed planning according to the standard value iteration method, where the starting values are $V_0(s) = 0$ for all the states except the goal state, for which $V_0(goal) = 1$. In the first experiment, the agent was only allowed to use primitive actions, while in the second case, it used both the primitive actions and the higher-level options.

The right panel in figure 1 shows the average root mean squared error in the estimate of the optimal value function over the whole environment. The average is computed over all possible positions of the goal state. The use of higher-level options introduces a significant speedup in convergence, even though the options have been chosen arbitrarily. Note that an iteration using all the options is slightly more expensive than an iteration using only primitive actions. This aspect can be improved by using more sophisticated methods of ordering the options before doing the update. However, such methods are beyond the scope of this paper.

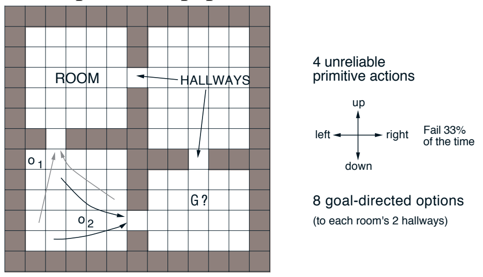

Fig. 2. Example Task. The natural options are to move from room to room

In order to analyze in more detail the effect of options, let us consider a second environment. In this case, the gridworld has four “rooms”. The basic dynamics of the environment are the same as in the previous case. For each state in a room, two higher-level options are available, which can take the agent to each of the hallways adjacent to the room. Each of these options has two outcome states: the target hallway, which corresponds to a successful outcome, and the state adjacent to the other hallway, which corresponds to failure (the agent has wandered out of the room). The completion function β is therefore 0 for all the states except these outcome states, where it is 1. The policy π underlying the option is the optimal policy for reaching the target hallway.

The goal state can have an arbitrary position in any of the rooms, but for this illustration let us suppose that the goal is two steps down from the right hallway. The value of the goal state is 1, there are no rewards along the way, and the discounting factor

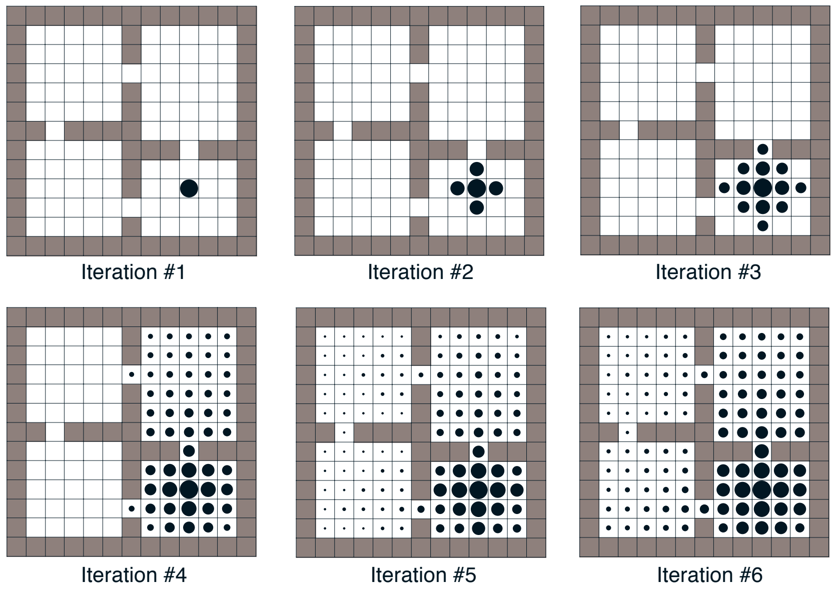

Fig. 3. Value iteration using primitive actions and higher-level options

is $\gamma = 0.9$. We performed planning again, according to the standard value iteration method, with the same setting as in the previous task.

When using only primitive actions, the values are propagated one step on each iteration. After six iterations, for instance, only the states that are within six steps of the goal are attributed non-zero values. Figure 3 shows the value function after each iteration, using all available options. The area of the circle drawn in each state is proportional to the value attributed to the state. The first three iterations are identical to the case in which only primitive actions are used. However, once the values are propagated to the first hallway, all the states in the rooms adjacent to the hallway receive values as well. For the states in the room containing the goal, these values correspond to performing the option of getting into the right hallway, and then following the optimal primitive actions to get to the goal. At this point, a path to the goal is known from each state in the right half of the environment, even if the path is not optimal for all the states. After six iterations, an optimal policy is known for all the states in the environment.

7 Discussion

Planning with multi-time models converges to correct solutions significantly faster than planning at the level of primitive actions. There are two intuitive reasons that justify this result. First, the temporal abstraction achieved by the models enables the agent to reason at a higher level. Second, the knowledge captured in the models only depends only on the MDP underlying the environment and on the option itself. The options and multi-time models can therefore be seen as an efficient means of transferring knowledge across different reinforcement learning tasks, as long as the dynamics of the environment are preserved.

Theoretical results similar to the ones presented in this paper are also available for planning in optimal stopping tasks [1], as well as for a different regime of executing options, which allows early termination. Further research will be devoted to integrating temporal and state abstraction, and to the issue of discovering useful options.

Acknowledgments
The authors thank Amy McGovern, Andy Fagg, Leo Zelevinsky, Manfred Huber and Ron Parr for helpful discussions and comments contributing to this paper. This research was supported in part by NSF grant ECS-9511805 to Andrew G. Barto and Richard S. Sutton, and by AFOSR grant AFOSR-F49620-96-1-0254 to Andrew G. Barto and Richard S. Sutton. Satinder Singh was supported by NSF grant IIS-9711753. Doina Precup also acknowledges the support of the Fulbright foundation.

References

1. Dimitri P. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models*. Prentice Hall, Englewood Cliffs, NJ, 1987.
2. Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5:613–624, 1993.
3. Peter Dayan and Geoff E. Hinton. Feudal reinforcement learning. In *Advances in Neural Information Processing Systems 5*, pages 271–278. Morgan Kaufmann, 1993.
4. Thomas G. Dietterich. Hierarchical reinforcement learning with MAXQ value function decomposition. Technical report, Computer Science Department, Oregon State University, 1997.
5. Manfred Huber and Roderic A. Grupen. A feedback control structure for on-line learning tasks. *Robotics and Autonomous Systems*, 22(3-4):303–315, 1997.
6. Leslie P. Kaelbling. Hierarchical learning in stochastic domains: Preliminary results. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 167–173. Morgan Kaufmann, 1993.
7. Richard E. Korf. *Learning to Solve Problems by Searching for Macro-Operators*. Pitman Publishing Ltd, 1985.
8. John E. Laird, Paul S. Rosenbloom, and Allan Newell. Chunking in SOAR: The anatomy of a general learning mechanism. *Machine Learning*, 1:11–46, 1986.
9. Sridhar Mahadevan and Jonathan Connell. Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, 55(2-3):311–365, 1992.
10. Amy McGovern, Richard S. Sutton, and Andrew H. Fagg. Roles of macro-actions in accelerating reinforcement learning. In *Grace Hopper Celebration of Women in Computing*, pages 13–17, 1997.
11. Andrew W. Moore and Chris G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, 13:103–130, 1993.
12. Ronald Parr and Stuart Russell. Reinforcement learning with hierarchies of machines. In *Advances in Neural Information Processing Systems 10*. MIT Press, 1998.
13. Jing Peng and John Williams. Efficient learning and planning within the Dyna framework. *Adaptive Behavior*, 4:323–334, 1993.
14. Doina Precup and Richard S. Sutton. Multi-time models for temporally abstract planning. In *Advances in Neural Information Processing Systems 10 (Proceedings of NIPS'97)*, pages 1050–1056. MIT Press, 1998.
15. Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 1994.
16. Earl D. Sacerdoti. *A Structure for Plans and Behavior*. Elsevier North-Holland, 1977.
17. Satinder P. Singh. Scaling reinforcement learning by learning variable temporal resolution models. In *Proceedings of the Ninth International Conference on Machine Learning*, pages 406–415. Morgan Kaufmann, 1992.
18. Richard S. Sutton. Integrating architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning ICML'90*, pages 216–224. Morgan Kaufman, 1990.
19. Richard S. Sutton. TD models: Modeling the world as a mixture of time scales. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 531–539. Morgan Kaufmann, 1995.
20. Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.
21. Richard S. Sutton and Brian Pinette. The learning of world models by connectionist networks. In *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, pages 54–64, 1985.
22. Christopher J. C. H. Watkins. *Learning with Delayed Rewards*. PhD thesis, Psychology Department, Cambridge University, Cambridge, UK, 1989.