Reward-Respecting Subtasks for Model-Based Reinforcement Learning

Richard S. Sutton[†‡§], **Marlos C. Machado**[†‡§], **G. Zacharias Holland**[†],
David Szepesvari[†], **Finbarr Timbers**[†], **Brian Tanner**[†], **Adam White**[†‡§]
[†]DeepMind, [‡] University of Alberta, [§] Alberta Machine Intelligence Institute (Amii)
Corresponding author: marlosm@deepmind.com

Abstract

To achieve the ambitious goals of artificial intelligence, reinforcement learning must include planning with a model of the world that is abstract in state and time. Deep learning has made progress in state abstraction, but, although the theory of time abstraction has been extensively developed based on the options framework, in practice options have rarely been used in planning. One reason for this is that the space of possible options is immense and the methods previously proposed for option discovery do not take into account how the option models will be used in planning. Options are typically discovered by posing subsidiary tasks such as reaching a bottleneck state, or maximizing a sensory signal other than the reward. Each subtask is solved to produce an option, and then a model of the option is learned and made available to the planning process. The subtasks proposed in most previous work ignore the reward on the original problem, whereas we propose subtasks that use the original reward plus a bonus based on a feature of the state at the time the option stops. We show that options and option models obtained from such reward-respecting subtasks are much more likely to be useful in planning and can be learned online and off-policy using existing learning algorithms. Reward respecting subtasks strongly constrain the space of options and thereby also provide a partial solution to the problem of option discovery. Finally, we show how the algorithms for learning values, policies, options, and models can be unified using general value functions.[1]

Keywords: Options, Model-Based Reinforcement Learning, Subtasks,
Option Learning, Model Learning, Planning.

Acknowledgements

The authors would like to thank Francesco Visin, Joseph Modayil, Martha White, and Michael Bowling for the useful discussions. Richard S. Sutton, Marlos C. Machado and Adam White are supported by a Canada CIFAR AI Chair.

[1]This paper is an extended abstract of the following article: "Sutton, R. S., Machado, M. C., Holland, G. Z., Szepesvari, D., Timbers, F., Tanner, B., White, A. (2022). Reward-Respecting Subtasks for Model-Based Reinforcement Learning. *ArXiv, abs/2202.03466."*

1 Introduction

In model-based reinforcement learning (MBRL), a reinforcement learning agent learns a model of the transition dynamics of its environment and then converts that model into improvements in its policy and, commonly, in its approximate value function. The conversion process, generically referred to as *planning*, is typically computationally expensive and may be distributed over many time steps (or even performed offline). MBRL is well-suited to environments whose transition dynamics are relatively simple but whose policy or value function are complex (such as in Chess and Go). More generally, MBRL may enable dramatically faster adaptation whenever the agent is long-lived, the environment is non-stationary, and much of the environment's transition dynamics are stable (as approximately modeled by the agent).

For planning to be tractable on large problems, an AI agent's model must be abstract in state and time. Abstraction in state is important because the original states of the world may be too numerous to deal with individually, or may not be observable by the agent. In these cases, how the state should be constructed from observations is an important problem on which much work has been done with deep learning (e.g., Mnih et al., 2015). We do not address state abstraction in this paper other than by allowing the agent's state representation to be a non-Markov feature vector.

This paper concerns how we should create and work with environment models that are abstract in *time*. The most common way of formulating temporally-extended and temporally-variable ways of behaving in a reinforcement learning agent is as *options* (Sutton, Precup, & Singh, 1999), each of which comprises a way of behaving (a policy) and a way of stopping. The appeal of options is that they are in some ways interchangeable with actions. Just as we can learn models of action's consequences and plan with those models, so we can learn and plan with models of options' effects. There remains the critical question of where the options come from. A common approach to option discovery is to pose subsidiary tasks such as reaching a bottleneck state or maximizing a sensory signal. Given a subtask, the agent can then develop temporally abstract structure for its cognition by following a progression in which the subtask is solved to produce an option, the option's consequences are learned to produce a model, and the model is used in planning. We refer to this progression (SubTask, Option, Model, Planning) as the STOMP progression for the development of temporally-abstract cognitive structure, which was first described in the the original paper on options.

The conceptual innovation of the current work is to introduce the notion of a *reward-respecting* subtask, that is, of a subtask that optimizes the rewards of the original task until terminating in a state that is sometimes of high value. Reward-respecting subtasks contrast with commonly used subtasks, such as shortest path to bottleneck states (e.g., McGovern & Barto, 2001; Simsek & Barto, 2004), pixel maximization (Jaderberg et al., 2017), and diffusion maximization (c.f. Machado, Barreto, & Precup, 2021), which maximize the cumulative sum of a signal other than the reward of the original task.

For example, consider the two-room gridworld shown inset in Figure 1, with a start state in one room, a terminal goal state in the other, and a hallway state in-between. The usual four actions move the agent one cell up, down, right, or left unless blocked by a wall. A reward of +1 is received on reaching the goal state, which ends the episode. Transitions ending in the gray region between the start and hallway states produce a reward of 1 per step, while all other transitions produce a reward of zero. The discount is = 0:99, so the optimal path from start to goal, traveling the roundabout path that avoids the negative reward, yields a return of $0{:}99^{18}$ 0:83. The hallway state is a bottleneck and thus a natural terminating subgoal for a subtask on this problem (as in Solway et al., 2014). With a reward-respecting subtask, the agent learns a way to the hallway that maximizes the reward along the way; in this gridworld it finds the option that goes down from the start and *around* the negative rewards. In contrast, solving this subtask without taking the reward into consideration leads to the shortest path from start to hallway, passing *through* the negative rewards.

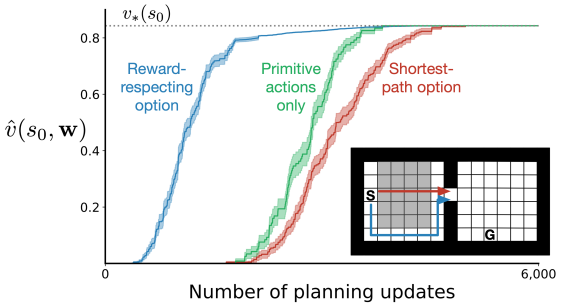

Figure 1: A gridworld example contrasting the progress of planning with different sets of options. Each line is averaged over 100 runs and the shading represents the standard error.

Which of these two options are more useful when their models are used in planning? Assuming the optimal options for both subtasks, and that all the models of the options and actions are known, we can contrast the progress of planning by value iteration with the primitive actions only, and with the primitive actions augmented by the models of the two options. The results, shown in Figure 1, are much as you would expect. Planning using the shortest-path option is less efficient than when using primitive actions only; likely because the option being considered is *not* part of the final solution. Importantly, planning with the model of the option based on the reward-respecting subtask is most efficient.

2 Reward-respecting subtasks

We now define the primary task, GVF subtasks, reward-respecting subtasks, and the reward-respecting subtask used to produce Figure 1. This is the first step in the STOMP progression for developing temporally-abstract cognitive structure.

This, alongside option learning, is the main focus of this extended abstract. In the longer version of this work (c.f. Sutton et al., 2022), we also discuss the model learning and planning steps, with further details on the results presented.

We consider an agent interacting with an environment in a sequence of episodes, each beginning in environment state $S_0 \in \mathcal{S}$ and ending in the terminal state $S_L = ?$ at time step $L \in \mathbb{N}$. At time steps $t < L$, the agent selects an action $A_t \in \mathcal{A}$, and the environment emits a reward $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$ and transitions to a next state $S_{t+1} \in \mathcal{S}^+ = \mathcal{S} \cup \{?\}$ with probability $p(s', r|s, a) \doteq \Pr\{S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a\}$, where $\mathcal{A}$, $\mathcal{S}$, and $\mathcal{R}$ are all finite sets. Capitalized letters denote random variables that differ from episode to episode. The agent's primary task is to find a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ that maximizes the expected discounted sum of rewards, $\mathbb{E}_\pi[R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots + \gamma^{L-1} R_L]$, where $\gamma \in [0, 1)$ is the discount rate.

Solution methods for the primary task often involve approximating the value function $v_\pi : \mathcal{S} \to \mathbb{R}$ for the agent's current policy π, defined by $v_\pi(s) \doteq \mathbb{E}_\pi\left[\sum_{j=1}^{L} \gamma^{j-1} R_j \mid S_0 = s\right], \forall s \in \mathcal{S}$; where the expectation is conditional on the actions being selected according to π. We consider linear approximations to v_π in this work, where each state is converted to a feature vector by a function $\mathbf{x} : \mathcal{S} \to \mathbb{R}^d$ (with $\mathbf{x}(?) \doteq \mathbf{0}$), which might be provided by all the layers of a neural network except for the last, or by a domain expert. Then, $v_\pi(s)$ is approximated as linear in $\mathbf{x}(s)$ and a modifiable weight vector $\mathbf{w} \in \mathbb{R}^d$: $v_\pi(s) \approx \hat{v}(\mathbf{x}(s), \mathbf{w}) \doteq \mathbf{w}^\top \mathbf{x}(s) \doteq \sum_{i=1}^{d} w_i x_i(s), \forall s \in \mathcal{S}$, where w_i and $x_i(s)$ are individual components of $\mathbf{w}$ and $\mathbf{x}(s)$.

We now generalize the primary task to include a full range of tasks for the episodic setting, based on the framework of general value functions (GVFs; Sutton et al., 2011). Instead of maximizing the sum of rewards R_t, we maximize the sum of a cumulant $C_t \doteq c(S_t)$ for some function $c : \mathcal{S} \to \mathbb{R}$. Instead of the accumulation stopping only at the terminal state, it stops at every state with probability $\beta(S_t)$ for some function $\beta : \mathcal{S} \to [0, 1]$ (with $\beta(?) \doteq 1$). Upon stopping, there is an additional contribution to the accumulation, a final *stopping value* $z(S_t)$ for some function $z : \mathcal{S} \to \mathbb{R}$ (with $z(?) \doteq 0$). The functions c, z define the task, and the policy and stopping function, π, β, which together constitute an *option*, define a solution. If option $\langle \pi, \beta \rangle$ were initiated in state S_t, then A_t and subsequent actions would be selected according to π until the option ended, or *stopped*, according to β at step K. Given a GVF task c, z, the objective is to find an option that maximizes $\mathbb{E}_\pi[C_1 + \gamma C_2 + \cdots + \gamma^{K-1} C_K + \gamma^{K-1} z(S_K)]$ for any state S_0 the option started in. That is, to maximize the GVF:

$$v^{c,z}_{\pi,\beta}(s) \doteq \mathbb{E}_{\pi,\beta}\left[\sum_{j=1}^{K} \gamma^{j-1} c(S_j) + \gamma^{K-1} z(S_K) \mid S_0 = s\right], \tag{1}$$

where the expectation is conditional on the actions determined by π and the stopping time K is determined by β. Note that "stopping" is not termination, as it does not affect the actual flow of events in the trajectory. The primary task is a special case of a GVF task in which $C_t \doteq R_t$ and $Z_t \doteq -1, \forall t < L$. Shortest path subtasks are defined by $C_t \doteq -1$ and $z(s) \doteq 0$ at subgoal states and $z(s) \doteq -1$ otherwise. GVF tasks includes all the common subtasks in the literature including (if we allow the state to formally include agent-internal variables) all those based on curiosity and other intrinsic motivations.

Reward-respecting subtasks are GVF tasks with $C_t = R_t$, and whose stopping values take into account the estimated value of the state stopped in. The stopping values cannot be equal to the estimated values, as then the subtask would approximate the primary task (exactly, in the tabular case) and solving it would add nothing new.

In this paper we consider *reward-respecting subtasks of feature attainment*, in which the objective is to maximize an individual feature of the state representation at stopping time while being mindful of the rewards received along the way. We will use the superscript position to indicate the task number. We assume that we can have at most one task per feature, so the feature index i can also serve as a task number. The subtask for maximizing feature i has stopping-value function z^i:

$$z^i(s) \doteq \hat{v}(\mathbf{x}(s), \mathbf{w}) + x_i(s)(w^i - w_i); \tag{2}$$

where $\mathbf{w}$ and its ith component, w_i, are the weights for approximating the primary task, and w^i is a *bonus weight* for subtask i, provided as part of defining the subtask. Note that under the linear form for $\hat{v}$, z^i does not depend on w_i. The quantity $w^i x_i$ is called the *stopping bonus*. Generally, it is only useful to construct subtasks for attaining a feature i if its contribution to value on the primary task, w_i, is sometimes high and sometimes low. If w_i never varied, then its static value could be fully taken into account without planning. If w_i does vary, then its bonus weight should be set to one of its higher values so that an option can be learned in preparation for the occasional times at which w_i is high.

Finally, the reward-respecting subtask used in producing Figure 1 was for the tabular feature for the hallway state, with bonus weight $w^h = 1$, where h denotes the index of the feature for the hallway state. The shortest-path subtask used $C_t = -1$ and stopped upon reaching the hallway or terminal goal states.

3 Option learning for feature attainment

In this section we specify the off-policy learning algorithms we use to approximate the optimal value functions and optimal options. This is the second step in the STOMP progression for developing temporally-abstract cognitive structure.

Define the *optimal value function* $v^i_* : \mathcal{S} \to \mathbb{R}$ for the reward-respecting subtask for attaining the ith feature by (1) with $C_t \doteq R_t$, $z \doteq z^i$, and the π and β that maximize the value. Define the *optimal option* as that pair $\langle \pi, \beta \rangle$. We will describe

these algorithms in a somewhat unusual way that lets us cover all the cases very compactly and uniformly. First we define a general TD (Temporal Difference) error function $\quad : \mathsf{R}^4 \quad [0;1] \; ! \; \mathsf{R}$: $(c; z; v; v'; \;) \doteq c + \quad z + \quad (1 \quad)v' \quad v$:

We use various TD errors to specify our learning algorithms, together with a general update procedure for learning with traces, which we call *UpdateWeights&Traces* (UWT), presented on the right. The first two arguments to UWT are a weight vector and an eligibility-trace vector. These arguments are both inputs and outputs; the same pair are expected to be provided together on every time step. The weight vector is the ultimate result of learning. The eligibility trace is a short-term memory that helps with credit assignment. The third argument is usually a gradient vector with respect to the weight vector. The fourth and sixth arguments to UWT are scalars—the names of the formal arguments are just suggestive of their use. Finally, the fifth argument is a scalar importance-sampling ratio used in off-policy learning (for on-policy learning it should be one).

Procedure UWT $\mathsf{w}; \mathsf{e}; r; \quad ; \; ; \quad (1 \quad)$:
$\mathsf{e} \quad (\mathsf{e} + r)$
$\mathsf{w} \quad \mathsf{w} + \quad \mathsf{e}$
$\mathsf{e} \quad (1 \quad)\mathsf{e}$

Now we show how to use these tools to learn the value functions and options for the subtasks, off-policy. Let $\mathsf{F} \; f0; : : : ; dg$ be the set of features for which we have subtasks. Each subtask will have a weight vector $\mathsf{w}^i \; 2 \; \mathsf{R}^d$ such that $\hat{v}(\mathsf{x}(s); \mathsf{w}^i) \doteq \mathsf{w}^{i\top}\mathsf{x}(s)$ approximates a GVF where $C_t = R_t$, $z = z^i$, $\quad = \; (j; \theta^i_{\cdot})$, where $\theta^i \; 2 \; \mathsf{R}^{d'}$ is another modifiable weight vector, and $\quad = \; ^i$, defined by $\; ^i(s) \doteq 1;$ if $z^i(s) \quad \mathsf{w}^{i\top}_{\cdot}\mathsf{x}(s)$, and $\; ^i(s) \doteq 0$ otherwise. $8s \; 2 \; \mathsf{S}$, with $\; ^i(?) \doteq 1$. To learn off-policy we need the importance sampling ratios $\; ^i_t \doteq \; (A_t|S_t;\theta^i)= \; (A_t|S_t)$, where $\quad : \mathsf{A} \quad \mathsf{S} \; ! \; [0;1]$ is the *behavior policy*. Then, on each time step on which S_t is nonterminal, for each subtask $i \; 2 \; \mathsf{F}$ we do:

$$R_{t+1}; z^i_{t+1}; \hat{v}(\mathsf{x}_t; \mathsf{w}^i); \hat{v}(\mathsf{x}_{t+1}; \mathsf{w}^i); \; ^i_{t+1} \; ; \tag{3}$$

$$\text{UWT} \; \mathsf{w}^i; \mathsf{e}^i; \; r_{\mathsf{w}}\hat{v}(\mathsf{x}_t; \mathsf{w}^i); \quad ; \; ^i_t; \quad (1 \quad ^i_{t+1} \quad ; \text{ and} \tag{4}$$

$$\text{UWT} \; \theta^i; \mathsf{e}^{\theta^i}; \; r_{\theta^i} \ln \; (A_t|S_t; \theta^i); \; ' \; ; \; ^i_t; \quad '(1 \quad ^i_{t+1} \quad ; \tag{5}$$

where here we use the shorthands $z^i_t \doteq z^i(S_t)$ and $\; ^i_t \doteq \; ^i(S_t)$. Under this algorithm, the learned approximate values $\hat{v}(\mathsf{x}(s); \mathsf{w}^i) \doteq \mathsf{w}^{i\top}\mathsf{x}(s)$ come to approximate the *optimal subtask values* $v^i_*(s) \doteq \max_{\; ;} \; v^i_{\; ;}(s)$, $8s \; 2 \; \mathsf{S}$ and $i \; 2 \; \mathsf{F}$, and the options $h \; (j; \theta^i); \; ^i i$ come to approximate the corresponding optimal options.

To learn the options described in Figure 1, we used a behavior policy that selected all four actions with equal probability. The state-feature vectors were 1-hot, producing a tabular representation with $d = 72$. The policy was of the softmax form with linear preferences: $(a|s; \theta) \doteq e^{\theta^\top \phi(s,a)} = \sum_b e^{\theta^\top \phi(s,b)}$, where the state-action feature vectors $\phi(s; a) \; 2 \; \mathsf{R}^{d'}; 8s \; 2 \; \mathsf{S}; a \; 2 \; \mathsf{A}$ were again 1-hot or tabular ($d' = 288$). We ran for 50,000 steps and averaged the results over 100 runs.

4 Multiple options and stochasticity

The results in Figure 1 were mostly to illustrate the idea of reward-respecting subtasks and to serve as a simple example on how to formulate them with the language of GVFs. We now present results for a larger STOMP progression with multiple subtasks in a stochastic environment, and we compare reward-respecting options to other choices. As before, we focus on planning performance here, with the other steps of the STOMP progression being discussed in the longer version of this paper (Sutton et al., 2022). Specifically, we used the four-room episodic gridworld depicted in Figure 2a, with a start state in one room, and a terminal goal state in another. As in the two-room gridworld, a reward of $+1$ is received on reaching the goal, which ends the episode, and passing through the gray region produces a reward of $\quad 1$ per step, while all other transitions produce a reward of zero. The discount factor is $\quad = 0{:}99$. There are four actions, available in all nonterminal states: `up`, `down`, `left`, and `right`. Importantly, each action causes the agent to *stochastically* move in the corresponding direction with probability $2{=}3$ and in one of the other three directions with probability $1{=}9$.

We defined four subtasks, each subtask directed toward reaching one of the four hallways states. The hallways are labeled in Figure 2a as H1-H4. Notice that reward-respecting options take into consideration the fact that the environment's stochasticity is not under the agent's control. The options often take the shortest path to the hallway or the goal state, but the uncontrolled stochasticity and the negative reward region causes some options to prefer a longer (and safer) path. For example, when in the South-West room, options sometimes take the roundabout way to reach H3 and H4. Besides the shortest-path option, we consider *eigenoptions* (Machado et al., 2017; 2018) and the *option-critic* (Bacon, Harb, & Precup, 2017) as alternative option construction methods. The option-critic parameterizes options' policies and uses option-value functions and the policy gradient theorem to learn options that maximize the return, leading to reward-respecting options; but they were never used for planning. Eigenoptions are options designed for temporally-extended exploration that do not take rewards into consideration.

Figure 2b shows that reward-respecting options are more appropriate than other subtask formulations when these options are to be used in the STOMP progression. Moreover, considering multiple options in the STOMP progression does not impact performance. The stochastic nature of the environment does not impact the applicability of our approach.