Discovering Options by Minimizing the Number of Composed Options to Solve Multiple Tasks

Yi Wan
University of Alberta
wan6@ualberta.ca

Richard S. Sutton
University of Alberta, DeepMind
rsutton@ualberta.ca

Abstract

We propose a new objective for option discovery that emphasizes the computational advantage of using options in planning. For a given set of episodic tasks and a given number of options, the objective prefers options that can be used to achieve a high return by composing *few* options. By composing few options, fast planning can be achieved. When faced with new tasks similar to the given ones, the discovered options are also expected to accelerate planning. Our objective extends the objective proposed by Harb et al. (2018) for the single-task setting to the multi-task setting. A closer look at Harb et al.'s objective shows that the best options discovered given one task are not likely to be useful for future unseen tasks and that the multi-task setting is indeed necessary for this purpose. We propose an off-policy algorithm that approximately maximizes the objective. Testing our algorithm in the the four-room domain, we find that 1) a higher objective value is typically associated with options with which fewer planning iterations are needed to achieve near-optimal performance, 2) the best number of planning iterations given the discovered options is much smaller and matches the iteration number given human-designed options, and 3) the options produced by our algorithm also make intuitive sense because they move to and terminate at cells near hallways connecting two neighbor rooms.

Keywords: Reinforcement Learning, Temporal Abstractions, Options

1 Introduction

The options framework (Sutton, Precup, Singh 1999) is a way to achieve temporal abstraction, which is perceived as a cornerstone of artificial intelligence. The options are extended course of actions, defining different ways of behaving. Once the agent has a set of options, it could learn a model that predicts the outcomes of executing options. Planning with option models would require much less computation than planning with action models because options specify jumpy moves.

A natural question to ask is where options come from, which is a challenging active research problem. The first and maybe the most important step towards the option discovery problem is to specify what options should be discovered, which involves defining a metric that can be used to evaluate options. Existing works provide various metrics. For example, some works argue that good options should lead to subgoal states in the environment and the subgoal states could be, for example, bottlenecks in the environment (Bacon 2013). Some other works argue that good options are those such that, when choosing from these options, the agent can achieve high returns (Bacon, Harb, Precup 2017). Among all of the existing metrics, there is one metric (Jinnai et al. 2019) that emphasizes the importance of the role of options in planning. Their work searches for "the smallest set of options so that planning converges in less than a given maximum of value-iteration passes". They also provide a clear analysis of their metric and algorithms to approximate solutions of the metric. However, it is unclear if options that enable fast convergence of value iteration also work well with other planning algorithms.

We propose a simple objective that also emphasizes the importance of options in planning and the discovered options are general. Our objective prefers options with which good solutions to multiple given tasks can be constructed using few options in planning. Using a machine that can perform parallel computation, the speed of constructing a solution using a planning method depends solely on the number of composed options and thus it should be fast for planning to find solutions composed of few options. The multi-task setting is critical to our objective. In fact, our objective reduces to the objective proposed by Harb et al.(2018) if there is only one task. We argue that, with only one task, the best options are specialized to the given task because they solve the entire given task and are typically not useful for future unseen tasks.

To optimize the proposed objective, we propose an off-policy algorithm that approximately maximizes the objective. Empirical study shows that, in the four-room domain 1) the discovered options move to and terminate at cells near hallway cells, 2) the objective value and the number of iterations a classic planning algorithm (value iteration) performs to achieve near-optimal performance are strongly correlated, 3) the number of iterations obtained using the best set of discovered options is the same as the number of iterations obtained using human-designed options.

2 Problem Setting

Our new option discovery objective involves solving a finite set of episodic tasks $\mathcal{I}$, all of which share the same state space $\mathcal{S}$ and action space $\mathcal{A}$. The state and action spaces are finite. For each task i, there is an associated set of terminal states $\perp^i$, which is a set containing one or multiple states from $\mathcal{S}$. The transition dynamics is shared across different tasks, except that starting from a terminal state of a task the agent moves back to the same state regardless of the action. The reward settings for different tasks are typically different. Let $p^i : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ be the transition function of task i with $p^i(s', r \mid s, a) \doteq$ the probability of resulting in state s' and reward r given state-action pair (s, a) and task i.

The agent's interaction with the environment produces a sequence of episodes. The first episode starts from state S_0 sampled from an initial state distribution d_0. A task I_0 is chosen randomly from $\mathcal{I}$ and remains unchanged within the episode. The agent observes S_0 and I_0 and takes an action A_0. The environment then emits task I_0's reward R_1 and the next state S_1 according to transition function p^{I_0}. Such an agent-environment interaction keeps going until the end of the episode when the agent reaches a terminal state. Let the time step at which the first episode terminates T. A new episode starts at $T + 1$, with a new initial state S_{T+1} sampled from d_0 and a new task I_{T+1} sampled from $\mathcal{I}$. The agent may choose its actions following a stationary policy $\pi : \mathcal{A} \times \mathcal{S} \to [0, 1]$ with $\pi : (a \mid s) \doteq$ the probability of choosing action a in state s. Denote the set of all stationary policies Π.

Given a task i, assume that every stationary policy reaches a terminal state with positive probability in at most $\left|\mathcal{S} \setminus \perp^i\right|$ steps, regardless of the initial state. Under this assumption, the value function of a policy $\pi \in \Pi$ in task $i \in \mathcal{I}$, $v^i_\pi(s)$ can defined: $v^i_\pi(s) \doteq \mathbb{E}[R_1 + \cdots + R_T \mid S_0 = s, I_0 = i, A_{0:T-1} \sim \pi]$, where T is a random variable denoting the time step at which the episode terminates. The optimal value function of task i is defined to be: $v^i_*(s) \doteq \max_{\pi \in \Pi} v^i_\pi(s)$. Here the $\max$ always exists. Standard reinforcement learning algorithms like Q-learning can be applied to obtain optimal values for these tasks (Section 5.6 Bertsekas & Tsitsiklis 1996).

Define an option as a pair of a stationary policy and a termination probability. Denote $\mathbf{O}$ as the space of all possible options. That is, $\mathbf{O} \doteq \Pi \times \Gamma$ where Γ denotes the set of termination probabilities $\{f \mid f : \mathcal{S} \to \Delta([0, 1])\}$, where $\Delta(\{0, 1\})$ is the probability simplex over $\{0, 1\}$. Here "1" means terminating and "0" means continuing.

Suppose the agent has a set of k adjustable options $\mathcal{O} \in \mathbf{O}^k$, in addition to $|\mathcal{A}|$ number of primitive actions (non-adjustable options), making it a total of $|\mathcal{A}| + k$ options. The adjustable options can be learned from data. Let $\mathcal{H} \doteq \{1, 2, \ldots, k + |\mathcal{A}|\}$ denote the set of indices of options [1] and $\mathcal{H}^{adj} \doteq \{1, 2, \ldots, k\}$ denote the set of indices of adjustable options. Therefore options with indices $k + 1, \ldots, k + |\mathcal{A}|$ correspond to the primitive actions. Define the set of all functions mapping from $\mathcal{S}$ to $\Delta(\mathcal{H})$ to be $\mathbf{F}$. For each task i, we define an associated *meta-policy* $f^i \in \mathbf{F}$. Let $\mathcal{F} \doteq \{f^i : i \in \mathcal{I}\}$. And thus $\mathcal{F} \in \mathbf{F}^{|\mathcal{I}|}$. Note that meta-policies are designed to choose from indices of options rather than options themselves because the agent's options can change over time and while the indices don't.

Define a function $\pi_{\mathcal{O}} : \mathcal{A} \times \mathcal{S} \times \mathcal{H} \to [0, 1]$ with $\pi_{\mathcal{O}}(a \mid s, h) \doteq$ the probability action a is taken at state s given option o_h, $a \in \mathcal{A}, s \in \mathcal{S}, h \in \mathcal{H}$. and a function $\beta_{\mathcal{O}} : \mathcal{S} \times \mathcal{H} \to [0, 1]$ with $\beta_{\mathcal{O}}(s, h) \doteq$ the probability action a is taken at state s given option o_h, $\quad s \in \mathcal{S}, h \in \mathcal{H}$. Note that the termination probabilities for non-adjustable options (primitive actions) are 1. That is $\beta_{\mathcal{O}}(s, h) = 1, \forall s \in \mathcal{S}, \forall h \in \{k + 1, \cdots, k + |\mathcal{A}|\}$. The policy for these options follows the corresponding actions deterministically. That is, $\pi_{\mathcal{O}}(a \mid s, h) = \mathbf{1}_{a=a_h}, \forall s \in \mathcal{S}, h \in \{k + 1, \cdots, k + |\mathcal{A}|\}, a \in \mathcal{A}$.

In order to chose actions, the agent could choose to execute its options. Such a way of behaving is called *call-and-return*. In particular, suppose the task i is chosen and the agent has a set of options $\mathcal{O}$, it first picks an option's index h from $\mathcal{H}$ according to policy f^i, and then executes the index option o_h until the option terminates, at which it chooses a new option. The process keeps going until the terminal state $\perp^i$ is reached, marking the end of the episode. Define $q^i_{\mathcal{O},\mathcal{F}}(s, h, a)$ to be the expected cumulative reward with a set of options $\mathcal{O}$ and a set of policies $\mathcal{F}$ if the agent starts from state s, chooses option o_h, and action a, and behaves in the call-and-return fashion. Let $q^i_{\mathcal{O},\mathcal{F}}(s, h) \doteq \sum_a \pi_{\mathcal{O}}(a \mid s, h) q^i_{\mathcal{O},\mathcal{F}}(s, h, a)$, and $v^i_{\mathcal{O},\mathcal{F}}(s) \doteq \sum_h f^i(h \mid s) q^i_{\mathcal{O},\mathcal{F}}(s, h)$. It can be seen that the best achievable value when choosing and following options is the optimal value. That is, $\max_{\mathcal{O} \in \mathbf{O}^k, \mathcal{F} \in \mathbf{F}^{|\mathcal{I}|}} v^i_{\mathcal{O},\mathcal{F}}(s) = v^i_*(s), \forall i \in \mathcal{I}, s \in \mathcal{S}$. This equation holds because any pair $(\mathcal{O}, \mathcal{F})$ defines a non-stationary policy flat policy (policy over primitive actions). We know for any non-stationary flat policy π, $v^i_\pi(s) \leq v^i_*(s), \forall i, s$ (Puterman 1994). Also, $\mathcal{F}$ contains all policies that only choose from primitive actions and thus must contain an optimal policy, thus $\max_{\mathcal{O},\mathcal{F}} v^i_{\mathcal{O},\mathcal{F}}(s) = v^i_*(s), \forall i \in \mathcal{I}, s \in \mathcal{S}$.

3 The New Objective

Intuitively, the new objective is the weighted average of the average return achieved by a certain way of composing options and the negative of the average number of the composed options. We start by formally defining the objective. We then explain what options would be preferred under our objective and why planning with these options are faster.

Define $\tilde{q}^i_{\mathcal{O},\mathcal{F}}(s, h, a)$ to be the negative expected cumulative number of option terminations given a set of options $\mathcal{O}$ and a set of policies $\mathcal{F}$ if the agent starts from state s, chooses option o_h, and action a, and behaves in the call-and-return fashion. Let $\tilde{q}^i_{\mathcal{O},\mathcal{F}}(s, h) \doteq \sum_a \pi_{\mathcal{O}}(a \mid s, h) \tilde{q}^i_{\mathcal{O},\mathcal{F}}(s, h, a)$, and $\tilde{v}^i_{\mathcal{O},\mathcal{F}}(s) \doteq \sum_h f^i(h \mid s) \tilde{q}^i_{\mathcal{O},\mathcal{F}}(s, h)$. We see that $-\tilde{v}^i_{\mathcal{O},\mathcal{F}}(s) + 1$ is the expected number of composed options to solve task i starting from s, given $\mathcal{O}$ and $\mathcal{F}$. We propose to maximize the following objective w.r.t. $\mathcal{O}, \mathcal{F}$: $J(\mathcal{O}, \mathcal{F}, c) \doteq \sum_i \sum_s d_0(s) v^i_{\mathcal{O},\mathcal{F}}(s) + c \sum_i \sum_s d_0(s) \tilde{v}^i_{\mathcal{O},\mathcal{F}}(s)$, where $c > 0$ is a problem parameter. The objective is a sum of two parts. The first part is the average value over all tasks. The second part is the number of option terminations following policies $\mathcal{F}$, again averaged over all tasks. The problem parameter c specifies one's relative interest in how fast solutions to the given tasks can be obtained in planning over the performance of these solutions.

Remark: If there is only one task, our objective J reduces to the objective proposed by Harb et al. (2018). However, with only one task, the best strategy is to learn an option that solves the entire task and to always choose that option so that no termination cost will be incurred. On the contrary, if there are multiple tasks and the number of tasks is more than the number of options, the agent can not assign for each task an option, and learned options would better be *overlapped part* of the solutions to different tasks. The next example illustrates this point.

Example. There are four episodic tasks in the two-room domain shown in Figure 1. Each green cell marks a termination state of a task. All rewards are -1. Each episode starts from a state randomly picked from the left room. The agent has four primitive actions (left, right, up, down) and one adjustable option. The best set of options consists of four primitive actions and the option marked in the figure. For states in the left room and the hallway states, actions taken by the option's policy are marked by arrows. For states in the right room, the actions being chosen by the option can be arbitrary because the option will not be chosen for those states in optimal solutions of the tasks. The option terminates deterministically in yellow cells. Note that the option is shared by all four tasks. Also, note that if the option terminates in the hallway cell, one more primitive termination would be incurred in cell A in the solutions to all four tasks. And if the option was not terminated in cell A, no matter what action is assigned to this cell, the action is not optimal for some tasks.

[1] Throughout the paper, all sets are indexed and given a set $\mathcal{X}$, we use the notation x_i to denote the i-th element of $\mathcal{X}$.

Figure 2 helps understand the relation between the objective J, defined over a set of *training* tasks, and the number of planning iterations used to achieve near-optimal performance in a set of *testing* tasks that are similar to the training tasks. The environment, training tasks, and testing tasks used to produce this figure are introduced in the next section. Each blue dot corresponds to a set of options and a set of policies, which are obtained by running our new algorithm (MCOM) introduced in the next section with different parameter settings. The x-axis is the estimate of the objective value, computed by averaging compound returns (a compound return of an episode is the difference between the episodic return and the number of option terminations in the episode) of 500 episodes. The y-axis is the number of iterations required to obtain near-optimal solutions in all testing tasks. As a baseline, we plotted a yellow dot, corresponding to the number of iterations and the best average compound return when the set of options consists of two human-designed hallway options and primitive actions and plot in red the number of iterations required given only primitive actions. It can be seen that the estimate of objective value and the number of iterations are negatively correlated – higher estimate of the objective value for training tasks is typically associated with fewer planning iterations for testing tasks.

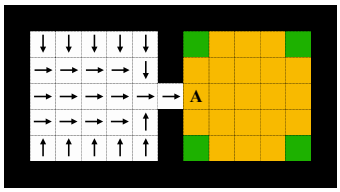


Figure 1: Illustration of a good set of options under the our objective.

In order to maximize the objective, a common way is to apply the gradient ascent algorithm. Suppose that adjustable options' termination probabilities are parameterized by θ_β and their policies are parameterized by θ_π. In this case, $\mathcal{O}$ is parameterized by θ_π and θ_β. With this in mind, we omit $\mathcal{O}$ in $\pi_\mathcal{O}$ and $\beta_\mathcal{O}$ for simplicity. Let $\bar{v}^i_{\mathcal{O},\mathcal{F}}(s) \doteq v^i_{\mathcal{O},\mathcal{F}}(s) + c\tilde{v}^i_{\mathcal{O},\mathcal{F}}(s), \forall s \in \mathcal{S}$. Similarly, we define $\bar{q}^i_{\mathcal{O},\mathcal{F}}(s,h) \doteq q^i_{\mathcal{O},\mathcal{F}}(s,h) + c\tilde{q}^i_{\mathcal{O},\mathcal{F}}(s,h), \forall s \in \mathcal{S}, h \in \mathcal{H}$ and $\bar{q}^i_{\mathcal{O},\mathcal{F}}(s,o,a) \doteq q^i_{\mathcal{O},\mathcal{F}}(s,h,a) + c\tilde{q}^i_{\mathcal{O},\mathcal{F}}(s,h,a), \forall s \in \mathcal{S}, h \in \mathcal{H}, a \in \mathcal{A}$. The gradient w.r.t. J is given in the following proposition.

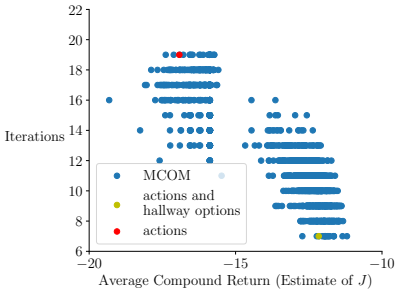


Figure 2: Relation between the average compound return (estimate of the objective with $c = 1$) in a set of training tasks and the number of iterations used to achieve near-optimal performance in a set of testing tasks.

Proposition 3.1 (Gradient of J). *Assuming that $\frac{\partial J(\mathcal{O},\mathcal{F},c)}{\partial \theta_\pi}$ and $\frac{\partial J(\mathcal{O},\mathcal{F},c)}{\partial \theta_\beta}$ exist, we have*

$$\frac{\partial J(\mathcal{O},\mathcal{F},c)}{\partial \theta_\pi} = \sum_i \sum_{s,h} d^i_{\mathcal{O},\mathcal{F}}(s,h) \sum_a \frac{\partial \pi(a \mid s,h)}{\partial \theta_\pi} \bar{q}^i_{\mathcal{O},\mathcal{F}}(s,h,a),$$

$$\frac{\partial J(\mathcal{O},\mathcal{F},c)}{\partial \theta_\beta} = -\sum_i \sum_{h,s'} d'^i_{\mathcal{O},\mathcal{F}}(h,s') \frac{\partial \beta(s',h)}{\partial \theta_\beta} (\bar{q}^i_{\mathcal{O},\mathcal{F}}(s',h) - \bar{v}^i_{\mathcal{O},\mathcal{F}}(s') + c).$$

Here $d^i_{\mathcal{O},\mathcal{F}}(s,h)$ is the number of time steps on average option o_h is used in state s in a single episode and $d'^i_{\mathcal{O},\mathcal{F}}(h,s')$ is the number of time steps on average option o_h is used to reach state s' in a single episode.

The proof is essentially the same as the proof of Theorem 5.2 by Bacon (2018) and is omitted.

4 The New Algorithm

In this section, we introduce a new algorithm that maximizes J. We call our algorithm MCOM to emphasize its *minimizing the number of composed options to solve multiple tasks*. MCOM is an off-policy algorithm and thus its behavior policy is not necessarily the same as its target policy. The agent's behavior is defined as follows. Each time step t, the agent observes task I_t and state S_t, and takes action A_t according to a behavior policy $b^{I_t}_t(\cdot \mid S_t)$. For simplicity, let Z_{t+1} denote the termination signal of the episode. That is, $Z_{t+1} = 0$ if the current episode continues ($S_{t+1} \notin \perp^{I_t}$) and $Z_{t+1} = 1$ if the the current episode is terminates ($S_{t+1} \in \perp^{I_t}$).

MCOM maintains an $|\mathcal{I}| \times |\mathcal{S}| \times |\mathcal{H}|$ table of option-value estimate Q and a $|\mathcal{S}| \times \left|\mathcal{H}^{adj}\right|$ table of termination preferences W^β used compute termination probabilities of adjustable options. Given W^β, an adjustable option $h \in \mathcal{H}^{adj}$, the probability of terminating option o_h at s is obtained by applying the sigmoid function to the preference: $\beta_\mathcal{O}(s,h) \doteq \text{sigmoid}(W^\beta(s,h))$. The algorithm also maintains a $|\mathcal{S}| \times \left|\mathcal{H}^{adj}\right| \times |\mathcal{A}|$ table of policy preferences W^π used to compute policies of adjustable options. Given W^π, a state s, and an adjustable option $h \in \mathcal{H}^{adj}$, the probabilities of taking different actions of option o_h at state s are obtained by applying the softmax function to the preferences of the corresponding actions. $\pi_\mathcal{O}(a \mid s,h) \doteq \frac{e^{W^\pi(s,h,a)}}{\sum_{\bar{a}} e^{W^\pi(s,h,\bar{a})}}$. The set of options $\mathcal{O}$ is parameterized by W^π and W^β. With this in mind, we omit the superscript $\mathcal{O}$ in $\pi_\mathcal{O}$ and $\beta_\mathcal{O}$ for simplicity.

At time step t, our algorithm updates an adjustable option indexed by X_t [2], which is chosen in the following way. With $1-\zeta$ probability, the adjustable option is the *best* adjustable option in state S_t for task I_t ($X_t = \text{argmax}_{h \in \mathcal{H}^{adj}} Q_t^{I_t}(S_t, h)$) and with ζ probability, it is a randomly picked adjustable option ($X_t \sim \text{Uniform}(\{1, 2, \ldots, k\})$). Here $1 \geq \zeta \geq 0$ is a small scalar. At time step $t+1$, MCOM updates W^π using

$$W^\pi_{t+1}(S_t, X_t, a) \doteq W^\pi_t(S_t, X_t, a) + \alpha \rho_t(x) \left((\mathbf{1}_{a=A_t} - \pi(a \mid S_t, X_t))\delta_t(x) + \eta \frac{\partial H(\pi(\cdot \mid S_t, X_t))}{\partial W^\pi(a \mid S_t, X_t)} \right), \forall a \in \mathcal{A}$$

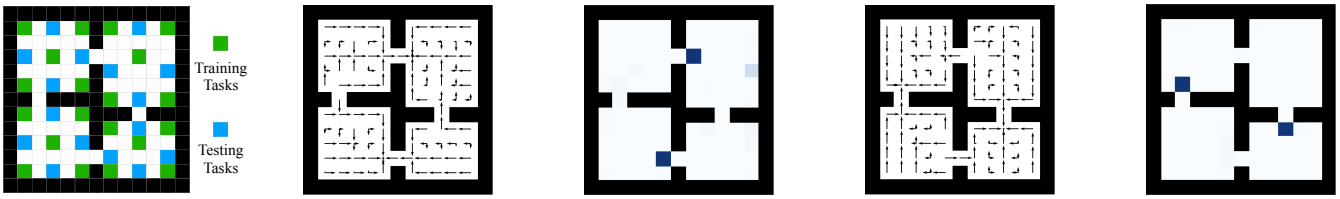
where α is the stepsize, $H(\pi(\cdot \mid s, h))$ is the entropy of $\pi(\cdot \mid s, h)$, and $\frac{\partial H(\pi(\cdot|s,h))}{\partial W^\pi(s,h,a)} = -\pi(a \mid s, h)(\log \pi(a \mid s, h) + H(\pi(\cdot \mid s, h))$ is the partial derivative of the entropy, $\delta_t(X_t) \doteq R^{I_t}_{t+1} + U^{I_t}_t(S_{t+1}, X_t) - Q^{I_t}_t(S_t, X_t)$ is the temporal difference (TD) error with $U^{I_t}_t(S_{t+1}, X_t) \doteq (1 - Z_{t+1})(\beta(S_{t+1}, X_t)(V^{I_t}_t(S_{t+1}) - c) + (1 - \beta(S_{t+1}, X_t))Q^{I_t}_t(S_{t+1}, X_t))$, where $V^{I_t}_t(S_{t+1}) \doteq \max_x Q^{I_t}_t(S_{t+1}, x)$, and $\rho_t(X_t) \doteq \frac{\pi(A_t|S_t,X_t)}{b^{I_t}_t(A_t|S_t)}$ is the importance sampling ratio. The algorithm updates W^β using

$$W^\beta_{t+1}(S_{t+1}, X_t) \doteq W^\beta_t(S_{t+1}, X_t) - \alpha \rho_t(x)(1 - Z_{t+1}) \left(\beta(S_{t+1}, X_t)(1 - \beta(S_{t+1}, X_t))A^{I_t}_t(S_{t+1}, X_t) - \eta \frac{\partial H([\beta(S_{t+1}, X_t), 1 - \beta(S_{t+1}, X_t)])}{\partial W^\beta(S_{t+1}, X_t)} \right),$$

where $A^{I_t}_t(s, h) \doteq Q^{I_t}_t(s, h) - V^{I_t}_t(s) + \bar{c}$ is the advantage of continuing (not terminating) the option indexed by h at state s, $\bar{c}$ is a solution parameter balancing the low termination cost over high average return and can be different from the problem parameter c, $H([\beta(s, x), 1 - \beta(s, x)])$ is the entropy of option o_x's termination probability at state s, and $\frac{\partial H([\beta(s,x),1-\beta(s,x)])}{\partial W^\beta(s,x)} = \beta(s, x)(1 - \beta(s, x)) \log \frac{(1-\beta(s,x))}{\beta(s,x)}$. The algorithm updates value estimates for all options using

$$Q^{I_t}_{t+1}(S_t, x) \doteq Q^{I_t}_t(S_t, x) + \alpha \rho_t(x) \delta_t(x), \forall x \in \mathcal{H} \tag{1}$$

We now present an experiment testing our new algorithm. In this experiment, the agent interacts with a four-room grid world with 16 tasks (Figure 3a). The agent uses a step-size 0.01 for all updates. $c = 1$ and $\epsilon = 0.1$. The agent takes a random action at every time step. That is, $b(\cdot \mid \cdot) = 1/4$. The agent is trained for 3×10^8 steps and we show the learned two options' policies and termination probabilities also in Figure 3.



(a) Four-room domain (b) π of option 1 (c) β of option 1 (d) π of option 2 (e) β of option 2

Figure 3: (a) Illustration of the four-room domain. Green cells mark terminal states of different tasks. All rewards are -1 and the discount factor is 1. (b) The learned policy of the first option. The dark blue cell in the bottom left room marks the terminal state of the current episode. The learned policy moves the agent in the clockwise direction. (c) The learned termination probability of the first option, the option terminates near the hallway in all four rooms. This observation also applies to the second option as shown in (e). (d) The learned policy of the second option moves the agent in the counter-clockwise direction.

We can see that the two options have two different policies: option 1 moves clockwise while option 2 moves counter-clockwise. Second, the termination probabilities of the two options seem to concentrate near the hallway cells. This observation makes intuitive sense because just like the example shown in Figure 1, a good set of options should move the agent to different hallways and terminate afterward.

References

Bacon, P. L. (2013). On the bottleneck concept for options discovery. Ph. D. dissertation, Masters thesis.

Bacon, P. L., Harb, J., & Precup, D. (2017, February). The option-critic architecture. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 31, No. 1).

Harb, J., Bacon, P. L., Klissarov, M., & Precup, D. (2018, April). When waiting is not an option: Learning options with a deliberation cost. In Thirty-Second AAAI Conference on Artificial Intelligence.

Jinnai, Y., Abel, D., Hershkowitz, D., Littman, M., & Konidaris, G. (2019, May). Finding options that minimize planning time. In International Conference on Machine Learning (pp. 3120-3129). PMLR.

[2] We use X_t to denote the index of the option being updated by our algorithm at time t and use H_t to denote the option being executed at time t if the agent's behavior is call-and-return.