# Abstract Value Iteration for Hierarchical Reinforcement Learning

**Kishor Jothimurugan**
University of Pennsylvania

**Osbert Bastani**
University of Pennsylvania

**Rajeev Alur**
University of Pennsylvania

## Abstract

We propose a novel hierarchical reinforcement learning framework for control with continuous state and action spaces. In our framework, the user specifies subgoal regions which are subsets of states; then, we (i) learn options that serve as transitions between these subgoal regions, and (ii) construct a high-level plan in the resulting abstract decision process (ADP). A key challenge is that the ADP may not be Markov, which we address by proposing two algorithms for planning in the ADP. Our first algorithm is conservative, allowing us to prove theoretical guarantees on its performance, which help inform the design of subgoal regions. Our second algorithm is a practical one that interweaves planning at the abstract level and learning at the concrete level. In our experiments, we demonstrate that our approach outperforms state-of-the-art hierarchical reinforcement learning algorithms on several challenging benchmarks.

## 1 INTRODUCTION

Deep reinforcement learning (RL) has recently been applied to solve challenging robotics control problems, including multi-agent control (Khan et al., 2019), object manipulation (Andrychowicz et al., 2020), and control from perception (Levine et al., 2016). In these applications, the approach is typically to learn a policy in simulation and then deploy this policy on an actual robot. Our focus is on the problem of using RL to learn a robot control policy in simulation.

A key challenge in this setting is that long-horizon tasks are often computationally intractable for RL, at best requiring huge amounts of computation to solve (Andrychowicz et al., 2020). Hierarchical RL is a promising approach to scaling RL to long-horizon tasks. The idea is to use a high-level policy to generate a sequence of high-level goals, and then use low-level policies to generate sequences of actions to achieve each successive goal. By abstracting away details of the low-level dynamics, the high-level policy can efficiently plan over much longer time horizons.

There are two approaches to designing the high-level policy. First, we can use model-free RL to learn the high-level policy (Nachum et al., 2018, 2019). While this approach is very general, it cannot take advantage of the available structure in the high-level planning problem. Alternatively, we can use model-based RL—i.e., learn a model of the high-level planning problem and then plan in this model. This approach can significantly improve performance by leveraging high-level structure. However, they are typically restricted to finite state and action spaces (Gopalan et al., 2017; Abel et al., 2020; Winder et al., 2020); at best, they can handle continuous state spaces but finite action spaces (Roderick et al., 2018).

We propose a hierarchical RL algorithm using model-based RL for high-level planning that can handle continuous state and action spaces. To ensure that our model of the high-level problem is finite, we abstract over both states and actions. First, to abstract over states, we consider *subgoal regions* that are subsets of states; intuitively, they should aggregate states with similar transition probabilities and rewards. Subgoal regions are similar to abstract states but do not need to cover the entire state space (Dietterich, 2000; Andre and Russell, 2002). Next, to abstract over actions, we consider *options* (also called *abstract actions*, *temporal abstractions*, or *skills*), which are low-level policies designed to achieve short-term goals such as walking to a goal or grasping an object (Precup et al., 1998; Sutton et al., 1999; Theocharous and Kaelbling, 2004). Intuitively, subgoal regions finitize the state space and options finitize the action space. Finally, our algorithm represents the high-level planning problem as an *abstract decision process (ADP)* whose states are

Proceedings of the 24[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130. 

subgoal regions and whose actions are options.

One question is how to obtain the subgoal regions and options. Similar to previous works (Gopalan et al., 2017; Winder et al., 2020; Abel et al., 2020), which assume that the state abstractions are provided by the user, we assume that the subgoal regions are given by the user. Given subgoal regions, our algorithm uses model-free RL to automatically train options that serve as transitions between these subgoal regions.

We believe domain experts can often provide effective choices of subgoal regions; thus, our approach gives the user a way to express domain knowledge to improve performance. Furthermore, many recent RL algorithms ask the user to provide significantly more information to improve performance—e.g., a high-level plan for solving the task (Sun et al., 2019; Jothimurugan et al., 2019). Finally, in the spirit of probabilistic road maps (LaValle, 2006), we consider automatically constructing subgoal regions by randomly sampling them; we find that this approach has higher sample complexity but still achieves good reward.

A challenge is that the ADP may not be an MDP—i.e., it may not satisfy the Markov condition that transition probabilities and rewards only depend on the current subgoal region. In particular, an option may not work equally well for different states in a single subgoal region, and thus, the transition probabilities depend on the state, which violates the Markov condition.

We propose two algorithms for planning in the ADP that address this challenge. First, *robust abstract value iteration (R-AVI)* models the unknown perturbations adversarially. This algorithm is designed so we can theoretically characterize the properties of our approach. In particular, we establish bounds on its performance and discuss how these bounds can guide the design of subgoal regions that yield good performance in the context of our approach. Second, *alternating abstract value iteration (A-AVI)* does not model the unknown perturbations. To account for the non-Markov nature of the ADP, it alternates between (i) planning in the ADP to construct a high-level policy, and (ii) updating its estimates of the ADP transition probabilities and rewards based on the current high-level policy. This algorithm is designed to be practical; it does not have theoretical guarantees but performs well in practice.

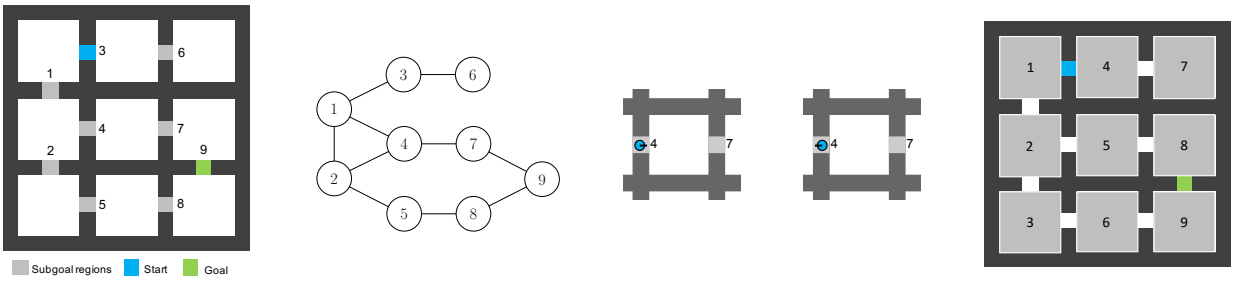
We demonstrate that our approach outperforms several state-of-the-art baselines, including approaches that solve the ADP without accounting for uncertainty (Winder et al., 2020), HIRO (which does not require user-provided subgoal regions) (Nachum et al., 2018), and SpectRL (which requires user-provided information that is more complex than subgoal regions) (Jothimurugan et al., 2019), on both a robot navigating a maze of rooms as well as the MuJoCo Ant performing sequences of tasks (Todorov et al., 2012; Nachum et al., 2018).

**Illustrative example.** Consider the example in Figure 1 (a). The goal is for the robot to drive from an initial state (in the blue square region) to a goal region (the green square). The states are $S \subseteq \mathbb{R}^3$, where $s = (x, y, \theta)$ encodes the $(x, y)$ position of the robot and its orientation $\theta$. The actions are $A \subseteq \mathbb{R}^2$, where $a = (v, \phi)$ encodes its speed $v$ and steering angle $\phi$. The reward is 1 upon reaching the goal region and 0 otherwise. The user provides the subgoal regions; in this example, they are the doorway regions that connect the rooms (gray squares). These subgoal regions are designed to satisfy the conditions based on our theoretical analysis of R-AVI: (i) these regions are bottlenecks, and (ii) the transition probabilities and rewards are similar across states in a subgoal region. Our algorithm uses model-free RL to learn low-level policies that serve as transitions between adjacent subgoal regions; the resulting ADP is shown in Figure 1 (b). The high-level policy we construct is $3 \to 1 \to 4 \to 7 \to 9$.

Next, (c,d) show why the ADP is not Markov. We assume the robot cannot move backward. If it starts in state (c), then it can easily take the transition $4 \to 7$. However, if it starts in state (d), then it is much more costly for it to take $4 \to 7$. One might hope to resolve this issue by including a distinct subgoal region for every heading $\theta$; however, there would then be infinitely many subgoal regions since $\theta$ is continuous. Another example of why the ADP is not Markov is that the robot is more likely to run into a wall and incur a negative reward near the boundary of a subgoal region than in the interior. In general, the ADP is only Markov if the transition probabilities and rewards for *all* options are *exactly the same* for *all* states in a subgoal region.

In our experiments, we show that an ablation that ignores the fact that the ADP is not Markov performs poorly. We also show that HIRO (Nachum et al., 2018) and SpectRL (Jothimurugan et al., 2019), which use model-free RL for high-level planning, perform poorly since they do not systematically explore in the ADP.

**Related work.** There has been work on planning with options (Precup et al., 1998; Sutton et al., 1999; Theocharous and Kaelbling, 2004), including in the setting of deep RL (Bacon et al., 2017; Tiwari and Thomas, 2019). There has been work on leveraging action abstractions in the setting of deep RL (Kulkarni et al., 2016; Nachum et al., 2018, 2019). For instance, Co-Reyes et al. (2018) propose a hierarchical RL algorithm that uses model-based RL for high-level




(a) (b) (c) (d) (e)

Figure 1: (a) A rooms environment; subgoal regions are in light gray, the starting region is blue, and the goal region is green. (b) The corresponding abstract graph; transitions are bi-directional. (c, d) Robot in region 4 facing right vs. left. (e) A different choice of subgoal regions.

planning, but they do not leverage state abstractions. More closely related, there has been work on leveraging state abstractions, typically in conjunction with action abstractions (Dietterich, 2000; Andre and Russell, 2002; Theocharous et al., 2005; Li et al., 2006; Gopalan et al., 2017; Choudhury et al., 2019; Winder et al., 2020; Abel et al., 2020). For instance, Gopalan et al. (2017) propose an algorithm for planning in hierarchical MDPs, but assume that the abstract MDPs are given and furthermore satisfy the Markov property. There has been subsequent work that uses model-based reinforcement learning to learn both the concrete and abstract MDPs (Winder et al., 2020); however, they also do not account for the fact that the abstract MDP may not satisfy the Markov condition, and their approach is furthermore limited to finite state MDPs. The most closely related work is Abel et al. (2020), which analyzes how the failure of the Markov property affects planning in the abstract MDP. However, their algorithm still performs planning with respect to the concrete states; thus, their approach scales poorly with the size of the state space, and furthermore cannot be applied to continuous state spaces. Finally, there has been work on performing value iteration with upper/lower bounds (Givan et al., 2000); however, this approach only applies to finite MDPs and furthermore is not designed to handle action abstractions.

There has been work on inferring options, by transferring options to new domains (Konidaris and Barto, 2007), inferring options in multi-task reinforcement learning (Stolle and Precup, 2002; Konidaris and Barto, 2009; Machado et al., 2017; Finn et al., 2017; Eysenbach et al., 2018), from demonstrations (Hausman et al., 2018), or using expectation-maximization (Daniel et al., 2016). Similarly, there has been interest in planning by composing low-level skills (Burridge et al., 1999; Majumdar and Tedrake, 2017). In contrast, our approach constructs action abstractions from user-provided state abstractions; thus, our approach has the benefit of not requiring additional information such as related tasks for learning. There has also been interest in inferring state abstractions (Ferns et al., 2004; Jong and Stone, 2005; Abel et al., 2019), by transferring them to new domains (Walsh et al., 2006), from demonstrations (Cobo et al., 2011), and from options (Jonsson and Barto, 2001; Konidaris et al., 2014). There has been work on inferring state abstractions (Ferns et al., 2004; Taylor et al., 2009; Taïga et al., 2018; Castro, 2019) and options (Castro and Precup, 2011) by measuring state similarity in terms of reward and transition properties, but only for finite MDPs.

## 2 PROBLEM FORMULATION

**Background.** A Markov decision process (MDP) is a tuple $(\mathcal{S}, \mathcal{A}, T, R, \gamma, \eta_0)$, where $\mathcal{S} \subseteq \mathbb{R}^n$ are the states, $\mathcal{A} \subseteq \mathbb{R}^m$ are the actions, $T(s, a, s') = p(s' \mid s, a) \in \mathbb{R}$ is the probability density of transitioning from $s$ to $s'$ on action $a$, $R(s, a) \in [0, 1]$ is the reward for action $a$ in state $s$, $\gamma \in [0, 1)$ is the discount factor, and $\eta_0$ is the initial state distribution. A (deterministic) policy is a function $\pi : \mathcal{S} \to \mathcal{A}$, where $a = \pi(s)$ is the action to take in state $s$. The value of a state $s$ under policy $\pi$ is denoted by $V^\pi(s)$ and the optimal policy is $\pi^* = \arg\max_\pi J(\pi)$, where $J(\pi) = \mathbb{E}_{s_0 \sim \eta_0}[V^\pi(s_0)]$ is the expected reward under policy $\pi$; we let $V^* = V^{\pi^*}$.

An *option* $o$ is a tuple $(\pi, I, \beta)$, where $\pi$ is a policy, $I \subseteq \mathcal{S}$ is a set of initial states from which $\pi$ can be used, and $\beta : \mathcal{S} \to [0, 1]$ is the termination probability (Sutton et al., 1999). A set of options $\mathcal{O}$ defines a *multi-time model* $(\mathcal{S}, \mathcal{O}, T_{\text{opt}}, R_{\text{opt}})$ (Sutton et al., 1999), where for $s, s' \in \mathcal{S}$ and $o = (\pi, I, \beta) \in \mathcal{O}$,

$$T_{\text{opt}}(s, o, s') = \sum_{t=1}^{\infty} \gamma^t p(s' \mid t, s, o) P(t \mid s, o)$$

is the time-discounted probability density of transitioning from $s$ to $s'$ when using option $o$, where

$P(t \mid s, o)$ is the probability that $o$ terminates after $t$ steps when starting from $s$, and $p(s' \mid t, s, o)$ is the probability density of $o$ terminating in $s'$ when starting from $s$ given that it terminates after $t$ steps. The expected reward before termination using $o$ from $s$ is

$$R_{\text{opt}}(s, o) = \mathbb{E}_{s_0, a_0, \ldots, s_t \sim o}\left[\sum_{i=0}^{t-1} \gamma^i R(s_i, a_i) \mid s_0 = s\right],$$

where $t$ is the random time at which $o$ terminates when started at $s$. A (deterministic) *option policy* $\rho : \mathcal{S} \to \mathcal{O}$ maps each state $s$ to an option $(\pi, I, \beta) = \rho(s)$ with $s \in I$, to use starting from $s$; $\rho$ induces a policy[1] $\pi_\rho$ for the underlying MDP. The *optimal option policy* is $\rho^*(s) = \arg\max_{o \in \mathcal{O}} Q^*_{\mathcal{O}}(s, o)$, where $Q^*$ is defined by the Bellman equations (Sutton et al., 1999):

$$V^*_{\mathcal{O}}(s) = \max_{o \in \mathcal{O}} Q^*_{\mathcal{O}}(s, o), \tag{1}$$

$$Q^*_{\mathcal{O}}(s, o) = R_{\text{opt}}(s, o) + \int_{\mathcal{S}} T_{\text{opt}}(s, o, s') V^*_{\mathcal{O}}(s') ds'.$$

We can use these equations in conjunction with option value iteration to compute $V^*_{\mathcal{O}}$ and $Q^*_{\mathcal{O}}$.

**Problem formulation.** We assume we have access to simulation of the concrete MDP $\mathcal{M}$—i.e., we can obtain samples $s' \sim p(\cdot \mid s, a)$ from the transitions $T$ using any concrete action $a \in \mathcal{A}$ from any concrete state $s \in \mathcal{S}$. We also assume we are given a finite set of subgoal regions $\tilde{\mathcal{S}}$, where each $\tilde{s} \in \tilde{\mathcal{S}}$ is a subset of concrete states $\tilde{s} \subseteq \mathcal{S}$. We assume they are disjoint—i.e., $\tilde{s} \cap \tilde{s}' = \varnothing$ if $\tilde{s} \neq \tilde{s}'$.[2] Subgoal regions are similar to abstract states except they do not need to cover the state space of $\mathcal{M}$. Intuitively, they should include all subgoals that an optimal policy might need to reach to achieve the goal. In particular, given the subgoal regions, our algorithm only considers options such that (i) their initial set is a subgoal region, and (ii) they terminate upon entering any other subgoal region—i.e., options serve as transitions between different subgoal regions.

**Definition 2.1.** Given subgoal regions $\tilde{\mathcal{S}}$, an option $o = (\pi, I, \beta)$ is a *subgoal transition* if $I = \tilde{s}$ for some $\tilde{s} \in \tilde{\mathcal{S}}$ and $\beta(s) = \mathbb{1}(s \in \bar{S} \setminus \tilde{s})$, where $\bar{S} \subseteq \mathcal{S}$ is the union of all subgoal regions—i.e., $\bar{S} = \bigcup_{\tilde{s} \in \tilde{\mathcal{S}}} \tilde{s}$.

We also assume we are given a set of edges $E \subseteq \tilde{\mathcal{S}} \times \tilde{\mathcal{S}}$ used to learn the subgoal transitions; by default, we can take $E = \tilde{\mathcal{S}} \times \tilde{\mathcal{S}}$. These edges are used to constrain the number of subgoal transitions—i.e., we only learn options that serve as transitions between $(\tilde{s}, \tilde{s}') \in E$. We denote by $\tilde{s}_0 \in \tilde{\mathcal{S}}$ the initial region and assume that the initial state distribution $\eta_0$ assigns zero probability to $\mathcal{S} \setminus \tilde{s}_0$.

Finally, part of our analysis considers the special case of reachability problems in deterministic MDPs with sparse rewards; many MDPs used in practice satisfy this assumption.

**Assumption 2.2.** The concrete MDP $\mathcal{M}$ has deterministic transitions $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. Furthermore, there is a distinguished subgoal region $\tilde{s}_g \in \tilde{\mathcal{S}}$, called the *goal region*, such that (i) $\tilde{s}_g$ is a sink—i.e., for all $s \in \tilde{s}_g$ and $a \in \mathcal{A}$, $T(s, a) = s$, and (ii) the rewards are 1 if transitioning to $\tilde{s}_g$ and 0 otherwise—i.e., $R(s, a) = \mathbb{1}(s \notin \tilde{s}_g \wedge T(s, a) \in \tilde{s}_g)$.

# 3 ROBUST ABSTRACT VALUE ITERATION

First, we propose *robust abstract value iteration (R-AVI)*, which takes a set of subgoal transitions $\mathcal{O}$ and computes an option policy $\tilde{\rho}$. This algorithm is intended for theoretical analysis; it provides insights on what kinds of subgoal regions can achieve good performance, which can in turn guide the design of subgoal regions. In particular, we prove that $\tilde{\rho}$ is close to the optimal option policy $\rho^*$ when all states within each subgoal region are similar. We also provide a way to construct the subgoal transitions $\mathcal{O}$ and show that, for these options, the computed policy $\pi_{\tilde{\rho}}$ is close to the optimal policy $\pi^*$ for $\mathcal{M}$ if the subgoal regions are bottlenecks and $\mathcal{M}$ has sparse rewards.

**Algorithm.** Recall that we can in principle compute $\rho^*$ using (1); however, for continuous state spaces, we would need to use function approximation on $V^*_{\mathcal{O}}$ to do so. R-AVI leverages subgoal regions to avoid this issue—in particular, for each subgoal region $\tilde{s}$, it computes an interval $[V^*_{\text{inf}}(\tilde{s}), V^*_{\text{sup}}(\tilde{s})]$ such that for all $s \in \tilde{s}$, we have $V^*_{\mathcal{O}}(s) \in [V^*_{\text{inf}}(\tilde{s}), V^*_{\text{sup}}(\tilde{s})]$. It uses upper and lower bounds on the concrete transitions and rewards to do so. In particular, for $\tilde{s} \in \tilde{\mathcal{S}}$, let

$$\tilde{T}_{\text{inf}}(\tilde{s}, o, \tilde{s}') = \inf_{s \in \tilde{s}} \tilde{T}(s, o, \tilde{s}')$$

$$\tilde{T}_{\text{sup}}(\tilde{s}, o, \tilde{s}') = \sup_{s \in \tilde{s}} \tilde{T}(s, o, \tilde{s}'),$$

where, for $s \in S$, $o \in \mathcal{O}$, and $\tilde{s}' \in \tilde{\mathcal{S}}$,

$$\tilde{T}(s, o, \tilde{s}') = \sum_{t=1}^{\infty} \gamma^t P(\tilde{s}', t \mid s, o) \tag{2}$$

is the time-discounted probability of transitioning from concrete state $s$ to subgoal region $\tilde{s}'$ using option $o$, and where $P(\tilde{s}', t \mid s, o)$ is the probability that option $o$ terminates in subgoal region $\tilde{s}'$ after $t$ steps when starting from $s \in \tilde{s}$. The upper and lower bounds

[1] The induced policy $\pi_\rho$ depends on an additional internal state (i.e., the option being used) to make decisions.

[2] Given $\tilde{s}, \tilde{s}'$ such that $\tilde{s} \cap \tilde{s}' \neq \varnothing$, we can simply take $\tilde{s}' = \tilde{s}' \setminus \tilde{s}$.

on the rewards are similar—i.e., for $\tilde{s} \in \tilde{\mathcal{S}}$ and $o \in \mathcal{O}$,

$$\tilde{R}_{\text{inf}}(\tilde{s}, o) = \inf_{s \in \tilde{s}} R_{\text{opt}}(s, o)$$
$$\tilde{R}_{\text{sup}}(\tilde{s}, o) = \sup_{s \in \tilde{s}} R_{\text{opt}}(s, o).$$

While computing these bounds may be intractable in general, they can be approximated via sampling; since our focus in this section is on theoretical guarantees, we assume we have computed them exactly. Given these bounds, R-AVI computes $\tilde{V}_z^* : \tilde{\mathcal{S}} \to \mathbb{R}$ (for $z \in \{\text{inf}, \text{sup}\}$) by solving the recursive equations

$$\tilde{V}_z^*(\tilde{s}) = \max_{o \in \mathcal{O}} \tilde{Q}_z^*(\tilde{s}, o)$$
$$\tilde{Q}_z^*(\tilde{s}, o) = \tilde{R}_z(\tilde{s}, o) + \sum_{\tilde{s}' \in \tilde{\mathcal{S}}} \tilde{T}_z(\tilde{s}, o, \tilde{s}') \cdot \tilde{V}_z^*(\tilde{s}')$$

using value iteration. Since there are only finitely many subgoal regions and options, we can do so using tabular value iteration even though $\mathcal{M}$ has continuous state and action spaces. We define the *conservative optimal option policy* to be $\tilde{\rho}(s) = \arg\max_{o \in \mathcal{O}} \tilde{Q}_{\text{inf}}^*(\tilde{s}, o)$, for all $s \in \tilde{s}$ and $\tilde{s} \in \tilde{\mathcal{S}}$. Note that $\tilde{\rho}$ is only defined on $\bar{S}$ and also has a finite representation.

This algorithm can be interpreted as a version of value iteration on the *abstract decision process (ADP)* $\tilde{\mathcal{M}} = (\tilde{\mathcal{S}}, \mathcal{O}, \tilde{T}_{\text{inf}}, \tilde{T}_{\text{sup}}, \tilde{R}_{\text{inf}}, \tilde{R}_{\text{sup}}, \gamma, \tilde{s}_0)$, which is similar to an MDP except we are only given upper and lower bounds on the transitions and rewards rather than a single value. Intuitively, the gap in the difference between the upper and lower bounds captures the degree to which $\tilde{\mathcal{M}}$ fails to be Markov.

**Bound vs. $\rho^*$.** Next, we establish conditions under which we can bound the performance of $\tilde{\rho}$ compared to the optimal option policy $\rho^*$. Let $\varepsilon_T$ be the worst-case difference between $\tilde{T}_{\text{sup}}$[3] and $\tilde{T}_{\text{inf}}$, and $\varepsilon_R$ be the worst-case difference between $\tilde{R}_{\text{sup}}$ and $\tilde{R}_{\text{inf}}$:

$$\varepsilon_T = \max_{\tilde{s}, \tilde{s}' \in \tilde{\mathcal{S}},\ o \in \mathcal{O}} \tilde{T}_{\text{sup}}(\tilde{s}, o, \tilde{s}') - \tilde{T}_{\text{inf}}(\tilde{s}, o, \tilde{s}'),$$
$$\varepsilon_R = \max_{\tilde{s} \in \tilde{\mathcal{S}},\ o \in \mathcal{O}} \tilde{R}_{\text{sup}}(\tilde{s}, o) - \tilde{R}_{\text{inf}}(\tilde{s}, o).$$

Then, we assume that $\varepsilon_T$ is not too large.

**Assumption 3.1.** We have $|\tilde{\mathcal{S}}|\varepsilon_T < 1 - \gamma$.

As discussed above, $\varepsilon_T$ captures the degree to which $\tilde{\mathcal{M}}$ fails to be Markov. Then, abstract value iteration converges and $\tilde{\rho}$ has performance close to that of $\rho^*$.

**Theorem 3.2.** *Under Assumption 3.1, R-AVI converges and* $J(\pi_{\tilde{\rho}}) \geq J(\pi_{\rho^*}) - \frac{(1-\gamma)\varepsilon_R + |\tilde{\mathcal{S}}|\varepsilon_T}{(1-\gamma)(1-(\gamma+|\tilde{\mathcal{S}}|\varepsilon_T))}$.

[3]If $T$ is deterministic, then $\tilde{T}_{\text{sup}}(\tilde{s}, (\pi, \tilde{s}, \beta), \tilde{s}') = \gamma^N$, where $N$ is the minimum number of steps it takes for $\pi$ to reach $\tilde{s}'$ starting from some state $s \in \tilde{s}$ (or 0 if $\pi$ does not reach $\tilde{s}'$ from any $s \in \tilde{s}$).

**Constructing subgoal transitions.** So far, we have assumed that $\mathcal{O}$ is given. Given subgoal regions $\tilde{\mathcal{S}}$ and edges $E \subseteq \tilde{\mathcal{S}} \times \tilde{\mathcal{S}}$, R-AVI automatically constructs options that serve as subgoal transitions—namely, it constructs the following *ideal subgoal transitions*:

$$\tilde{\mathcal{O}} = \{(\pi(\tilde{s}, \tilde{s}'), \tilde{s}, \beta) \mid (\tilde{s}, \tilde{s}') \in E, \tilde{s} \neq \tilde{s}' \text{ and } \tilde{s} \neq \tilde{s}_g\},$$

where $\beta$ is as in Definition 2.1, and

$$\pi(\tilde{s}, \tilde{s}') = \arg\max_{\pi}\ \tilde{T}_{\text{sup}}(\tilde{s}, (\pi, \tilde{s}, \beta), \tilde{s}') \qquad (3)$$

maximizes the best-case reward for transitioning from $\tilde{s}$ to $\tilde{s}'$ over initial concrete states $s \in \tilde{s}$. We can approximately compute $\tilde{\mathcal{O}}$ using model-free RL; however, as before, our focus is on theoretical guarantees.

**Bound vs. $\pi^*$.** Theorem 3.2 is for a fixed set of options $\mathcal{O}$—i.e., both $\tilde{\rho}$ and $\rho^*$ use $\mathcal{O}$. In general, the choice of $\mathcal{O}$ determines how $\pi_{\rho^*}$ compares to the optimal policy $\pi^*$ for the underlying MDP $\mathcal{M}$. Suppose Assumption 2.2 holds; then, we can prove that $\tilde{\rho}$ constructed using the ideal subgoal transitions $\tilde{\mathcal{O}}$ performs nearly as well as $\pi^*$ under the *bottleneck assumption*.

**Assumption 3.3.** For any trajectory $s_0, a_0, s_1, \ldots, s_t$ such that $s_0 \in \tilde{s}_0$ and $s_t \in \tilde{s}_g$, there exists a sequence of indices $0 = i_0 < \ldots < i_k = t$ and a sequence of subgoal regions $\tilde{s}_0, \ldots, \tilde{s}_k$ such that (i) for all $0 \leq j \leq k$, $s_{i_j} \in \tilde{s}_j$, and (ii) for all $j < k$, $(\tilde{s}_j, \tilde{s}_{j+1}) \in E$.

Intuitively, this assumption says that the subgoal regions are bottlenecks—i.e., any path from an initial state to a goal state can be represented as a sequence of subgoal transitions. Then, the option policy $\tilde{\rho}$ R-AVI computes has performance close to that of the optimal policy $\pi^*$ for the concrete MDP $\mathcal{M}$.

**Theorem 3.4.** *Under Assumptions 2.2, 3.1, & 3.3, we have* $J(\pi_{\tilde{\rho}}) \geq J(\pi^*) - \frac{(1-\gamma)\varepsilon_R + |\tilde{\mathcal{S}}|\varepsilon_T}{(1-\gamma)(1-(\gamma+|\tilde{\mathcal{S}}|\varepsilon_T))}$.

This result is stronger than Theorem 3.2 since it compares to $\pi^*$, not $\rho^*$, but relies on stronger assumptions.

**Implications.** Our results establish two conditions on the subgoal regions $\tilde{\mathcal{S}}$ such that $\tilde{\rho}$ performs nearly as well as $\pi^*$: (i) Theorem 3.2 suggests that for any option $o$ and subgoal region $\tilde{s}$, the reward and time-discounted transition probabilities for $o$ are similar starting from any concrete state $s \in \tilde{s}$, and (ii) Theorem 3.4 suggests that the subgoal regions should be bottlenecks in the underlying MDP.

# 4 ALTERNATING ABSTRACT VALUE ITERATION

There are two shortcomings of R-AVI. First, computing $\tilde{\mathcal{O}}$, $\tilde{T}_{\text{inf}}$, and $\tilde{R}_{\text{inf}}$ may be computationally infeasible since we only assume the ability to obtain samples

**Algorithm 1** (A-AVI) Iterative algorithm for constructing hierarchical policy.

**function** LearnPolicy($\mathcal{M}, \tilde{\mathcal{S}}, E, N$)
  Initialize $\mathcal{D}$
  **for** $i \in \{1, ..., N\}$ **do**
    Learn the ideal subgoal transitions $\mathcal{O}_\mathcal{D}$
    Estimate $\tilde{T}_\mathcal{D}$ and $\tilde{R}_\mathcal{D}$ for $\mathcal{O}_\mathcal{D}$
    Compute $\tilde{\rho}_\mathcal{D}$ using abstract value iteration
    **for** $\tilde{s} \in \tilde{\mathcal{S}}$ **do**
      $\bar{\mathcal{D}} \leftarrow$ Distribution over $\tilde{s}$ induced by $\pi_{\tilde{\rho}_\mathcal{D}}$
      Update $\mathcal{D} \leftarrow (1 - \alpha_i)\mathcal{D} + \alpha_i \bar{\mathcal{D}}$
    **end for**
  **end for**
  **return** $\mathcal{O}_\mathcal{D}$, $\pi_{\rho_\mathcal{D}}$
**end function**

from $\mathcal{M}$. Second, making conservative assumptions about $\tilde{T}$ and $\tilde{R}$ can lead to suboptimal $\tilde{\rho}$.

We propose *alternating abstract value iteration (A-AVI)* (shown in Algorithm 1). This algorithm addresses the issues with R-AVI by planning according to the expected values of $\tilde{T}$ and $\tilde{R}$ with respect to some distribution $\mathcal{D}$ over concrete states $\mathcal{S}$. Naïvely, this approach only works when the ADP $\tilde{\mathcal{M}}$ is Markov; otherwise, the estimates of $\tilde{T}$ and $\tilde{R}$ depend on the choice of $\mathcal{D}$. To address this issue, A-AVI alternates between (i) given $\mathcal{D}$, learn the subgoal transitions $\mathcal{O}_\mathcal{D}$ using model-free RL and estimate the expected values of $\tilde{T}$ and $\tilde{R}$ for $\mathcal{O}_\mathcal{D}$, and (ii) given $\mathcal{O}_\mathcal{D}$ and the estimates of $\tilde{T}$ and $\tilde{R}$, compute the optimal option policy $\rho_\mathcal{D}$ using value iteration, and update $\mathcal{D}$ to be the state distribution induced by using $\rho_\mathcal{D}$.

**Step 1.** For step (i), we first define the ideal subgoal transitions with respect to $\mathcal{D}$ to be

$$\mathcal{O}_\mathcal{D} = \{(\pi_\mathcal{D}(\tilde{s}, \tilde{s}'), \tilde{s}, \beta) \mid (\tilde{s}, \tilde{s}') \in E,\ \tilde{s} \neq \tilde{s}',\ \tilde{s} \neq \tilde{s}_g\}$$
$$\pi_\mathcal{D}(\tilde{s}, \tilde{s}') = \arg\max_\pi \mathbb{E}_{s \sim \mathcal{D}}[\tilde{T}(s, (\pi, \tilde{s}, \beta), \tilde{s}') \mid s \in \tilde{s}],$$

where $\tilde{T}$, defined in (2), is the time-discounted probability density of transitioning to $\tilde{s}'$ from a concrete state $s \in \tilde{s}$. Intuitively, $\pi_\mathcal{D}(\tilde{s}, \tilde{s}')$ is the policy that maximizes probability of transitioning to $\tilde{s}'$ from $\tilde{s}$. Then, computing $\pi_\mathcal{D}(\tilde{s}, \tilde{s}')$ can be formulated as the RL problem for the MDP $\mathcal{M}' = (\mathcal{S}, \mathcal{A}, T', R', \gamma, \eta_0')$, where

$$T'(s, a, s') = \begin{cases} T(s, a, s') & \text{if } s \notin \bar{S} \setminus \tilde{s} \\ \frac{\mathbb{1}(s' \in \tilde{s}'')}{\int_\mathcal{S} \mathbb{1}(s' \in \tilde{s}'') ds'} & \text{if } s \in \tilde{s}'' \subseteq \bar{S} \setminus \tilde{s} \end{cases}$$
$$R'(s, a) = \mathbb{1}(s \notin \tilde{s}') \cdot \int_{\tilde{s}'} T(s, a, s') ds',$$

and $\eta_0' = \mathcal{D}_{\tilde{s}}$, where $\mathcal{D}_{\tilde{s}} = \mathcal{D} \mid s \in \tilde{s}$ is the conditional distribution of $\mathcal{D}$ given that $s \in \tilde{s}$. That is, $\mathcal{M}'$ is the concrete MDP $\mathcal{M}$ except where all subgoal regions in $\tilde{\mathcal{S}} \setminus \{\tilde{s}\}$ are sinks, the rewards $R'$ encode that $\tilde{s}'$ is the goal region, and the initial state distribution is $\mathcal{D}_{\tilde{s}}$. Since we assumed we can simulate $\mathcal{M}$ starting at any state $s \in \mathcal{S}$, we can also simulate $\mathcal{M}'$ by terminating episodes upon reaching $\bar{S} \setminus \tilde{s}$. Thus, existing RL algorithms can be used to learn these policies.

**Step 2.** Next, for step (ii), A-AVI plans in the ADP $\tilde{\mathcal{M}}_\mathcal{D}$, whose transitions $\tilde{T}_\mathcal{D}$ and rewards $\tilde{R}_\mathcal{D}$ are the expected values of the time-discounted transition probabilities $\tilde{T}$ and rewards $\tilde{R}$, respectively—i.e.,

$$\tilde{T}_\mathcal{D}(\tilde{s}, o, \tilde{s}') = \mathbb{E}_{s \sim \mathcal{D}_{\tilde{s}}}[\tilde{T}(s, o, \tilde{s}')]$$
$$\tilde{R}_\mathcal{D}(\tilde{s}, o) = \mathbb{E}_{s \sim \mathcal{D}_{\tilde{s}}}[\tilde{R}(s, o)],$$

which can be estimated using sampled rollouts. Then, A-AVI uses value iteration to solve

$$\tilde{V}^*_\mathcal{D}(\tilde{s}) = \max_{o \in \mathcal{O}_\mathcal{D}} \tilde{Q}^*_\mathcal{D}(\tilde{s}, o)$$
$$\tilde{Q}^*_\mathcal{D}(\tilde{s}, o) = \tilde{R}_\mathcal{D}(\tilde{s}, o) + \sum_{\tilde{s}' \in \tilde{\mathcal{S}}} \tilde{T}_\mathcal{D}(\tilde{s}, o, \tilde{s}') \cdot \tilde{V}^*_\mathcal{D}(\tilde{s}'),$$

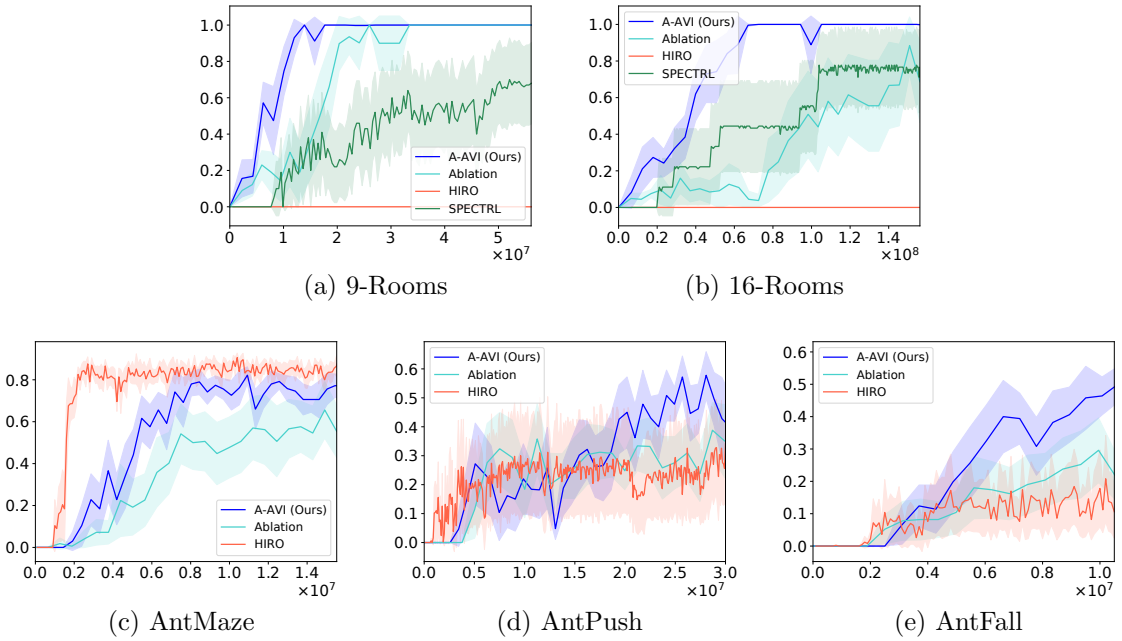
and computes the policy $\tilde{\rho}_\mathcal{D} : \bar{S} \to \mathcal{O}_\mathcal{D}$ as $\tilde{\rho}_\mathcal{D}(s) = \arg\max_{o \in \mathcal{O}_\mathcal{D}} \tilde{Q}^*_\mathcal{D}(\tilde{s}, o)$ for all $s \in \tilde{s}$ and $\tilde{s} \in \tilde{\mathcal{S}}$. Next, we estimate $\mathcal{D}$ to equal the state distribution over $\mathcal{S}$ induced by using policy $\pi_{\tilde{\rho}_\mathcal{D}}$ in the concrete MDP $\mathcal{M}$. Intuitively, this condition says that $\pi_{\tilde{\rho}_\mathcal{D}}$ is used on the same state distribution as it was trained. More precisely, we take $\mathcal{D} = (1 - \alpha_i)\mathcal{D} + \alpha_i \bar{\mathcal{D}}$, where $\bar{\mathcal{D}}$ is the state distribution over $\mathcal{S}$ induced by using $\pi_{\tilde{\rho}_\mathcal{D}}$. This approach, based on dataset aggregation, is a heuristic to facilitate convergence of A-AVI (Ross et al., 2011).

**R-AVI vs A-AVI.** We emphasize that with R-AVI, we can theoretically characterize the quality of a given set of subgoal regions, which enables us to guide their design. The upper and lower bounds in R-AVI are necessary for the theoretical guarantees, but computing them for continuous state and action spaces is intractable. Consequently, A-AVI takes a different approach for dealing with the non-Markov nature of the ADP; in particular, it uses the expected MDP but then uses alternation to improve robustness. This makes A-AVI a practical hierarchical RL algorithm that we evaluate empirically in our experiments.

## 5 EXPERIMENTS

We evaluate our approach[4] on two continuous control benchmarks: (i) two room environments where a robot must navigate a maze of rooms, which are continuous variants of standard hierarchical RL benchmarks (Gopalan et al., 2017; Abel et al., 2020) and (ii)

[4]Our implementation is available at https://github.com/keyshor/abstract-value-iteration.




(a) 9-Rooms (b) 16-Rooms

(c) AntMaze (d) AntPush (e) AntFall

Figure 2: Comparison with baselines for different environments; $x$-axis is number of samples (steps) from the environment, and $y$-axis is the probability of reaching the goal. Results are averaged over 10 executions.

a hierarchical RL benchmark (Nachum et al., 2018) based on the MuJoCo ant (Todorov et al., 2012). The room environments have comparatively simple dynamics, but the high-level task is very challenging due to the nonconvex state space. Alternatively, the ant environments have more complex robot dynamics, but comparatively simple high-level tasks. Our approach, A-AVI, outperforms state-of-the-art baselines in both cases.

**Room environments.** We consider two environments, 9-Rooms and 16-Rooms, consisting of interconnected rooms. They have states $(x, y) \in \mathbb{R}^2$ encoding 2D position, actions $(v, \theta) \in \mathbb{R}^2$ encoding speed and direction, and transitions $s' = s + (v\cos(\theta), v\sin(\theta))$. Figure 1 (a) shows 9-Rooms. Subgoal regions are the light gray squares; edges connect adjacent subgoal regions. The agent starts in a uniformly random state in the initial subgoal region (the blue square); its goal is to reach the goal region (the green square). The agent receives a reward of 1 upon reaching the goal. We learn subgoal transitions using ARS (Mania et al., 2018), with a shaped reward equal to the negative distance to the center of the target subgoal region; each policy is a fully connected neural network with 2 hidden layers with 30 neurons each. We give details in Appendix B.

**Ant environments.** We also consider three MuJoCo (Todorov et al., 2012) ant environments from Nachum et al. (2018): AntMaze (navigate a U-shaped corridor), AntPush (push away a large block to reach the region behind it), and AntFall (push a large block into a chasm to form a bridge to get to the other side). We consider subgoal regions that are subsets of the state space where the ant position is in a small rectangular region on the plane. AntFall has four subgoal regions: the initial region, an intermediate region at each of the two turns, and the goal region. AntPush has five subgoal regions: the initial region, two at the bottom-left and top-left corners, one at the gap where the ant must enter to reach the goal, and the goal region; in the abstract MDP, there are three paths from the initial region to the goal region. AntFall has five subgoal regions: the initial region, three along the path to the goal region, and the goal region. We learn subgoal transitions using TD3 (Fujimoto et al., 2018). To improve sample efficiency, we retain the state of TD3 (i.e., actor-networks, critic-networks, replay buffer, etc.) across iterations of A-AVI. The neural network architecture and hyperparameters are similar to Nachum et al. (2018). We give details, including visualizations of the subgoal regions, in Appendix B.

**Results.** We show results in Figure 2. Our approach tends to perform better than our baselines in tasks requiring significant exploration—i.e., it substantially outperforms all baselines for 9-Rooms, 16-Rooms, and AntFall. We discuss comparisons below.

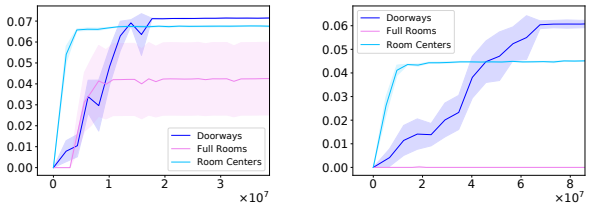
**Comparison to HIRO.** We compare to a state-of-the-art hierarchical deep RL algorithm called

HIRO (Nachum et al., 2018), which uses a high-level policy to generate intermediate goals that a low-level policy aims to achieve. As with our algorithm, we used shaped rewards—i.e., the negative distance from the current state to the center of the goal region, which is fixed for each environment. HIRO does not require any additional information about the environment; in particular, it is not given the subgoal regions as input. As seen in Figure 2, HIRO performs substantially worse on 9-Rooms and 16-Rooms. HIRO does not know the structure of the state space, so it is unable to discover the path from the initial region to the goal region and gets stuck in a local optimum. Intuitively, HIRO is designed to decouple complex dynamics (e.g., the ant) from high-level planning (e.g., sequence of tasks), not to solve challenging high-level planning problems.

**Comparison to SpectRL.** We compare to SpectRL (Jothimurugan et al., 2019), a framework where the user specifies a task as a sequence of subgoals (rather than as a reward function); then, SpectRL uses the subgoals to generate a reward function. It is essentially a hierarchical RL framework that uses options but not state abstractions. SpectRL takes as input not only the subgoals, but also potential sequences of subgoals that can be used to reach the goal, thereby requiring additional user-provided information beyond what is needed by our approach. For each room environment, we specify the task as a choice between two sequences of subgoals representing two paths from the initial region to the goal region; each subgoal is centered at a subgoal region along the path. As can be seen in Figure 2, SpectRL learns policies with suboptimal success rates since it sometimes chooses the wrong path. This shortcoming is because SpectRL does not solve for the optimal policy at the abstract level; instead, it uses a greedy heuristic.

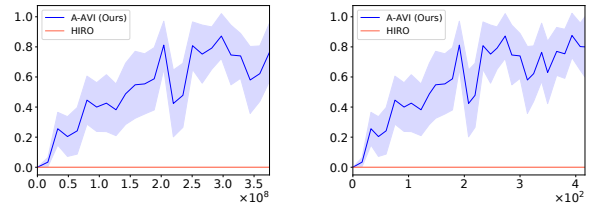
**No alternation.** Our A-AVI algorithm uses alternating optimization to handle the non-Markov nature of the abstract MDP. To evaluate the effectiveness of this approach, we consider an ablation of our algorithm where $\mathcal{D}$ is not updated (equivalently, one iteration of A-AVI). This ablation can be thought of as extending Gopalan et al. (2017) to learn the transitions and rewards of the abstract MDP, or extending Winder et al. (2020) to handle continuous state spaces. As can be seen in Figure 2, our ablation performs poorly, likely because it is unable to account for the non-Markov nature of the ADP. Empirically, we observe that the robot often becomes stuck at walls at the boundary of subgoal regions; these states are very rarely sampled under the distribution $\mathcal{D}$.

**Choice of subgoal regions.** We study the impact of the choice of subgoal regions on performance for the room environments. Our choice of "doorways" used in the above experiments is motivated by our theory for R-AVI, which suggests that subgoal regions should be bottlenecks that are small in size. We evaluate two alternatives: (i) "room center" consists of a square at the center of each room that is the same size and shape as the doorways, and (ii) "full room" consists of a square for each room covering the entire room (similar to Abel et al. (2020)) as shown in Figure 1 (e). Results are shown in Figure 3. Our original choice "doorways" achieves the highest discounted reward in all environments, validating our theory. "Full rooms" performs poorly, likely because the large regions induce large defects in the ADP. "Room centers" converges quickly but to a suboptimal value, likely because the robot cannot travel diagonally across rooms (e.g., the $1 \to 4$ transition in Figure 1 (a)). Thus, using smaller subgoal regions is crucial for good performance, whereas using bottlenecks is of secondary importance.




(a) 9-Rooms (b) 16-Rooms

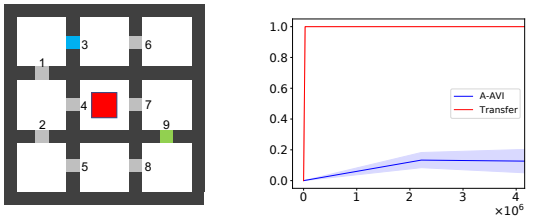
Figure 3: Comparison of subgoal regions for room environments; $x$-axis is number of samples (steps) from the environment, and $y$-axis is the discounted reward. Results are averaged over 10 executions.




(a) Sample Complexity (b) Learning Time

Figure 4: Learning curves of A-AVI with randomly generated subgoal regions in 9-Rooms; the plots show the probability of reaching the goal ($y$-axis) as a function of (a) number of samples (steps) from the environment and (b) time since the beginning of training (in minutes). Results are averaged over 10 executions.

**Random subgoal regions.** We also consider randomly sampling subgoal regions for the room environments. We first sample $N$ points uniformly at random from the 2D plane. For each point, we define a subgoal region which is a small square with that point in its center. Next, we add edges from each point to its




(a) 9-Rooms-Obstacle (b) Sample Complexity

Figure 5: Planning using R-AVI in 9-Rooms-Obstacle using options learned in 9-Rooms; $x$-axis is the number of samples (steps) from the environment, and $y$-axis is the probability of reaching the goal.

$K$ nearest neighbors. For the 9-Rooms environment, we set $N = 20$ and $K = 7$. As shown in Figure 4, this approach performs significantly better than HIRO without any additional input from the user. Although the sample complexity is high, the learning time is low since the options are learned in parallel. The learning curves for the 16-Rooms environment are in Appendix B.

**Transferring learned options.** An advantage of our framework is that the subgoal transitions we learn can be reused across different tasks. One such task we consider is 9-Rooms-Obstacle shown in Figure 5 (a). Here, we have added an obstacle to the middle room. The high-level plan for 9-Rooms is no longer feasible since the option corresponding to the edge $4 \to 7$ causes the robot to collide with the obstacle. Nonetheless, we can reuse the existing options and compute a different high-level plan. In this case, we use R-AVI to avoid re-learning the options (which is significantly more expensive in terms of sample complexity); it computes the plan $3 \to 1 \to 2 \to 5 \to 8 \to 9$. As shown in Figure 5 (b), the new plan achieves a high probability of reaching the goal given just a small number of samples, since we only need a few samples to estimate $\tilde{T}_{\text{inf}}$ and $\tilde{R}_{\text{inf}}$. Similarly, we were also able to compute plans for different start and goal regions.

## 6 CONCLUSIONS

We have proposed a hierarchical RL algorithm that constructs an abstract high-level model using user-provided subgoal regions and plans in this high-level model using abstract value iteration. Future work includes incorporating algorithms to automatically discover subgoal regions which would mitigate the need for additional information from the user as well as applying the approach to more complex benchmarks.

### Acknowledgements

We thank the anonymous reviewers for their insightful comments. This research was partially supported by ONR award N00014-20-1-2115, as well as NSF award CCF 1910769.

### References

Abel, D., Arumugam, D., Asadi, K., Jinnai, Y., Littman, M. L., and Wong, L. L. (2019). State abstraction as compression in apprenticeship learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3134–3142.

Abel, D., Umbanhowar, N., Khetarpal, K., Arumugam, D., Precup, D., and Littman, M. L. (2020). Value preserving state-action abstractions. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*.

Andre, D. and Russell, S. J. (2002). State abstraction for programmable reinforcement learning agents. In *AAAI/IAAI*, pages 119–125.

Andrychowicz, O. M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., et al. (2020). Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20.

Bacon, P.-L., Harb, J., and Precup, D. (2017). The option-critic architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Burridge, R. R., Rizzi, A. A., and Koditschek, D. E. (1999). Sequential composition of dynamically dexterous robot behaviors. *The International Journal of Robotics Research*, 18(6):534–555.

Castro, P. S. (2019). Scalable methods for computing state similarity in deterministic markov decision processes. *arXiv preprint arXiv:1911.09291*.

Castro, P. S. and Precup, D. (2011). Automatic construction of temporally extended actions for mdps using bisimulation metrics. In *European Workshop on Reinforcement Learning*, pages 140–152. Springer.

Choudhury, S., Knickerbocker, J. P., and Kochenderfer, M. J. (2019). Dynamic real-time multimodal routing with hierarchical hybrid planning. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2397–2404. IEEE.

Co-Reyes, J., Liu, Y., Gupta, A., Eysenbach, B., Abbeel, P., and Levine, S. (2018). Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings. In *International Conference on Machine Learning*, pages 1009–1018. PMLR.

Cobo, L. C., Zang, P., Isbell Jr, C. L., and Thomaz, A. L. (2011). Automatic state abstraction from demonstration. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

Daniel, C., Van Hoof, H., Peters, J., and Neumann, G. (2016). Probabilistic inference for determining options in reinforcement learning. *Machine Learning*, 104(2-3):337–357.

Dietterich, T. G. (2000). State abstraction in maxq hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 994–1000.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. (2018). Diversity is all you need: Learning skills without a reward function. In *ICLR*.

Ferns, N., Panangaden, P., and Precup, D. (2004). Metrics for finite markov decision processes. In *UAI*, volume 4, pages 162–169.

Finn, C., Yu, T., Fu, J., Abbeel, P., and Levine, S. (2017). Generalizing skills with semi-supervised reinforcement learning. In *ICLR*.

Fujimoto, S., Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596.

Givan, R., Leach, S., and Dean, T. (2000). Bounded-parameter markov decision processes. *Artificial Intelligence*, 122(1-2):71–109.

Gopalan, N., Littman, M. L., MacGlashan, J., Squire, S., Tellex, S., Winder, J., Wong, L. L., et al. (2017). Planning with abstract markov decision processes. In *Twenty-Seventh International Conference on Automated Planning and Scheduling*.

Guadarrama, S., Korattikara, A., Ramirez, O., Castro, P., Holly, E., Fishman, S., Wang, K., Gonina, E., Wu, N., Kokiopoulou, E., Sbaiz, L., Smith, J., Bartók, G., Berent, J., Harris, C., Vanhoucke, V., and Brevdo, E. (2018). TF-Agents: A library for reinforcement learning in tensorflow. `https://github.com/tensorflow/agents`. [Online; accessed 25-June-2019].

Hausman, K., Springenberg, J. T., Wang, Z., Heess, N., and Riedmiller, M. (2018). Learning an embedding space for transferable robot skills. In *ICLR*.

Jong, N. K. and Stone, P. (2005). State abstraction discovery from irrelevant state variables. In *IJCAI*, volume 8, pages 752–757.

Jonsson, A. and Barto, A. G. (2001). Automated state abstraction for options using the u-tree algorithm. In *Advances in neural information processing systems*, pages 1054–1060.

Jothimurugan, K., Alur, R., and Bastani, O. (2019). A composable specification language for reinforcement learning tasks. In *Advances in Neural Information Processing Systems*, pages 13021–13030.

Khan, A., Tolstaya, E., Ribeiro, A., and Kumar, V. (2019). Graph policy gradients for large scale robot control. In *CoRL*.

Konidaris, G. and Barto, A. G. (2007). Building portable options: Skill transfer in reinforcement learning. In *IJCAI*, volume 7, pages 895–900.

Konidaris, G. and Barto, A. G. (2009). Skill discovery in continuous reinforcement learning domains using skill chaining. In *Advances in neural information processing systems*, pages 1015–1023.

Konidaris, G., Kaelbling, L., and Lozano-Perez, T. (2014). Constructing symbolic representations for high-level planning. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Kulkarni, T. D., Narasimhan, K., Saeedi, A., and Tenenbaum, J. (2016). Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, pages 3675–3683.

LaValle, S. M. (2006). *Planning algorithms*. Cambridge university press.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373.

Li, L., Walsh, T. J., and Littman, M. L. (2006). Towards a unified theory of state abstraction for mdps. In *ISAIM*.

Machado, M. C., Bellemare, M. G., and Bowling, M. (2017). A laplacian framework for option discovery in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2295–2304. JMLR. org.

Majumdar, A. and Tedrake, R. (2017). Funnel libraries for real-time robust feedback motion planning. *The International Journal of Robotics Research*, 36(8):947–982.

Mania, H., Guy, A., and Recht, B. (2018). Simple random search of static linear policies is competitive for reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1800–1809.

Nachum, O., Gu, S., Lee, H., and Levine, S. (2019). Near-optimal representation learning for hierarchical reinforcement learning. In *ICLR*.

Nachum, O., Gu, S. S., Lee, H., and Levine, S. (2018). Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 3303–3313.

Precup, D., Sutton, R. S., and Singh, S. (1998). Theoretical results on reinforcement learning with temporally abstract options. In *European conference on machine learning*, pages 382–393. Springer.

Roderick, M., Grimm, C., and Tellex, S. (2018). Deep abstract q-networks. In *AAMAS*.

Ross, S., Gordon, G., and Bagnell, D. (2011). A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635.

Stolle, M. and Precup, D. (2002). Learning options in reinforcement learning. In *International Symposium on abstraction, reformulation, and approximation*, pages 212–223. Springer.

Sun, S.-H., Wu, T.-L., and Lim, J. J. (2019). Program guided agent. In *International Conference on Learning Representations*.

Sutton, R. S., Precup, D., and Singh, S. (1999). Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211.

Taïga, A. A., Courville, A., and Bellemare, M. G. (2018). Approximate exploration through state abstraction. *arXiv preprint arXiv:1808.09819*.

Taylor, J., Precup, D., and Panagaden, P. (2009). Bounding performance loss in approximate mdp homomorphisms. In *Advances in Neural Information Processing Systems*, pages 1649–1656.

Theocharous, G. and Kaelbling, L. P. (2004). Approximate planning in pomdps with macro-actions. In *Advances in Neural Information Processing Systems*, pages 775–782.

Theocharous, G., Mahadevan, S., and Kaelbling, L. P. (2005). Spatial and temporal abstractions in pomdps applied to robot navigation. Technical report, MIT.

Tiwari, S. and Thomas, P. S. (2019). Natural option critic. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5175–5182.

Todorov, E., Erez, T., and Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE.

Walsh, T. J., Li, L., and Littman, M. L. (2006). Transferring state abstractions between mdps. In *ICML Workshop on Structural Knowledge Transfer for Machine Learning*.

Winder, J., Milani, S., Landen, M., Oh, E., Parr, S., Squire, S., desJardins, M., and Matuszek, C. (2020). Planning with abstract learned models while learning transferable subtasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34.