Abstract Value Iteration for Hierarchical Deep 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: given user-provided *subgoal regions* which are subsets of states, it (i) constructs options that serve as transitions between subgoal regions, and (ii) constructs a high-level plan in the resulting abstract MDP. The key challenge is that the abstract MDP may not be Markov. We propose two algorithms for planning in the abstract MDP that address this issue. Our first algorithm is conservative, allowing us to prove theoretical guarantees on its performance; these results can help guide the design of subgoal regions. Our second algorithm is a practical variant that interweaves planning at the abstract level using value iteration and at the concrete level using model-free reinforcement learning. We demonstrate the benefits of our approach on several benchmarks that are challenging for state-of-the-art hierarchical reinforcement learning algorithms.

1 Introduction

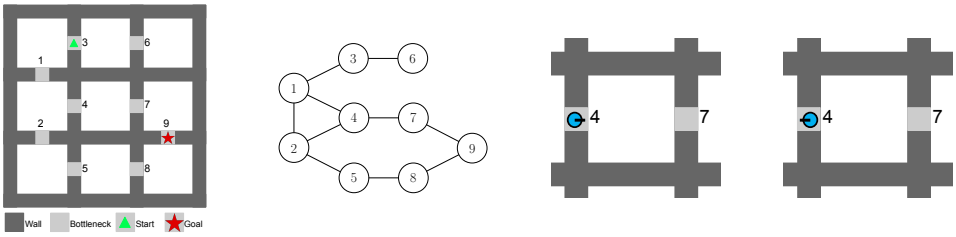
Deep reinforcement learning (RL) has been used to solve challenging robotics problems, including multi-agent control [23], object manipulation [4], and control from perception [28]. However, it remains primarily applied to tasks with short planning horizon; longer horizon tasks currently require huge amounts of computational power to solve [4].

Hierarchical RL is a promising approach to scaling deep 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 concrete 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.

We consider two kinds of abstractions. First, *action abstractions* (also called *options*, *temporal abstractions*, or *skills*) are low-level policies designed to achieve a short-term goal such as walking to a goal or grasping an object [35, 38, 41]. Second, *state abstractions* aggregate similar concrete states into a single abstract state [12, 3, 42, 29, 17, 46, 2]. Whereas action abstractions enable planning over long time horizons, state abstractions facilitate planning in large state spaces. They can be combined by constructing an *abstract MDP*, whose states are abstract states, whose actions are abstract actions, and where abstract actions serve as transitions between abstract states [46].

Existing approaches have two key shortcomings. First, they assume that the abstract MDP satisfies the Markov property. However, abstract actions do not necessarily work equally well for all states in an abstract state, and hence, future transitions may depend on which (concrete) state the agent reached during the previous transition violating the Markov assumption. Second, they typically require that users provide both the state and the action abstractions; this task is particularly challenging since the user must ensure that the resulting abstract MDP satisfies the Markov property.

Presented at Deep RL Workshop, NeurIPS 2020.



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

Figure 1: (a) A rooms environment; subgoal regions are in light gray, the initial region is marked by a green triangle, and the goal region is marked by a red star. (b) The corresponding abstract MDP; transitions are bi-directional. (c,d) Robot in region 4 facing right vs. left.

We propose a novel hierarchical reinforcement learning framework that addresses these challenges. For the second challenge, our framework only requires the user to provide state abstractions in the form of *subgoal regions* which are subsets of states; then, it uses reinforcement learning to learn options that transition between pairs of subgoal regions. For the first challenge, our framework models the abstract MDP as an *uncertain MDP* where the rewards and transitions are subject to unknown perturbations. Then, it uses an *abstract value iteration (AVI)* algorithm that is robust to these perturbations to plan in this MDP.

We propose two variants of AVI. First, *robust abstract value iteration (R-AVI)* models the unknown perturbations adversarially. It is of theoretical interest—we establish bounds on its performance, and show how these bounds can be used to guide the design of subgoal regions. Second, *alternating abstract value iteration (A-AVI)* is a practical variant that plans using only the expected value of the perturbations; to improve robustness, it alternates between estimating the expected perturbations and planning in the abstract MDP. Furthermore, it uses dataset aggregation to facilitate convergence [36].

We prove two bounds on the performance of R-AVI. First, we prove that it performs nearly as well as the optimal option policy (i.e., the best sequence of options), as long as concrete states in a subgoal region are all similar in terms of transition and reward behaviors of the options. Second, specializing to deterministic MDPs with sparse rewards, we prove that it performs nearly as well as the optimal concrete policy (i.e., best sequence of concrete actions), as long as subgoal regions are "bottlenecks" (i.e., they must be traversed to reach the goal) [19]. These two conditions can help guide the design of subgoal regions that perform well in the context of our framework.

We demonstrate that our approach outperforms several state-of-the-art baselines, including approaches that solve the abstract MDP without accounting for uncertainty [46], HIRO (which requires no user input) [34], and SpectRL (which requires more complex user input) [22], on both a self-driving robot navigating a maze of rooms as well as the MuJoCo Ant [44] performing sequences of tasks [34]. In addition, we demonstrate how our theory can guide the choice of good state abstractions.

Illustrative example. Consider the example in Figure 1 (a). The goal is for the robot to drive from an initial state (the green triangle) to a goal state (the red star). The reward is 1 upon reaching the goal and 0 otherwise. The user provides the subgoal regions; in this example, they are the doorways connecting 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 and reward behaviors are similar across concrete states in a subgoal region. Our framework learns low-level policies to transition between adjacent subgoal regions; the resulting abstract MDP is shown in Figure 1 (b). The high-level policy constructed using abstract value iteration is $3 \to 1 \to 4 \to 7 \to 9$. Finally, (c,d) demonstrate why the abstract MDP might not satisfy the Markov assumption. Suppose the robot cannot move backwards. If it previously took the transition $1 \to 4$, then it arrives in 4 in concrete state (c); in this case, it can easily take the transition $4 \to 7$ next. However, if it previously took $7 \to 4$, then it arrives in 4 in concrete state (d); then, it is much more costly for it to take $4 \to 7$ next. Our experiments show that existing approaches to solving the abstract MDP [17, 46] perform poorly since they do not account for this issue, and that HIRO [34] and SpectRL [22] perform poorly since they do not systematically explore in the abstract MDP.

Related work. There has been work on planning with options [35, 38, 41], including in the setting of deep RL [5, 43]. There has been work on leveraging action abstractions in the setting of deep

RL [27, 34, 33]. More closely related, there has been work on leveraging state abstractions, typically in conjunction with action abstractions [12, 3, 42, 29, 17, 9, 46, 2]. For instance, [17] proposes an algorithm for planning in hierarchical MDPs, but assumes 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 [46]; 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 [2], 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.

There has been work on inferring options, by transferring options to new domains [24], inferring options in multi-task reinforcement learning [37, 25, 30, 15, 13], from demonstrations [18], or using expectation-maximization [11]. Similarly, there has been interest in planning by composing low-level skills [6, 31]. 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 [14, 20, 1], by transferring them to new domains [45], from demonstrations [10], and from options [21, 26]. There has been work on inferring state abstractions [14, 40, 39, 7] and options [8] by measuring state similarity in terms of reward and transition properties, but only for finite MDPs.

2 Problem Formulation

Background. Consider a Markov decision process (MDP) $(\mathcal{S}, \mathcal{A}, P, r, \eta_0, \gamma)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions,[1] $P(s' \mid s, a) \in \mathbb{R}$ is the probability of transitioning from s to s' on action a, $r(s, a) \in [0, 1]$ is the reward for action a is state s, η_0 is the initial state distribution, and $\gamma \in [0, 1)$ is the discount factor. We consider policies $\pi : \mathcal{S} \to \mathcal{A}$, where $a = \pi(s)$ is the action to take in state s. The value function is $V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s\right]$, and the optimal policy is $\pi^* = \arg\max_\pi J(\pi)$, where $J(\pi) = \mathbb{E}_{s_0 \sim \eta_0}[V^\pi(s_0)]$; we let $V^* = V^{\pi^*}$.

An *option* o is a tuple (π, I, β), where π is a policy, $I \subseteq \mathcal{S}$ is a set of initial states from which π can be used, and $\beta : \mathcal{S} \to [0, 1]$ is the termination probability [38]. Given an underlying MDP $(\mathcal{S}, \mathcal{A}, P, r, \eta_0, \gamma)$ and options $\mathcal{O}$, a *multi-time model* is a tuple $(\mathcal{S}, \mathcal{O}, P_{\text{opt}}, r_{\text{opt}})$ [38], where for $s, s' \in \mathcal{S}$ and $o = (\pi, I, \beta) \in \mathcal{O}$,

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

$$r_{\text{opt}}(s, o) = \mathbb{E}_{s_0, a_0, \ldots, s_t \sim o}\left[r(s_0, a_0) + \cdots + \gamma^{t-1} r(s_{t-1}, a_{t-1}) \mid s_0 = s\right]$$

are the time-discounted transition probabilities,[2] where $p(s', t \mid s, o)$ is the probability that o terminates in s' after t steps when starting from s, and the expected reward before termination using o from s, where t is the random time at which o terminates when started at s, respectively. 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; ρ induces a policy π_ρ for the underlying MDP. The optimal option policy is $\rho^*(s) = \arg\max_{o \in \mathcal{O}} Q^*_{\mathcal{O}}(s, o)$, where we use *option value iteration* [38] to compute

$$Q^*_{\mathcal{O}}(s, o) = r_{\text{opt}}(s, o) + \sum_{s' \in \mathcal{S}} P_{\text{opt}}(s' \mid s, o) V^*_{\mathcal{O}}(s') \qquad \text{where} \qquad V^*_{\mathcal{O}}(s) = \max_{o \in \mathcal{O}} Q^*_{\mathcal{O}}(s, o).$$

Problem formulation. We assume given a set of *subgoal regions* $\tilde{\mathcal{S}}$, where each $\tilde{s} \in \tilde{\mathcal{S}}$ is a set of concrete states $\tilde{s} \subseteq S$, and the subgoal regions are disjoint from each other (i.e., $\tilde{s} \cap \tilde{s}' = \varnothing$ if $\tilde{s} \neq \tilde{s}'$). Unlike abstract states, the subgoal regions do not need to cover the state space of the underlying MDP. Then, action abstractions are given by options that start in some subgoal region and terminate in any other subgoal region.

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

[1] We allow the state and action spaces to be continuous (i.e., $\subseteq \mathbb{R}^n$).

[2] While P_{opt} is not a probability distribution, option value iteration converges to the optimal policy [38].

We let $\mathcal{M}$ be the underlying MDP and $\mathcal{O}$ be a set of subgoal transitions. For simplicity, we assume all initial states are in a single subgoal region—i.e., there exists an *initial region* $\tilde{s}_0 \in \tilde{\mathcal{S}}$ such that $\text{supp}(\eta_0) \in \tilde{s}_0$, where $\text{supp}(\eta_0)$ is the support of η_0. Our goal is to compute an *abstract option policy* $\tilde{\rho} : \mathcal{S} \to \mathcal{O}$, which is an option policy such that $\tilde{\rho}(s)$ is the same for all $s \in \tilde{s}$—i.e., for all $\tilde{s} \in \tilde{\mathcal{S}}$ and all $s, s' \in \tilde{s}$, we have $\tilde{\rho}(s) = \tilde{\rho}(s')$. Initially, we assume $\mathcal{O}$ is given; later, we describe how our framework automatically constructs $\mathcal{O}$. In particular, given subgoal regions $\tilde{\mathcal{S}}$ and edges $E \subseteq \tilde{\mathcal{S}} \times \tilde{\mathcal{S}}$, it constructs options $o \in \mathcal{O}$ to serve as transitions between subgoal regions $(\tilde{s}, \tilde{s}') \in E$. The edges E can be used to constrain the abstract actions; by default, $E = \tilde{\mathcal{S}} \times \tilde{\mathcal{S}}$.

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

Assumption 2.2. The underlying MDP $\mathcal{M}$ has deterministic transitions $P : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. 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 $a \in \mathcal{A}$, $P(s, a) = s$, and (ii) the rewards are 1 if transitioning to $\tilde{s}_g$ and 0 otherwise—i.e., $r(s, a) = \mathbb{1}(s \notin \tilde{s}_g \wedge P(s, a) \in \tilde{s}_g)$.

3 Robust Abstract Value Iteration

First, we propose *robust abstract value iteration (R-AVI)*, which computes an abstract option policy $\tilde{\rho}$ by bounding the degree to which the abstract MDP fails to be Markov, and then plans conservatively according to this bound. We prove two performance guarantees for R-AVI: (i) it is close to the optimal option policy when the failure degree is small, and (ii) it is close to the optimal concrete policy for $\mathcal{M}$ if the subgoal regions are bottlenecks and $\mathcal{M}$ has sparse rewards.

Algorithm. Intuitively, R-AVI performs option value iteration, but approximates the values of all concrete states in a given subgoal region with a single value. A challenge is that the (time-discounted) transition probabilities $P(s' \mid s, o)$ depend on the initial state s and the final state s' in $\mathcal{M}$. Thus, for two subgoal regions $\tilde{s}, \tilde{s}' \in \tilde{\mathcal{S}}$, the probability of transitioning from $\tilde{s}$ to $\tilde{s}'$ depends on the starting state $s \in \tilde{s}$ and the ending state $s' \in \tilde{s}'$. This dependence is a consequence of the fact that $\mathcal{M}$ does not satisfy the Markov condition. Instead, R-AVI uses upper and lower bounds on the transition probabilities over $s \in \tilde{s}$. In particular, for $s \in S$, $o \in \mathcal{O}$ and $\tilde{s}' \in \tilde{\mathcal{S}}$, let

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

where $p(\tilde{s}', t \mid s, o)$ is the probability that option o terminates in region $\tilde{s}'$ after t steps when starting from concrete state $s \in \tilde{s}$. Now, for $\tilde{s} \in \tilde{\mathcal{S}}$, let

$$\tilde{P}_{\text{inf}}(\tilde{s}' \mid \tilde{s}, o) = \inf_{s \in \tilde{s}} \tilde{P}(\tilde{s}' \mid s, o) \qquad \text{and} \qquad \tilde{P}_{\text{sup}}(\tilde{s}' \mid \tilde{s}, o) = \sup_{s \in \tilde{s}} \tilde{P}(\tilde{s}' \mid s, o).$$

R-AVI similarly uses upper and lower bounds on the rewards—i.e., for $\tilde{s} \in \tilde{\mathcal{S}}$ and $o \in \mathcal{O}$, we have

$$\tilde{r}_{\text{inf}}(\tilde{s}, o) = \inf_{s \in \tilde{s}} r_{\text{opt}}(s, o) \qquad \text{and} \qquad \tilde{r}_{\text{sup}}(\tilde{s}, o) = \sup_{s \in \tilde{s}} r_{\text{opt}}(s, o).$$

Given these bounds, R-AVI computes $\tilde{V}_z^* : \tilde{\mathcal{S}} \to \mathbb{R}$ using value iteration to solve the equations

$$\tilde{V}_z^*(\tilde{s}) = \max_{o \in \mathcal{O}} \tilde{Q}_z^*(\tilde{s}, o) \qquad \text{where} \qquad \tilde{Q}_z^*(\tilde{s}, o) = \tilde{r}_z(\tilde{s}, o) + \sum_{\tilde{s}' \in \tilde{\mathcal{S}}} \tilde{P}_z(\tilde{s}' \mid \tilde{s}, o) \cdot \tilde{V}_z^*(\tilde{s}'),$$

where $z \in \{\text{inf}, \text{sup}\}$. Then, the *conservative optimal option policy* is given by $\tilde{\rho}(s) = \arg\max_{o \in \mathcal{O}} \tilde{Q}_{\text{inf}}^*(\tilde{s}, o)$, for all $s \in \tilde{s}$ and $\tilde{s} \in \tilde{\mathcal{S}}$. Note that $\tilde{\rho}$ is only defined on $\bar{\mathcal{S}}$. In addition, it is based on the lower-bounds—i.e., it conservatively makes worst-case assumptions about the transitions and rewards (we define the upper bounds since they are needed for our theoretical guarantees).

Constructing subgoal transitions. So far, we have assumed the options are given. Given subgoal regions $\tilde{\mathcal{S}}$ and edges $E \subseteq \tilde{\mathcal{S}} \times \tilde{\mathcal{S}}$, our framework automatically constructs subgoal transitions that serve as transitions between subgoal regions. In particular, it constructs the set of *ideal subgoal transitions*

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

where β is as in Definition 2.1, and

$$\pi(\tilde{s}, \tilde{s}') = \arg\max_{\pi} \tilde{P}_{\text{sup}}(\tilde{s}' \mid \tilde{s}, (\pi, \tilde{s}, \beta)) \tag{1}$$

maximizes the best-case reward for transitioning from $\tilde{s}$ to $\tilde{s}'$ over initial concrete states $s \in \tilde{s}$.

Bound vs. ρ^*. We bound the performance of $\tilde{\rho}$ compared to the optimal option policy ρ^*. Let

$$\varepsilon_P = \sup_{\tilde{s},\tilde{s}' \in \tilde{\mathcal{S}}, o \in \mathcal{O}} \tilde{P}_{\text{sup}}(\tilde{s}' \mid \tilde{s}, o) - \tilde{P}_{\text{inf}}(\tilde{s}' \mid \tilde{s}, o) \qquad \text{and} \qquad \varepsilon_r = \sup_{\tilde{s} \in \tilde{\mathcal{S}}, o \in \mathcal{O}} \tilde{r}_{\text{sup}}(\tilde{s}, o) - \tilde{r}_{\text{inf}}(\tilde{s}, o),$$

i.e., ε_P is the worst-case difference between $\tilde{P}_{\text{sup}}$ and $\tilde{P}_{\text{inf}}$, and ε_r is the worst-case difference between $\tilde{r}_{\text{sup}}$ and $\tilde{r}_{\text{inf}}$.[3] We assume that ε_P is not too large.

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

Then, our first result shows that abstract value iteration converges.

Theorem 3.2. *Under Assumption 3.1, R-AVI converges.*

Our second result says that $\tilde{\rho}$ has performance close to that of ρ^*.

Theorem 3.3. *Under Assumption 3.1, we have* $J(\pi_{\tilde{\rho}}) \geq J(\pi_{\rho^*}) - \frac{(1-\gamma)\varepsilon_r + |\tilde{\mathcal{S}}|\varepsilon_P}{(1-\gamma)(1-(\gamma+|\tilde{\mathcal{S}}|\varepsilon_P))}$.

Bound vs. π^*. Theorem 3.3 is for a fixed set of options $\mathcal{O}$—i.e., both $\tilde{\rho}$ and ρ^* use $\mathcal{O}$. In general, the choice of $\mathcal{O}$ determines the how $\pi_{\tilde{\rho}}$ compares to the optimal policy π^* for the underlying MDP $\mathcal{M}$. Suppose Assumption 2.2 holds; then, we can prove that $\tilde{\rho}$ constructed using the ideal subgoal transitions $\mathcal{O}^*$ performs nearly as well as π^* under the *bottleneck assumption*.

Assumption 3.4. For any trajectory $(s_0, a_0, s_1, a_1, \ldots, a_{t-1}, s_t)$ 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 $j < k$, there is an edge $(\tilde{s}_j, \tilde{s}_{j+1}) \in E$.

Intuitively, this assumption says that the subgoal regions $\tilde{\mathcal{S}}$ act as bottlenecks when reaching the final states from the initial states—i.e., optimal paths from initial states to final states can be represented as taking sequences of subgoal transitions. We can now bound the performance of the conservative optimal option policy $\tilde{\rho}$ obtained using abstract value iteration with the set of options $\mathcal{O}^*$.

Theorem 3.5. *Under Assumptions 2.2, 3.1, & 3.4, we have* $J(\pi_{\tilde{\rho}}) \geq J(\pi^*) - \frac{(1-\gamma)\varepsilon_r + |\tilde{\mathcal{S}}|\varepsilon_P}{(1-\gamma)(1-(\gamma+|\tilde{\mathcal{S}}|\varepsilon_P))}$.

Unlike Theorem 3.3, this result bounds the performance compared to the optimal policy π^* for the underlying MDP $\mathcal{M}$; the shortcoming is that it relies on additional assumptions.

Implications. Our results establish two conditions on $\tilde{\mathcal{S}}$ such that $\tilde{\rho}$ performs nearly as well as π^*: (i) Theorem 3.3 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 3.5 suggests that subgoal regions should be bottlenecks in the underlying MDP.

4 Alternating Abstract Value Iteration

There are two shortcomings of R-AVI. First, exactly computing $\mathcal{O}^*$, $\tilde{P}_{\text{inf}}$, and $\tilde{r}_{\text{inf}}$ can be computationally infeasible since they involve solving a potentially non-convex optimization problem. Second, making conservative assumptions about the defects of the abstract MDP can lead to suboptimal performance.

Thus, we propose *alternating abstract value iteration (A-AVI)* (shown in Algorithm 1), a practical variant of R-AVI. A-AVI maintains a set of distributions $\mathcal{D}$; for each subgoal region $\tilde{s} \in \tilde{\mathcal{S}}$, $\mathcal{D}$ contains a distribution $\mathcal{D}_{\tilde{s}}$ over concrete states $s \in \tilde{s}$. Then, it uses $\mathcal{D}$ both to estimate the ideal subgoal transitions $\mathcal{O}^*$ as well as the abstract MDP. First, we define the ideal subgoal transitions with respect to $\mathcal{D}$ as

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

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

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

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

where

$$\pi_{\mathcal{D}}(\tilde{s}, \tilde{s}') = \arg\max_{\pi} \tilde{P}_{\mathcal{D}}(\tilde{s}' \mid \tilde{s}, (\pi, \tilde{s}, \beta)), \tag{2}$$

and where $\tilde{P}_{\mathcal{D}}$, defined below in (3), is the expected (time-discounted) probability of transitioning to $\tilde{s}'$ from a concrete state $s \sim \mathcal{D}_{\tilde{s}}$ chosen randomly from $\tilde{s}$ according to $\mathcal{D}$. Intuitively, $\pi_{\mathcal{D}}(\tilde{s}, \tilde{s}')$ is the policy that maximizes probability of transitioning to $\tilde{s}'$ from $\tilde{s}$. Then, the optimization problem (2) can be formulated as the RL problem for the MDP $(\mathcal{S}, \mathcal{A}, P', r', \eta_0', \gamma)$, where

$$P'(s' \mid s, a) = \begin{cases} P(s' \mid s, a) & \text{if } s \notin \bar{\mathcal{S}} \setminus \tilde{s} \\ \mathbb{1}(s' = s) & \text{otherwise,} \end{cases}$$

$r'(s, a) = \mathbb{1}(s \notin \tilde{s}') \sum_{s' \in \tilde{s}'} P(s' \mid s, a)$, and $\eta_0' = \mathcal{D}_{\tilde{s}}$—i.e., the underlying MDP $\mathcal{M}$ except where all states in $\bar{\mathcal{S}} \setminus \tilde{s}$ are sinks, the rewards are given by r' encoding that $\tilde{s}'$ is the goal, and the initial state distribution is $\eta_0' = \mathcal{D}_{\tilde{s}}$. Therefore, existing RL algorithms can be used to learn these policies.

Second, A-AVI plans in the abstract MDP $\tilde{\mathcal{M}}_{\mathcal{D}}$ whose transitions and rewards are the expected values of the abstract (time-discounted) transition probabilities $\tilde{P}$ and rewards r_{opt}, respectively—i.e.,

$$\tilde{P}_{\mathcal{D}}(\tilde{s}' \mid \tilde{s}, o) = \mathbb{E}_{s \sim \mathcal{D}_{\tilde{s}}}[\tilde{P}(\tilde{s}' \mid s, o)] \qquad \text{and} \qquad \tilde{r}_{\mathcal{D}}(\tilde{s}, o) = \mathbb{E}_{s \sim \mathcal{D}_{\tilde{s}}}[r_{\text{opt}}(s, o)], \tag{3}$$

which can be estimated using sampled rollouts. Now, A-AVI returns policy $\rho_{\mathcal{D}} : \bar{\mathcal{S}} \to \mathcal{O}$ defined by $\rho_{\mathcal{D}}(s) = \arg\max_{o \in \mathcal{O}} \tilde{Q}^*_{\mathcal{D}}(\tilde{s}, o)$ for all $s \in \tilde{s}$ and $\tilde{s} \in \tilde{\mathcal{S}}$, where it uses value iteration to solve

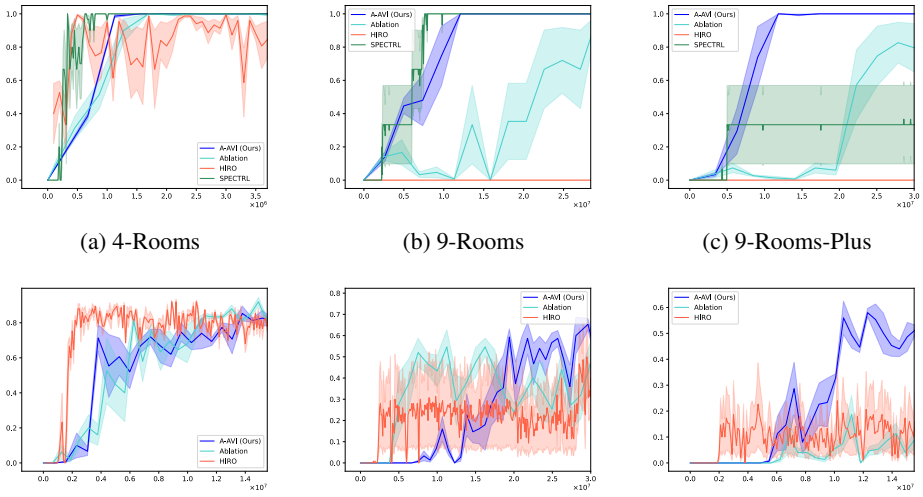
$$\tilde{V}^*_{\mathcal{D}}(\tilde{s}) = \max_{o \in \mathcal{O}} \tilde{Q}^*_{\mathcal{D}}(\tilde{s}, o) \qquad \text{and} \qquad \tilde{Q}^*_{\mathcal{D}}(\tilde{s}, o) = \tilde{r}_{\mathcal{D}}(\tilde{s}, o) + \sum_{\tilde{s}' \in \tilde{\mathcal{S}}} \tilde{P}_{\mathcal{D}}(\tilde{s}' \mid \tilde{s}, o) \cdot \tilde{V}^*_{\mathcal{D}}(\tilde{s}').$$

It remains to choose $\mathcal{D}$. A natural goal is for $\mathcal{D}_{\tilde{s}}$ to equal the state distribution over $\tilde{s}$ induced by $\pi_{\rho_{\mathcal{D}}}$—i.e., $\mathcal{D}_{\tilde{s}} = \bar{\mathcal{D}}_{\tilde{s}}$ for all $\tilde{s} \in \tilde{\mathcal{S}}$, where $\bar{\mathcal{D}}_{\tilde{s}}$ is the state distribution induced by $\pi_{\rho_{\mathcal{D}}}$ over $\tilde{s}$. Intuitively, this condition says that $\pi_{\rho_{\mathcal{D}}}$ is used on the same state distribution as it was trained.

To achieve this goal, A-AVI alternates between (i) computing the option policy $\rho_{\mathcal{D}}$ for $\mathcal{D}$, and (ii) computing the state distributions $\bar{\mathcal{D}}_{\tilde{s}}(\rho_{\mathcal{D}})$ induced by $\pi_{\rho_{\mathcal{D}}}$. Finally, at the end of each iteration, it updates $\mathcal{D}_{\tilde{s}} = (1 - \alpha_i)\mathcal{D}_{\tilde{s}} + \alpha_i \bar{\mathcal{D}}_{\tilde{s}}(\rho_{\mathcal{D}})$. This update rule builds on the principle of dataset aggregation motivated by the follow-the-leader algorithm for online learning to facilitate convergence [36]—i.e., it aggregates distributions $\mathcal{D}_{\tilde{s}}$ over iterations. Finally, in practice, we approximate $\bar{\mathcal{D}}_{\tilde{s}}(\rho_{\mathcal{D}})$ as the empirical state distribution over $\tilde{s}$ obtained by sampling rollouts using $\pi_{\rho_{\mathcal{D}}}$.

5 Experiments

We evaluate our approach on two continuous control benchmarks: (i) a benchmark of environments where a robot must navigate a maze of rooms, which is a continuous variant of standard hierarchical RL benchmarks [17, 2] and (ii) a hierarchical RL benchmark [34] based on the MuJoCo ant environment [44]. The room environments have comparatively simple continuous dynamics, but the



(a) 4-Rooms (b) 9-Rooms (c) 9-Rooms-Plus

(d) AntMaze (e) AntPush (f) AntFall

Figure 2: Comparison with hierarchical RL 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 3 executions.

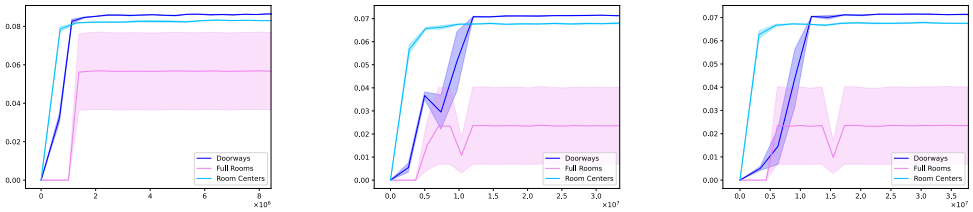
high-level task is very challenging due to the highly nonlinear structure of the state space. In contrast, the ant environment tasks have complex robot dynamics, but comparatively simple high level tasks. We demonstrate that our approach, A-AVI, outperforms state-of-the-art baselines in both cases.

Room environment. We consider three environments 4-Rooms, 9-Rooms, and 9-Rooms-Plus consisting of interconnected rooms; 9-Rooms-Plus is shown in Figure 1. They have (continuous) states $(x, y) \in \mathbb{R}^2$ encoding 2D position, actions $(v, \theta) \in \mathbb{R}^2$ encoding speed and direction, and transitions $s' = s + (v\cos(\theta), v\sin(\theta))$. Subgoal regions are the light gray squares; edges connect adjacent subgoal regions. The agent starts uniformly randomly in the initial region (the green triangle); its goal is to reach the goal region (the red star). The rewards are sparse—i.e., the agent receives a reward of 1 upon reaching the goal. While the dynamics are simple, the problem is challenging due to the non-convexity of the objective. We learn subgoal transitions using ARS [32], with a shaped reward equal to the negative distance to the center of the target region plus the sparse reward encoding (2); each policy is a fully connected neural network with 2 hidden layers with 30 neurons each. We give additional details on this task in Appendix.

Ant tasks. We also consider three MuJoCo [44] ant tasks from [34]: 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 4 subgoal regions: the initial region, an intermediate region at each of the two turns, and the goal region. AntPush has 5 subgoal regions: the initial region, 2 at the bottom-left and top-left corners, 1 at the gap where the ant must enter to reach the goal, and the goal region; in the abstract MDP, there are 3 paths from the initial region to the goal region. AntFall has 5 subgoal regions: the initial region, 3 along the path to the goal region, and the goal region. We learn subgoal transitions using TD3 [16]. We give additional details, along with visualizations of the subgoal regions, in Appendix.

Results. We show results on all benchmarks in Figure 2. The top row shows the room environments, and the bottom row shows the ant tasks. In each row, environments are ordered from simplest to most complex in terms of the abstract MDP—i.e., the number of subgoal transitions needed to complete the task. Overall, our approach tends to perform better compared to our baselines as the abstract MDP becomes more complex; in particular, it substantially outperforms all baselines for the 9-Rooms-Plus and AntFall environments. We discuss the comparisons in more detail below.

Comparison to HIRO. We compare to HIRO [34], a state-of-the-art hierarchical deep RL algorithm that uses a high-level policy to generate intermediate goals that a low-level policy aims to achieve.



(a) 4-Rooms (b) 9-Rooms (c) 9-Rooms-Plus

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 3 executions.

As with our algorithm, we used shaped rewards—i.e., the negative distance from the current state to the center of the goal region; we obtained worse results without shaped rewards.

As can be seen in Figure 2, HIRO performs similarly to our approach for 4-Rooms, but performs substantially worse in 9-Rooms and 9-Rooms-Plus. Because HIRO does not have any knowledge of the structure of the search space, it is unable to discover the path from the initial region to the goal region and gets stuck at a local optimum. Intuitively, HIRO is designed to decouple complex dynamics (e.g., an ant model) from high-level planning (e.g., sequence of tasks). However, it is not designed to solve challenging high-level planning problems.

Comparison to SpectRL. We compare to SpectRL [22], where the user specifies a task as sequences of subgoals. It can be thought of as a hierarchical RL framework that uses options but not state abstractions. For each room environment, we specify the task as a sequence of subgoals each centered at a subgoal region along the path from the initial region to the goal region. For the 9-Rooms-Plus environment, we encode the choice of paths using a disjunction operator in SpectRL.

As can be seen in Figure 2, SpectRL performs similarly to our approach for 4-Rooms and 9-Rooms, since it is essentially given the high-level policy. For 9-Rooms-Plus, it has a choice between two paths; it converges to the wrong path, yielding suboptimal reward. 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 that sets $\mathcal{D}_{\tilde{s}}$ to be the uniform distribution over $\tilde{s}$ and does not update it. This ablation can be thought of as extending [17] to learn the transitions and rewards of the abstract MDP, or extending [46] to handling continuous state spaces.

As can be seen in Figure 2, except on the simpler Room-4 environment, our ablation performs quite poorly, likely because it is unable to account for the non-Markov nature of the abstract MDP. 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 uniform distribution over the regions.

Choice of subgoal regions. We study the impact of the choice of subgoal regions on performance for the rooms environment. Our choice of "doorways" is motivated by our theory, which suggests that subgoal regions should be bottlenecks that are small in size. We consider two alternatives: (i) "room center" consists of a square at the center of each room with 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 [2]).

Results are in Figure 3. "Doorways" achieve the highest reward in all environments, validating our theory. "Full rooms" performs poorly, likely because the large regions induce large defects in the abstract MDP. "Room centers" converges quickly but to a suboptimal value, likely because the robot cannot travel diagonally across rooms (e.g., the $1 \rightarrow 4$ transition in Figure 1(a)). Using smaller subgoal regions is crucial for good performance, whereas using bottlenecks is less important.

Conclusions. We have proposed a hierarchical RL algorithm that accounts for defects in the abstract MDP during planning. Future work includes applying our results to more complex benchmarks, and designing ways to transfer state and action abstractions to new tasks.

References

[1] David Abel, Dilip Arumugam, Kavosh Asadi, Yuu Jinnai, Michael L Littman, and Lawson LS Wong. State abstraction as compression in apprenticeship learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3134–3142, 2019.

[2] David Abel, Nathan Umbanhowar, Khimya Khetarpal, Dilip Arumugam, Doina Precup, and Michael L. Littman. Value preserving state-action abstractions. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2020.

[3] David Andre and Stuart J Russell. State abstraction for programmable reinforcement learning agents. In *AAAI/IAAI*, pages 119–125, 2002.

[4] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[5] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

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

[7] Pablo Samuel Castro. Scalable methods for computing state similarity in deterministic markov decision processes. *arXiv preprint arXiv:1911.09291*, 2019.

[8] Pablo Samuel Castro and Doina Precup. Automatic construction of temporally extended actions for mdps using bisimulation metrics. In *European Workshop on Reinforcement Learning*, pages 140–152. Springer, 2011.

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

[10] Luis Carlos Cobo, Peng Zang, Charles Lee Isbell Jr, and Andrea Lockerd Thomaz. Automatic state abstraction from demonstration. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[11] Christian Daniel, Herke Van Hoof, Jan Peters, and Gerhard Neumann. Probabilistic inference for determining options in reinforcement learning. *Machine Learning*, 104(2-3):337–357, 2016.

[12] Thomas G Dietterich. State abstraction in maxq hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 994–1000, 2000.

[13] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *ICLR*, 2018.

[14] Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite markov decision processes. In *UAI*, volume 4, pages 162–169, 2004.

[15] Chelsea Finn, Tianhe Yu, Justin Fu, Pieter Abbeel, and Sergey Levine. Generalizing skills with semi-supervised reinforcement learning. In *ICLR*, 2017.

[16] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596, 2018.

[17] Nakul Gopalan, Michael L Littman, James MacGlashan, Shawn Squire, Stefanie Tellex, John Winder, Lawson LS Wong, et al. Planning with abstract markov decision processes. In *Twenty-Seventh International Conference on Automated Planning and Scheduling*, 2017.

[18] Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *ICLR*, 2018.

[19] Dinesh Jayaraman, Frederik Ebert, Alexei A Efros, and Sergey Levine. Time-agnostic prediction: Predicting predictable video frames. In *ICLR*, 2019.

[20] Nicholas K Jong and Peter Stone. State abstraction discovery from irrelevant state variables. In *IJCAI*, volume 8, pages 752–757, 2005.

[21] Anders Jonsson and Andrew G Barto. Automated state abstraction for options using the u-tree algorithm. In *Advances in neural information processing systems*, pages 1054–1060, 2001.

[22] Kishor Jothimurugan, Rajeev Alur, and Osbert Bastani. A composable specification language for reinforcement learning tasks. In *Advances in Neural Information Processing Systems*, pages 13021–13030, 2019.

[23] Arbaaz Khan, Ekaterina Tolstaya, Alejandro Ribeiro, and Vijay Kumar. Graph policy gradients for large scale robot control. In *CoRL*, 2019.

[24] George Konidaris and Andrew G Barto. Building portable options: Skill transfer in reinforcement learning. In *IJCAI*, volume 7, pages 895–900, 2007.

[25] George Konidaris and Andrew G Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. In *Advances in neural information processing systems*, pages 1015–1023, 2009.

[26] George Konidaris, Leslie Kaelbling, and Tomas Lozano-Perez. Constructing symbolic representations for high-level planning. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[27] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, pages 3675–3683, 2016.

[28] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[29] Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. In *ISAIM*, 2006.

[30] Marios C Machado, Marc G Bellemare, and Michael Bowling. 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, 2017.

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

[32] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search of static linear policies is competitive for reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1800–1809, 2018.

[33] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Near-optimal representation learning for hierarchical reinforcement learning. In *ICLR*, 2019.

[34] Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 3303–3313, 2018.

[35] Doina Precup, Richard S Sutton, and Satinder Singh. Theoretical results on reinforcement learning with temporally abstract options. In *European conference on machine learning*, pages 382–393. Springer, 1998.

[36] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 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, 2011.

[37] Martin Stolle and Doina Precup. Learning options in reinforcement learning. In *International Symposium on abstraction, reformulation, and approximation*, pages 212–223. Springer, 2002.

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

[39] Adrien Ali Taïga, Aaron Courville, and Marc G Bellemare. Approximate exploration through state abstraction. *arXiv preprint arXiv:1808.09819*, 2018.

[40] Jonathan Taylor, Doina Precup, and Prakash Panagaden. Bounding performance loss in approximate mdp homomorphisms. In *Advances in Neural Information Processing Systems*, pages 1649–1656, 2009.

[41] Georgios Theocharous and Leslie P Kaelbling. Approximate planning in pomdps with macro-actions. In *Advances in Neural Information Processing Systems*, pages 775–782, 2004.

[42] Georgios Theocharous, Sridhar Mahadevan, and Leslie P Kaelbling. Spatial and temporal abstractions in pomdps applied to robot navigation. Technical report, MIT, 2005.

[43] Saket Tiwari and Philip S Thomas. Natural option critic. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5175–5182, 2019.

[44] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[45] Thomas J Walsh, Lihong Li, and Michael L Littman. Transferring state abstractions between mdps. In *ICML Workshop on Structural Knowledge Transfer for Machine Learning*, 2006.

[46] John Winder, Stephanie Milani, Matthew Landen, Erebus Oh, Shane Parr, Shawn Squire, Marie desJardins, and Cynthia Matuszek. Planning with abstract learned models while learning transferable subtasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 2020.