# Robustly Learning Composable Options in Deep Reinforcement Learning

**Akhil Bagaria*** , **Jason Senthil*** , **Matthew Slivinski** and **George Konidaris**
Department of Computer Science, Brown University
{akhil_bagaria, jason_senthil, matthew_slivinski}@brown.edu, gdk@cs.brown.edu

## Abstract

Hierarchical reinforcement learning (HRL) is only effective for long-horizon problems when high-level skills can be reliably sequentially executed. Unfortunately, learning reliably composable skills is difficult, because all the components of every skill are constantly changing during learning. We propose three methods for improving the composability of learned skills: representing skill initiation regions using a combination of pessimistic and optimistic classifiers; learning re-targetable policies that are robust to non-stationary subgoal regions; and learning robust option policies using model-based RL. We test these improvements on four sparse-reward maze navigation tasks involving a simulated quadrupedal robot. Each method successively improves the robustness of a baseline skill discovery method, substantially outperforming state-of-the-art flat and hierarchical methods.

## 1 Introduction

Temporal abstraction is a promising approach to scaling reinforcement learning (RL) algorithms [Botvinick *et al.*, 2009] because it is capable of addressing some of the biggest challenges in RL: structured exploration [Jinnai *et al.*, 2019], transfer [Konidaris and Barto, 2007] and long-term credit assignment [Dietterich, 2000]. Hierarchical methods are particularly attractive for long-horizon, sparse reward tasks, where flat (non-hierarchical) RL algorithms often struggle.

Hierarchical approaches succeed by allowing the agent to sequentially execute high-level actions. This intuition has led to several skill-discovery algorithms that explicitly satisfy the composability objective: executing one skill takes the agent to a state where it can execute another. Such algorithms are widespread in the literature; from control-theory [Lyapunov, 1992; Burridge *et al.*, 1999; Tedrake, 2009], to robotics [Lozano-Perez *et al.*, 1984; Kaelbling and Lozano-Pérez, 2017], to the online [Randløv *et al.*, 2000; Konidaris and Barto, 2009; Shoeleh and Asadpour, 2017; Bagaria and Konidaris, 2020] and batch RL [Singh *et al.*, 2020] settings.

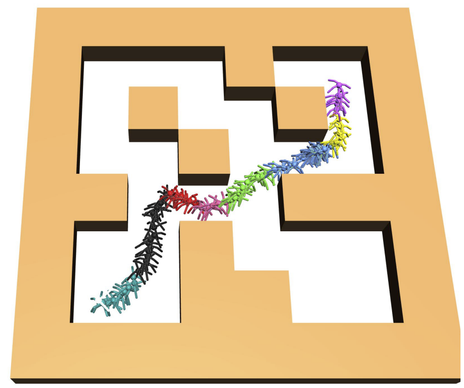


Figure 1: A learned solution to a long-horizon, sparse reward task that requires executing approximately 665 low-level steps (different colors denote discovered skills). Our proposed methods result in discovered skills that can be reliably sequentially executed, enabling skill discovery methods to more effectively solve such tasks.

Even when skills are not explicitly constructed to be sequentially executable, they must eventually be sequenced to solve goal-directed tasks [Sharma *et al.*, 2020; Frans *et al.*, 2018]. Additionally, skills that can be reliably sequenced can support abstract, high-level planning [Konidaris *et al.*, 2018].

The core difficulty that arises when discovering composable skills is that of *non-stationary subgoals*. Skill-discovery algorithms that explicitly optimize for composability must learn the regions (called *initiation sets* [Sutton *et al.*, 1999]) from which each skill can be successfully executed. To construct sequentially executable skills, the initiation set of one skill becomes the subgoal of a new skill [Konidaris and Barto, 2009]. These initiation sets are constantly changing as the agent learns, causing the target (and therefore reward function) of successive skills to also change over time, destabilizing composability and complicating learning.

We propose three methods to combat this problem. First, we propose to stabilize learned initiation sets using a dual-classifier approach. An optimistic classifier determines when the agent can execute the skill, which encourages exploration. Meanwhile, a pessimistic classifier is used as a subgoal target, which is likely to grow but unlikely to shrink, leading to stable chains. Next, to create skill policies that are robust to

*Equal contribution

Video, code and appendix can be found at https://sites.google.com/brown.edu/robustly-composing-options non-stationary subgoals, we propose to use goal-conditioned policies [Schaul *et al.*, 2015] for each skill. Such policies are robust to subgoal changes because they can always be retargeted towards a state in the new subgoal region. Finally, while the majority of skill-discovery work has been model-free, we show that using model-based RL to learn skill policies leads to a substantially more robust skills.

To evaluate our proposed changes, we consider four sparse-reward continuous control problems in MuJoCo [Todorov *et al.*, 2012]. Compared to a flat model-free solution [Fujimoto *et al.*, 2018; Andrychowicz *et al.*, 2017], our agent achieves at least two orders of magnitude better sample efficiency. While a flat model-based solution [Nagabandi *et al.*, 2018] is unable to solve any of the problems considered in this paper, our model-based hierarchy solves them with relative ease. Finally, our agent achieves at least a $5\times$ improvement in sample-efficiency over a state-of-the-art skill-discovery algorithm [Bagaria and Konidaris, 2020].

# 2 Background and Related Work

We consider decision making problems modelled as episodic, goal-oriented MDPs $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma, \mathcal{G})$ [Sutton and Barto, 2018], where $\mathcal{G}$ refers to a set of goals that the agent could be asked to reach, whereupon execution terminates with a non-negative reward. Like most goal-conditioned RL algorithms [Schaul *et al.*, 2015; Andrychowicz *et al.*, 2017], we assume access to a reward function $\mathcal{R} : \mathcal{S} \times \mathcal{G} \rightarrow \mathbb{R}$.

**Model-Free RL.** Model-free methods often use Q-learning [Watkins and Dayan, 1992] to learn $Q^{\pi}(s_t, a_t)$, estimating the expected sum of discounted future rewards conditioned on taking action $a_t$ from $s_t$, thereafter following policy $\pi$. The policy $\pi(s_t) = \arg\max_{a \in A} Q(s_t, a)$ chooses an action that greedily maximizes the Q-function at a given state. The Q-function is often represented by a neural network $\phi$ [Mnih *et al.*, 2015] and the policy by another neural network $\psi$ [Lillicrap *et al.*, 2015]. Many algorithms exist for learning $\phi$ and $\psi$ from interactions with $\mathcal{M}$ [Duan *et al.*, 2016].

**Hindsight Experience Replay (HER).** When the class of policies we wish to learn in $\mathcal{M}$ is restricted to goal-reaching policies [Kaelbling, 1993], HER can improve sample efficiency as follows. Given a trajectory $\tau_1 = (s_1, a_1, \mathcal{R}(s_2, g), s_2), ..., (s_n, a_n, \mathcal{R}(s_{n+1}, g), s_{n+1})$ executed while trying to reach goal $g$, HER [Andrychowicz *et al.*, 2017] replays $\tau_1$ assuming that the agent was trying to reach goal $s_{n+1}$, by augmenting its set of experiences with another trajectory $\tau_2 = (s_1, a_1, \mathcal{R}(s_2, s_{n+1}), s_2), ..., (s_n, a_n, \mathcal{R}(s_{n+1}, s_{n+1}), s_{n+1})$. Replaying trajectory $\tau_2$ trains the agent to reach state $s_{n+1}$, even when the reward function $\mathcal{R}$ is sparse.

**Model-Based RL.** While Q-learning directly learns the value function for control, model-based methods first learn an approximate model of the system dynamics. This model $f_\xi : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is often represented using another neural network $\xi$ [Nagabandi *et al.*, 2018]. Model-based methods can then select actions by solving:

$$\pi(s|g) = \underset{a \in \mathcal{U}^{|A|}(-1,1)}{\arg\max} \sum_{t=1}^{H} \gamma^{t-1} R(s_t, g), \tag{1}$$

such that $s_{t+1} = f_\xi(s_t, a)$. In Model Predictive Control (MPC) [Garcia *et al.*, 1989], the agent approximately solves this optimization problem at every time-step, executing only the first action from the resulting action sequence. By re-planning at every time-step, MPC is robust to the prediction errors that compound over time with a learned model [Asadi *et al.*, 2019].

## 2.1 Hierarchical Reinforcement Learning

The standard RL formulation considers an agent selecting primitive actions at every time-step. In hierarchical RL [Barto and Mahadevan, 2003], the agent instead selects temporally extended actions or *skills*. These skills are commonly modeled as options [Sutton *et al.*, 1999], where each option $o$ is described using three elements: the initiation region, $\mathcal{I}_o : \mathcal{S} \rightarrow \{0, 1\}$, describing the set of states from which the option can be executed; the termination region, $\beta_o : \mathcal{S} \rightarrow \{0, 1\}$, describing the subgoal region in which option execution must terminate; and the policy, $\pi_o : \mathcal{S} \rightarrow A$, which drives the agent from $\mathcal{I}_o$ to $\beta_o$. Several methods have been proposed to autonomously discover useful options (see Abel [2020], chapter 2.3, for a survey).

**Skill Chaining.** For two options $o_i$ and $o_j$ to be reliably sequentially executable, it must be the case that $\beta_{o_i} \subseteq \mathcal{I}_{o_j}$. The skill-chaining algorithm [Konidaris and Barto, 2009] explicitly satisfies this property by learning options such that $\beta_{o_i} = \mathcal{I}_{o_j}$. In skill-chaining, each discovered option $o$ learns its initiation region $\mathcal{I}_{o,\theta}$ using a binary classifier $\theta$ that describes the region from which the $\pi_o$ reaches $\beta_o$ with high probability. Simultaneously, $\pi_o$ is learned using a model-free RL algorithm to reach its subgoal region $\beta_o$. The algorithm is recursive: it first learns an option that initiates near the goal and reliably takes the agent to the goal; then it learns another option whose termination region is the initiation region of the first option; then it repeats the procedure targeting the new option. Options are chained together in this fashion until the agent discovers an option whose initiation region contains the start state. Deep skill chaining (DSC) [Bagaria and Konidaris, 2020] extended skill-chaining with deep RL, outperforming existing state-of-the-art skill-discovery algorithms [Levy *et al.*, 2019; Bacon *et al.*, 2017].

Since DSC *explicitly* constructs composable options, we build on top of it to show that our proposed augmentations can substantially improve the reliability of the resulting options. However, since any skill-discovery algorithm must eventually compose learned skills to solve goal-directed tasks, our insights could, at least in principle, improve the robustness of their solutions.

## 2.2 Related Work

**Robustness in HRL.** Several variants of the Option-Critic architecture [Bacon *et al.*, 2017] have showcased robustness to changes in transition dynamics or the reward function [Mankowitz *et al.*, 2018; Khetarpal and Precup, 2019; Tiwari and Thomas, 2019; Jain *et al.*, 2021]. By contrast, we seek to improve the reliability of hierarchical methods in stationary, sparse-reward MDPs.

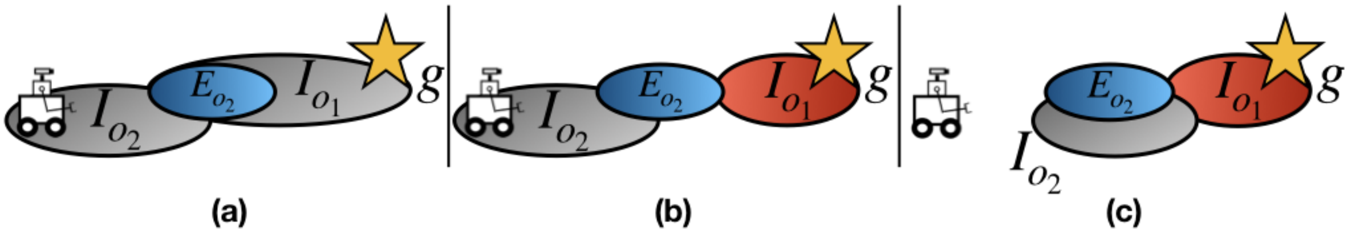



Figure 2: The non-stationary subgoal problem: (a) an agent has two options $o_1, o_2$ such that $o_2$ targets $I_{o_1}$ and $o_1$ targets goal $g$; $E_{o_2}$ is the next state distribution of $\pi_{o_2}$. (b) If $I_{o_1}$ shrinks, executing $o_2$ no longer allows the agent to execute $o_1$ (since $E_{o_2} \not\subseteq I_{o_1}$). (c) This causes $\beta_{o_2}$ to shift forward, which invalidates the previous policy $\pi_{o_2}$, which in turn causes $\mathcal{I}_{o_2}$ to shrink.

**Composing Options.** Some recent work has studied the problem of composing skills as linear [Barreto *et al.*, 2019] or non-linear [Qureshi *et al.*, 2020] combinations of options available to the agent. These methods require pre-trained options, while we discover them from scratch.

**Model-Based Skill Discovery.** Empowerment-driven skill-discovery methods [Gregor *et al.*, 2016; Eysenbach *et al.*, 2019] have recently been augmented with model-based RL via the DADS algorithm [Sharma *et al.*, 2020]. This led to a family of related methods that optimize for exploration [Campos Camúñez *et al.*, 2020], lifelong learning [Lu *et al.*, 2021], and skill acquisition in image-based observation spaces [Baumli *et al.*, 2020]. These methods represent substantial progress in unsupervised skill-discovery, but our setting differs from theirs in a few ways. First, unlike these methods, we do not require a special pre-training phase for skill-discovery. Second, while DADS is designed for the multi-task setting, we focus on creating more robust solutions to single-goal MDPs. Finally, skills learned by DADS have global support, whereas we consider composability for skills that specialize in different regions.

## 3 Robustly Learning Composable Options

The core difficulty in learning composable options is that, during learning, all three components $(\mathcal{I}_o, \beta_o, \pi_o)$ of every option are simultaneously in flux. This difficulty is compounded by the relationship between the initiation region of one option and the subgoal region of another—changes to one option's initiation region changes another's subgoal, in turn changing its own policy and initiation region. These changes cascade to downstream options, propagating instability and causing chains of composable options to become unreliable or even break. This situation—which can happen whenever composable options are being learned simultaneously—is illustrated in Figure 2.

To ameliorate this difficulty, which we call *nonstationary subgoals*, we propose to robustify the learning process of all three option components. First, we propose a two-classifier representation of the initiation region that provides a stable subgoal target while encouraging exploration. Second, we use hindsight-experience replay to learn a generalized policy that enables us to target subgoal states that maximize the probability of being able to execute the successor option. Finally, to manage the difficulty of learning so many functions in parallel, we leverage the higher stability of model-based RL methods to learn more robust option policies.

### 3.1 Dual Initiation Classifiers to Avoid Shrinkage

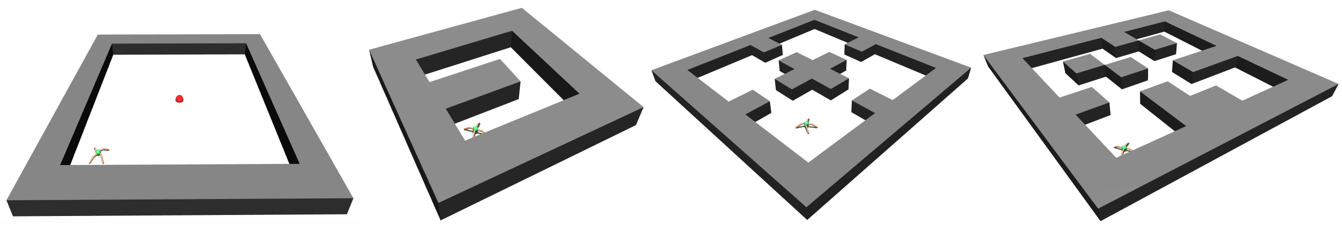
Previous approaches learned a single binary classifier to represent $\mathcal{I}_o$ for each option $o$ in the skill chain. If the option execution succeeds (i.e, the agent reaches $\beta_o$), it adds a new positive example for further refining $\mathcal{I}_o$. Since the positive example is from a region already inside the classifier, a successful execution leaves the classifier's decision boundary largely unchanged. Alternatively, if the option execution fails to reach $\beta_o$, the agent gets a negative example for the next training iteration of $\mathcal{I}_o$. Since this negative example comes from a region inside the classifier, it often shrinks the initiation region over time, with no opportunity to expand.

To avoid this issue, we propose a dual-classifier parameterization of $\mathcal{I}_o$—representing it using both optimistic and pessimistic classifiers. The pessimistic classifier represents the states from which we are highly confident that option execution will succeed, and so is a stable region for other options to target. However, if the agent could only ever choose to execute the option from inside this classifier, exploration would be hindered because the option would be prevented from expanding to new regions. To encourage exploration outside the pessimistic region, we also use an optimistic classifier to represent states where the agent can choose to execute the option. Eventually, the two classifiers should converge to approximate the "true" initiation region of the option.

Many techniques could be used to learn these two classifiers; we learn the optimistic classifier using a two-class SVM [Cortes and Vapnik, 1995] and the pessimistic classifier using a one-class SVM [Tax and Duin, 1999]. For more details, please refer to Appendix A3.

### 3.2 Robust Subgoals via Goal State Selection

We next turn to termination regions. The "chainability" of options $o_i$ and $o_j$ implies that the subgoal region $\beta_{o_i}$ is a subset of the initiation region $\mathcal{I}_{o_j}$. As $\mathcal{I}_{o_j}$ is refined over time, the subgoal region $\beta_{o_i}$ also changes, and consequently, so does the option's terminating reward. Since learning is highly sensitive to even small changes in the reward function [Packer *et al.*, 2018], this creates instability in learning $\pi_{o_i}$. DSC [Bagaria and Konidaris, 2020] mitigates this issue by freezing $\mathcal{I}_{o_j}$ after a fixed number of learning iterations, which can lead to the agent being stuck with poor estimates of $\mathcal{I}_{o_j}$ that hinder reliable skill composition.



Ant-Reacher Ant U-Maze Ant Four-Rooms Ant Large-Maze

Figure 3: Maze navigation problems used to test our algorithm. These tasks require that the agent simultaneously learn good gait policies that stabilize the "ant" robot and navigate to a distant goal in the presence of unknown obstacles.

Nevertheless, an option's subgoal region will unavoidably change continually as its target option refines its initiation classifier. To be robust to changes in $\beta_o$, we propose to make each option policy more flexible: rather than a fixed policy $\pi_o$, each option learns a goal-conditioned policy $\pi_o(s|g)$ using hindsight experience replay (HER). By conditioning the option policy $\pi_o$ on goals sampled from $\beta_o$, and postponing selecting $g$ until option execution time, the agent learns a policy robust to non-stationarity in $\beta_o$.

The goal-conditioned option policy strategy necessitates a strategy for sampling subgoal states from $\beta_o$. We propose optimizing two objectives for choosing subgoal states: (a) robustness and (b) hierarchical optimality.

**Selecting Subgoal States for Robustness**

Each option policy is rewarded for reaching its termination region; from its own perspective, all states in its termination region are equally rewarding. However, for a subsequent option $o$, some start states are better than others. How can we pick a subgoal for one option so that we increase the probability that a successive option execution will succeed?

One way is to evaluate the probability of every positive example of $o$ succeeding (by querying the probabilistic classifier representing $I_o$), but that becomes computationally very expensive as the agent gathers experience. Instead, we use the simple heuristic that, over time, the agent will learn to execute the option from reliable start states, and simply sample from the set of positive examples used to train $\mathcal{I}_o$.

**Selecting Subgoal States for Hierarchical Optimality**

The method above results in feasible trajectories that are, at best, *recursively optimal* [Barto and Mahadevan, 2003]. We would prefer to pick subgoals for each option in the skill-chain so that the overall solution trajectory is approximately *hierarchically optimal* [Barto and Mahadevan, 2003].

To select such subgoals, we first store the states $\hat{\beta}_o$ in which each option $o$ triggered $\beta_o$. Then, we use dynamic programming (DP) to distribute the value of reaching the MDP's goal $g$ to all the $\hat{\beta}_o$s along the skill chain. This results in a value table $\tilde{Q} : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}$ that can be used to pick a subgoal $s_g$ for the current option $o_t$ from state $s_t$: $s_g = \arg\max_{s \in \hat{\beta}_{o_t}} \tilde{Q}(s_t, s)$. The quality of the resulting sub-goals depends on how well $\hat{\beta}_o$ approximates $\beta_o$. If it is a perfect approximation, this DP algorithm yields the hierarchically optimal solution; otherwise, it yields a near-optimal solution. For more details, please refer to Appendix A1.

### 3.3 Learning Robust Option Policies

Finally, we consider the third component of an option: its policy. Given the number of components simultaneously being learned in hierarchical algorithms, policy learning must be highly stable for the agent to succeed. Although model-free methods are commonplace for learning option policies, model-based methods are often more stable and sample-efficient [Deisenroth and Rasmussen, 2011].

We therefore propose learning option policies using model-based RL. We follow Nagabandi *et al.* [2018] to learn a dynamics model $f_\xi$. However, Equation 1 is insufficient for good action-selection in sparse reward problems. So, we solve the following infinite-horizon optimization problem and then execute the first action [Lowrey *et al.*, 2019]:

$$\pi(s|g) = \arg\max_{a \in \mathcal{U}^{|A|}} \sum_{t=1}^{H} \gamma^{t-1} \mathcal{R}(s_t, g) + \gamma^H V_\phi(s_H|g). \quad (2)$$

We follow the same procedure as the model-free variant of our algorithm to learn the terminal value function $V_\phi$.

## 4 Experiments

Our experiments aim to answer the following questions: 1) do the proposed improvements increase the probability with which a sequence of options can be successfully composed? 2) Does the subgoal selection algorithm approximate hierarchically optimal trajectories? 3) How does the proposed algorithm compare to flat RL and other skill-discovery algorithms?

To answer these questions, we use a test-bed comprising four continuous-control maze-navigation tasks (shown in Figure 3) involving an "ant" robot simulated using MuJoCo [Todorov *et al.*, 2012; Duan *et al.*, 2016; Fu *et al.*, 2020].

**Sparse vs Dense Rewards.** Dense reward functions, although commonplace in RL, are often problematic because they demand cumbersome engineering [Yu *et al.*, 2020], can lead to sub-optimal solutions [Ng *et al.*, 1999] or reduce the problem so much that simple search might outperform RL [Mania *et al.*, 2018]. Since hierarchies are a promising way to address the challenges of sparse rewards, we evaluate all algorithms in the sparse reward setting.

### 4.1 Implementation Details

We use TD3 [Fujimoto *et al.*, 2018] and HER to learn goal-conditioned value functions in all our experiments. Transitions seen by all options were used to train a single dynamics

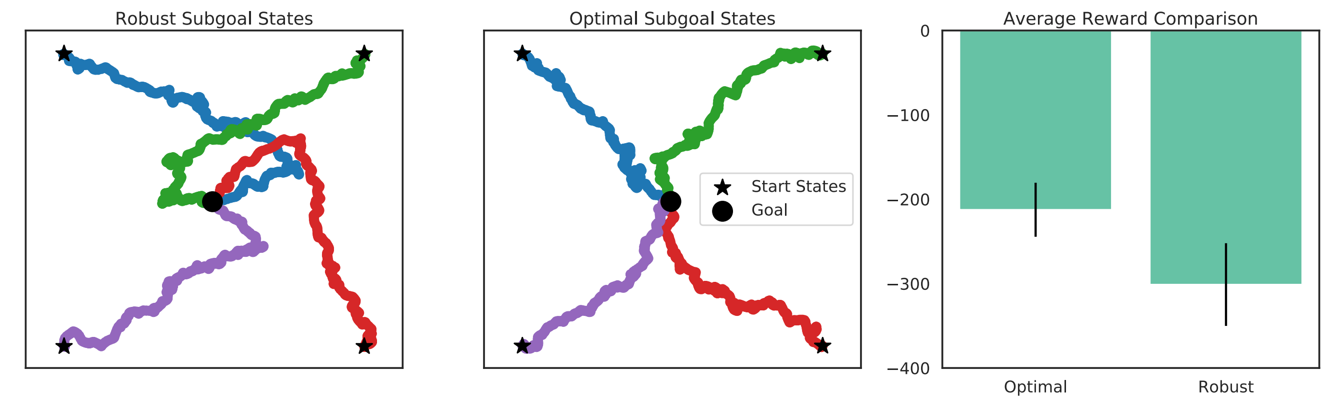



Figure 4: DSC trajectories (only the coordinates of the CoM are visualized) in Ant-Reacher when selecting subgoals *(left)* for robustness and *(middle)* using our dynamic programming algorithm. *(right)* When using DP, the agent scores better average reward (averaged over 5 runs; bars represent standard error; higher is better).

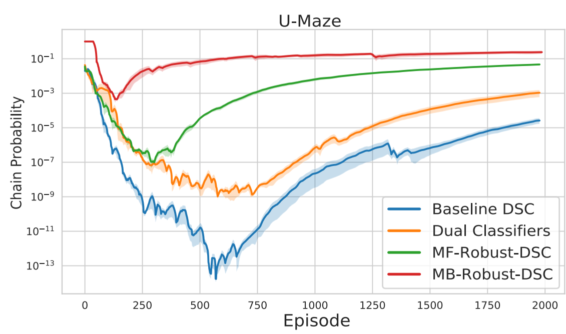



Figure 5: Comparing the robustness of skill chains discovered in the Ant U-Maze domain. For a description of the "chain probability" metric, please refer to Section 4.2. Vertical axis in log-scale; curves are averaged over 5 runs; shaded regions denote standard error.

model $f_\xi$, critic $V_\phi$, and actor $\pi_\zeta$. $V_\phi$ and $\pi_\zeta$ were used for selecting actions in the model-free case. Only $V_\phi$ from TD3 was used in the model-based case, where we approximately solved Equation 2 for action-selection. For more details, see Appendix A4.

## 4.2 Evaluating Robustness

We first evaluate whether the proposed changes increase the composability of discovered options, by measuring the robustness of skill-chains constructed in Ant U-Maze.

We measure the robustness of a sequence of options as the probability that executing each option would take the agent to a state from which it could successfully execute the next option (until it finally reaches the goal). We call this the *chaining probability*:

$$p_{chain}(o_1, ..., o_N) = \prod_{i=1}^{N} p(s_i \sim \beta_{o_{i-1}} \in \mathcal{I}_{o_i}), \quad (3)$$

where $p(s_i \sim \beta_{o_{i-1}} \in \mathcal{I}_{o_i})$ represents the open-loop probability that the agent will be able to execute $o_i$ after executing option $o_{i-1}$. We approximate each probability term by the empirical "success rate" of the option:

$$p(s_i \sim \beta_{o_{i-1}} \in \mathcal{I}_{o_i}) \approx \frac{\#successes(o_{i-1})}{\#executions(o_{i-1})},$$

where $\#successes(o_{i-1})$ is the number of times option $\pi_{o_{i-1}}$ was able to reach $\beta_{o_{i-1}}$.

We ablate our proposals by comparing the robustness of the following versions of our algorithm:

1. Baseline DSC: The deep skill chaining algorithm from Bagaria and Konidaris [2020] that we aim to improve.
2. Dual Classifiers: We add the dual classifier approach from Section 3.1 to the baseline DSC.
3. MF-Robust-DSC: We add goal-conditioned policies to (2); this is the model-free version of the algorithm from Section 3.2.
4. MB-Robust-DSC: We add model-based policies to (3); this is the version of the algorithm from Section 3.3.

**Results.** Figure 5 shows that each of our proposals successively increases the robustness of the discovered skill-chain. The shape of the curves warrants discussion: as the agent discovers new options, its chain probability, at first, decreases. This is for two reasons: (a) more terms between 0 and 1 are included in the product of Equation 3 and (b) when initialized, option policies are not strong enough to reach their subgoals. Over time, two factors are responsible for the increasing trend: (a) the agent has discovered as many options as it needs to solve the problem, at which point no more terms are added to the product and (b) each option's policy improves. Finally, the small absolute values on the vertical axis is due to the metric being an open-loop probability—we report the closed-loop success rate of the algorithm later in Section 4.4.

## 4.3 Evaluating Hierarchical Optimality

To compare the two subgoal selection strategies outlined in section 3.2, we ran the model-based variant of our algorithm on the Ant-Reacher domain. The goal state corresponds to the center of mass (CoM) of the ant being at $(0, 0)$; the starting position of the ant is sampled uniformly from $(-10, 10)$. The

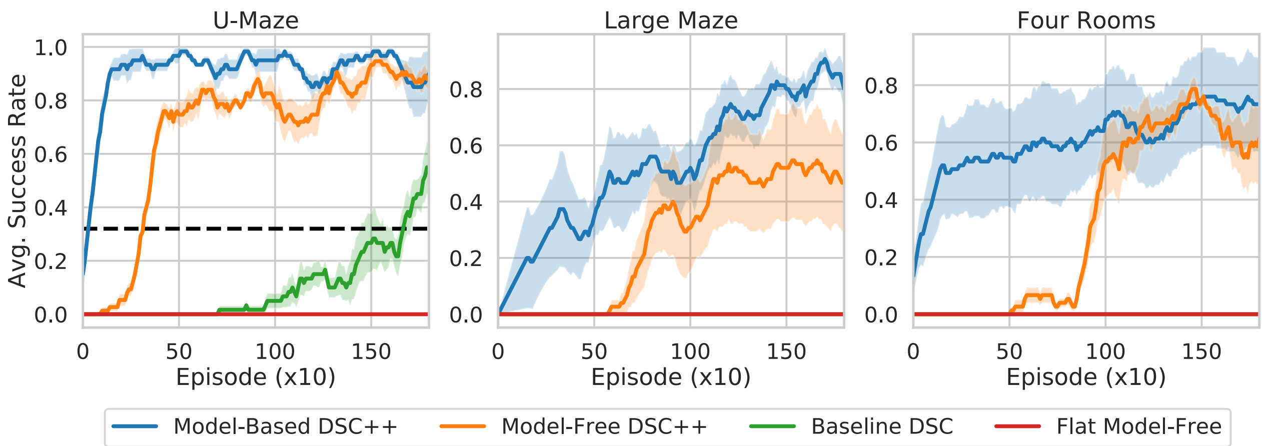



Figure 6: Average success rate in sparse-reward ant-navigation problems. Shaded regions denote standard error (averaged over 5 runs). The black dashed line in the leftmost subplot is the success rate of the flat model-free baseline after 12,000 episodes of training. In the middle and rightmost subplots, the green line is hidden below the red one.

learned initiation classifiers tended to lie in approximately concentric circles around the goal state (visualized in Figure 1 of the Appendix). After both algorithm variants were trained for 1000 episodes, they were evaluated on how many steps it would take them to reach the goal when starting in the four corners of the domain $\{(\pm 9, \pm 9)\}$.

**Results.** Robust subgoal selection often yielded subgoal states on the "other side" of the goal—if the robot was on the bottom-left of the domain, it could pick a subgoal closer to the top-right. This led to the sub-optimal trajectories shown in Figure 4a, where the ant would often over-shoot the goal. By contrast, the dynamic programming procedure picked subgoal states that were always between the agent's current state and the overall goal—leading to much more sensible, and approximately hierarchically optimal, solution trajectories (Figure 4b). Figure 4c shows that the agent using our DP algorithm for picking subgoals collected higher average rewards, further suggesting smoother overall trajectories to the goal.

### 4.4 Learning Curves

Our final experiment compares the following algorithms on ant U-maze, large-maze and four-rooms:

1. Flat model-free baseline: we used TD3+HER [Fujimoto *et al.*, 2018; Andrychowicz *et al.*, 2017] as a flat model-free baseline, since it is a state-of-the-art method for continuous control (we used the same algorithm to learn our model-free option policies).
2. Flat finite-horizon model-based baseline: we used the model-based algorithm from Nagabandi *et al.* [2018] as our flat model-based baseline (we used the same algorithm to learn our model-based option policies).
3. Flat infinite-horizon model-based baseline: given that our problems are sparse-reward, we augment the model-based baseline with TD3, which is used to learn a value function that informs action-selection (Equation 2).
4. Deep skill chaining (DSC): DSC is our HRL baseline, because we extend DSC and it outperformed other skill-discovery methods [Bagaria and Konidaris, 2020].
5. Model-Free DSC++ (ours): the model-free variant of our algorithm described in Section 3.2.
6. Model-based DSC++ (ours): the model-based variant of our algorithm described in Section 3.3.

We report the "average success rate" metric from Andrychowicz *et al.* [2017]. Every 10 episodes, we ran the algorithm from a fixed start position $(0, 0)$, checking if it could reach the goal within 1000 steps. Due to its simplicity, we use the robust subgoal selection algorithm from Section 3.3 for rolling out our goal-conditioned option policies.

**Results.** Figure 6 shows the average success rate of competing methods averaged over 5 random seeds. Both versions of the flat model-based baseline were unable to achieve a $> 0\%$ success rate; so we leave them out of the learning curves in Figure 6. Ant U-Maze was the only sparse-reward problem that TD3 and the baseline DSC were able to achieve a $> 0\%$ (but TD3 had to be trained for far more episodes). The proposed algorithm (shown with blue and orange curves) were easily able to outperform all baselines including DSC.

## 5 Conclusion

The success of skill discovery in goal-directed tasks depends on learning options that can be sequentially composed. However, robust composition is challenging because of non-stationary subgoal regions. We proposed methods that address all three components of option learning—initiation regions, termination conditions, and option policies—to learn reliably composable options. We experimentally showed that our augmentations measurably improve the robustness of discovered skills and eventually allow us to solve more challenging, long-horizon problems.

## Acknowledgements

We thank Kshitij Sachan, Cam Allen, Sam Lobel, and other members of the Brown BigAI group for their suggestions. This research was supported by NSF grants 1955361, 1717569 and the DARPA Lifelong Learning Machines program under grant FA8750-18-2-0117. This research was conducted using computational resources and services at the Center for Computation and Visualization, Brown University.

## References

[Abel, 2020] David Abel. *A Theory of Abstraction in Reinforcement Learning*. PhD thesis, Brown University, 2020.

[Andrychowicz *et al.*, 2017] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[Asadi *et al.*, 2019] Kavosh Asadi, Dipendra Misra, Seungchan Kim, and Michel L Littman. Combating the compounding-error problem with a multi-step model. *arXiv preprint arXiv:1905.13320*, 2019.

[Bacon *et al.*, 2017] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[Bagaria and Konidaris, 2020] Akhil Bagaria and George Konidaris. Option discovery using deep skill chaining. In *International Conference on Learning Representations*, 2020.

[Barreto *et al.*, 2019] André Barreto, Diana Borsa, Shaobo Hou, Gheorghe Comanici, Eser Aygün, Philippe Hamel, Daniel Toyama, Shibl Mourad, David Silver, and Doina Precup. The option keyboard: Combining skills in reinforcement learning. In *Advances in Neural Information Processing Systems*, 2019.

[Barto and Mahadevan, 2003] Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, pages 41–77, 2003.

[Baumli *et al.*, 2020] Kate Baumli, David Warde-Farley, Steven Hansen, and Volodymyr Mnih. Relative variational intrinsic control. *arXiv preprint arXiv:2012.07827*, 2020.

[Botvinick *et al.*, 2009] Matthew M Botvinick, Yael Niv, and Andew G Barto. Hierarchically organized behavior and its neural foundations: a reinforcement learning perspective. *Cognition*, 113(3), 2009.

[Burridge *et al.*, 1999] Robert R Burridge, Alfred A Rizzi, and Daniel E Koditschek. Sequential composition of dynamically dexterous robot behaviors. *The International Journal of Robotics Research*, pages 534–555, 1999.

[Campos Camúñez *et al.*, 2020] Víctor Campos Camúñez, Alex Trott, Caiming Xiong, Richard Socher, Xavier Giró Nieto, and Jordi Torres Viñals. Explore, discover and learn: unsupervised discovery of state-covering skills. In *International Conference on Machine Learning*, pages 1317–1327, 2020.

[Cortes and Vapnik, 1995] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 1995.

[Deisenroth and Rasmussen, 2011] Marc Deisenroth and Carl E Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning*, pages 465–472, 2011.

[Dietterich, 2000] Thomas G Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Aritifical Intelligence Research*, pages 227–303, 2000.

[Duan *et al.*, 2016] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338, 2016.

[Eysenbach *et al.*, 2019] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019.

[Frans *et al.*, 2018] Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. Meta learning shared hierarchies. In *International Conference on Learning Representations*, 2018.

[Fu *et al.*, 2020] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[Fujimoto *et al.*, 2018] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1582–1591, 2018.

[Garcia *et al.*, 1989] Carlos E Garcia, David M Prett, and Manfred Morari. Model predictive control: theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.

[Gregor *et al.*, 2016] Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *ArXiv*, abs/1611.07507, 2016.

[Jain *et al.*, 2021] Arushi Jain, Khimya Khetarpal, and Doina Precup. Safe option-critic: learning safety in the option-critic architecture. *The Knowledge Engineering Review*, 36, 2021.

[Jinnai *et al.*, 2019] Yuu Jinnai, Jee Won Park, David Abel, and George Konidaris. Discovering options for exploration by minimizing cover time. In *International Conference on Machine Learning*, pages 3130–3139, 2019.

[Kaelbling and Lozano-Pérez, 2017] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Learning composable models of parameterized skills. In *IEEE International Conference on Robotics and Automation*, pages 886–893, 2017.

[Kaelbling, 1993] Leslie Pack Kaelbling. Learning to achieve goals. In *International Joint Conference on Artificial Intelligence*, pages 1094–1099, 1993.

[Khetarpal and Precup, 2019] Khimya Khetarpal and Doina Precup. Learning options with interest functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 9955–9956, 2019.

[Konidaris and Barto, 2007] George Konidaris and Andrew Barto. Building portable options: Skill transfer in reinforcement learning. In *International Joint Conference on Artificial Intelligence*, pages 895–900, 2007.

[Konidaris and Barto, 2009] George Konidaris and Andrew Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. *Advances in Neural Information Processing Systems*, 22, 2009.

[Konidaris *et al.*, 2018] George Konidaris, Leslie Pack Kaelbling, and Tomas Lozano-Perez. From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 61:215–289, 2018.

[Levy *et al.*, 2019] Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Hierarchical reinforcement learning with hindsight. In *International Conference on Learning Representations*, 2019.

[Lillicrap *et al.*, 2015] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[Lowrey *et al.*, 2019] Kendall Lowrey, Aravind Rajeswaran, Sham Kakade, Emanuel Todorov, and Igor Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control. In *International Conference on Learning Representations*, 2019.

[Lozano-Perez *et al.*, 1984] Tomas Lozano-Perez, Matthew T Mason, and Russell H Taylor. Automatic synthesis of fine-motion strategies for robots. *The International Journal of Robotics Research*, 1984.

[Lu *et al.*, 2021] Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Reset-free lifelong learning with skill-space planning. In *International Conference on Learning Representations*, 2021.

[Lyapunov, 1992] Aleksandr Mikhailovich Lyapunov. The general problem of the stability of motion. *International journal of control*, 1992.

[Mania *et al.*, 2018] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search of static linear policies is competitive for reinforcement learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 1805–1814, 2018.

[Mankowitz *et al.*, 2018] Daniel Mankowitz, Timothy Mann, Pierre-Luc Bacon, Doina Precup, and Shie Mannor. Learning robust options. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, pages 6409–6416, 2018.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, pages 529–533, 2015.

[Nagabandi *et al.*, 2018] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566. IEEE, 2018.

[Ng *et al.*, 1999] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*, volume 99, pages 278–287, 1999.

[Packer *et al.*, 2018] Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing generalization in deep reinforcement learning. *arXiv preprint arXiv:1810.12282*, 2018.

[Qureshi *et al.*, 2020] Ahmed H Qureshi, Jacob J Johnson, Yuzhe Qin, Taylor Henderson, Byron Boots, and Michael C Yip. Composing task-agnostic policies with deep reinforcement learning. *International Conference on Learning Representations*, 2020.

[Randløv *et al.*, 2000] Jette Randløv, Andrew G Barto, and Michael T Rosenstein. Combining reinforcement learning with a local control algorithm. In *International Conference on Machine Learning*, pages 775–782, 2000.

[Schaul *et al.*, 2015] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International Conference on Machine Learning*, pages 1312–1320, 2015.

[Sharma *et al.*, 2020] Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations*, 2020.

[Shoeleh and Asadpour, 2017] Farzaneh Shoeleh and Masoud Asadpour. Graph based skill acquisition and transfer learning for continuous reinforcement learning domains. *Pattern Recognition Letters*, pages 104–116, 2017.

[Singh *et al.*, 2020] Avi Singh, Albert Yu, Jonathan Yang, Jesse Zhang, Aviral Kumar, and Sergey Levine. COG: Connecting new skills to past experience with offline reinforcement learning. In *Conference on Robot Learning*, 2020.

[Sutton and Barto, 2018] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[Sutton *et al.*, 1999] R.S. Sutton, , D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, pages 181–211, 1999.

[Tax and Duin, 1999] David MJ Tax and Robert PW Duin. Support vector domain description. *Pattern recognition letters*, 1999.

[Tedrake, 2009] Russ Tedrake. LQR-trees: Feedback motion planning on sparse randomized trees. In *Robotics: Science and Systems*, 2009.

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

[Todorov *et al.*, 2012] 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, 2012.

[Watkins and Dayan, 1992] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, pages 279–292, 1992.

[Yu *et al.*, 2020] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, volume 100, pages 1094–1100, 2020.