Skill Discovery for Exploration and Planning using Deep Skill Graphs

Akhil Bagaria [1] **Jason Crowley** [1] **Jing Wei Nicholas Lim** [1] **George Konidaris** [1]

Abstract

Temporal abstraction provides an opportunity to drastically lower the decision making burden facing reinforcement learning agents in rich sensorimotor spaces. Well constructed hierarchies induce state and action abstractions that can reduce large continuous MDPs into small discrete ones, in which planning with a learned model is feasible. We propose a novel algorithm, *Deep Skill Graphs*, for acquiring such a minimal representation of an environment. Our algorithm seamlessly interleaves discovering skills and planning using them to gain unsupervised mastery over ever increasing portions of the state-space. The constructed skill graph can be used to drive the agent to novel goals at test time, requiring little-to-no additional learning. We test our algorithm on a series of continuous control tasks where it outperforms baseline flat and hierarchical RL methods alike.

1. Introduction

Model-free reinforcement learning agents can acquire goal-directed behaviors in unknown environments (Sutton & Barto, 2018; Mnih et al., 2015; Lillicrap et al., 2015). These behaviors however, often tend to be reactive and must be reacquired when the agent is tasked with solving a different, albeit related task (Farebrother et al., 2018; Witty et al., 2018). Planning, on the other hand, can generate long-horizon behaviors (Campbell et al., 2002; Silver et al., 2017) but assumes access to a model of the environment, which can often be hard to specify. Our goal is to learn representations that enable an agent to transition from trial-and-error learning to solving long-horizon tasks using high-level planning.

One way to generate long-horizon behaviors is to plan using models learned from raw observations (Sutton, 1991; Levine et al., 2016; Ha & Schmidhuber, 2018; Hafner et al., 2018; Kaiser et al., 2020). However, model-based methods are plagued by the difficulty of learning effective models in high-dimensional spaces. Learning abstractions (Giunchiglia & Walsh, 1992; Konidaris, 2019; Abel & Littman, 2020) afforded by the hierarchical reinforcement learning (HRL) framework (Barto & Mahadevan, 2003) could facilitate planning without having to confront the problem of learning a detailed model of the world. The majority of research in HRL has examined the benefits of discovering and using abstract actions, or *skills* (Sutton et al., 1999; Mcgovern & Barto, 2002; Precup, 2001; Konidaris & Barto, 2011; Ravindran & Barto, 2004; Bacon, 2018). Less studied is the complimentary question of how discovered skills can inform useful state-abstraction.

We build on the key insight of Konidaris et al. (2018) that abstract actions create an opportunity to build an accompanying state abstraction that supports planning. They proved that to compress an SMDP (Sutton et al., 1999) into a discrete representation suitable for planning, the agent's skills must be sequentially executable—successful execution of one should permit the execution of another. Furthermore, Bagaria & Konidaris (2020) recently showed how skills that have this property can be autonomously discovered in high-dimensional continuous spaces. However, for skill discovery to proceed in their work, Bagaria & Konidaris (2020) assume access to a pre-specified goal state and an exploration policy that can generate a small number of trajectories that reach that goal state.

Such extrinsically defined goals are often unavailable, and even when they are, random exploration may be insufficient for the agent to ever reach them. So that skill-acquisition may proceed in the absence of such extrinsic goals, we discover them using principles of intrinsic motivation (Barto et al., 2004; Chentanez et al., 2005). We call such discovered target states *salient events*, and construct a collection of skills capable of moving the agent between such salient events, resulting in a *skill graph*. This graph is a compressed discrete representation of the original MDP where vertices correspond to states in which skills operate and edges correspond to skill policies. Because the skill graph was constructed to meet the specifications laid out by Konidaris et al. (2018), it is provably sound—meaning that plans constructed in the abstract MDP correspond to

[1]Department of Computer Science, Brown University, Providence, Rhode Island, USA. Correspondence to: Akhil Bagaria <akhil_bagaria@brown.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 108, 2020.

feasible solutions in the original ground MDP.

We test our algorithm on three challenging maze-navigation tasks in MuJoCo (Todorov et al., 2012; Fu et al., 2020), demonstrate its key properties, and find that it can significantly out-performs baseline flat and hierarchical RL methods alike.

2. Background

We consider the class of MDPs $\mathcal{M} = (S, A, \mathcal{R}, \mathcal{T}, \gamma, \rho)$. S denotes the state space, A denotes the action space and γ is the discount factor (Sutton & Barto, 2018). $\mathcal{R}$ is the set of reward functions and ρ is the set of start state distributions implied by the different tasks in the multi-task setting (Sutton et al., 2007; Tanaka & Yamamura, 2003; Brunskill & Li, 2013; Wilson et al., 2007). The transition dynamics $\mathcal{T}$ is assumed to be fixed among the different tasks.

Importantly, we do not assume access to $\mathcal{R}$ during training time—skill discovery proceeds without extrinsic rewards. Furthermore, ρ remains fixed at training time, but can change arbitrarily at test-time. This ensures that the agent cannot "teleport" to states that might otherwise be difficult to reach and thus must confront the exploration problem in its entirety (Even-Dar et al., 2005; Kakade & Langford, 2002).

2.1. The Options Framework

We model abstract actions (or skills) as *options* (Sutton et al., 1999). Each option o in the agent's option repertoire $\mathcal{O}$ is defined as a three element tuple $(\mathcal{I}_o, \pi_o, \beta_o)$. The initiation set $\mathcal{I}_o : s \rightarrow \{0, 1\}$ describes the set of states from which option o can be executed. The termination set $\beta_o : s \rightarrow \{0, 1\}$ describes the set of states in which option execution is deemed successful. Finally, the option policy $\pi_o : s \rightarrow a$ is a closed-loop controller that drives the agent from states in $\mathcal{I}_o$ to those in β_o. Augmenting the set of available actions with options results in a Semi-Markov Decision Process (SMDP) (Sutton et al., 1999) where the next state depends on the current state, action, and *time*.

In addition to the three elements described above, Konidaris et al. (2018) define the *effect set* $\mathcal{E}_o$ of option o as the set of states in which the agent might find itself after successfully executing π_o from anywhere in $\mathcal{I}_o$. Note that $\mathcal{E}_o \subseteq \beta_o$ since there may be states in β_o not reachable from states in $\mathcal{I}_o$.

2.2. Skill Chaining

The skill chaining algorithm (Konidaris & Barto, 2009; Bagaria & Konidaris, 2020) incrementally constructs options that extend backward from a goal state to the start states of the MDP. Discovered skills have the property that the initiation condition of an option o_i is the termination condition of the option o_{i-1} that precedes it in its chain, i.e, $\beta_{o_{i-1}} = \mathcal{I}_{o_i}$. Option policies in the skill chaining framework have local support: they specialize in different regions of the state-space and do not have to learn representations for states far outside their initiation region. This allows them to learn effective solutions to their own sub-problems which are then combined by a policy over options.

Each skill learns to initiate only from those states from which it can reliably solve its own sub-problem. As a result, the skill chaining algorithm adaptively discovers as many skills as it needs, with whatever granularity it needs, to most reliably get from its start state to its goal state. By deconstructing the solution to goal-oriented MDPs into sequential option executions, deep skill chaining (DSC) can learn overall policies that reach the goal far more effectively than flat reinforcement learning techniques.

Given a single start state and a goal state, skill chaining organizes skills in the form of a chain. When there are multiple start states, skill chaining organizes skills in the form of a tree rooted at the goal state. While effective in solving many goal-oriented MDPs, this topology is insufficient in the multi-task setting where the agent could start anywhere and be asked to go anywhere. Consequently, we propose to organize skills in the form of a graph to handle the multi-task setting.

2.3. Covering Options

Recent work has focused on learning task agnostic representations by exploiting the spectral properties of the environment's transition dynamics (Mahadevan & Maggioni, 2007; Machado et al., 2017; Liu et al., 2017; Machado et al., 2018). Most recently, Jinnai et al. noticed that the eigenvector corresponding to the second smallest eigenvalue of the graph Laplacian (Chung & Graham, 1997) (also known as the Fiedler vector (Fiedler, 1973)) captures information about states that are furthest apart in the topology of an MDP. In an effort to make the MDP easier to explore, they developed Covering Options (Jinnai et al.) to connect the two states in a discrete MDP that are furthest apart according to this Fiedler vector. Deep covering options (DCO) (Jinnai et al., 2020) used Wu et al. (2019)'s approximation of the graph Laplacian to extend covering options to large continuous spaces. Because of their ability to identify states in the frontier and their demonstrated superiority to other methods that discover options for exploration (Eysenbach et al., 2019a; Machado et al., 2017), we use it to boost exploration in our algorithm.

2.4. Rapidly Exploring Random Trees (RRT)

RRT is a motion planning algorithm used to find a collision-free path between two points in a robot's configuration space (LaValle, 1998). The RRT algorithm, which builds a tree

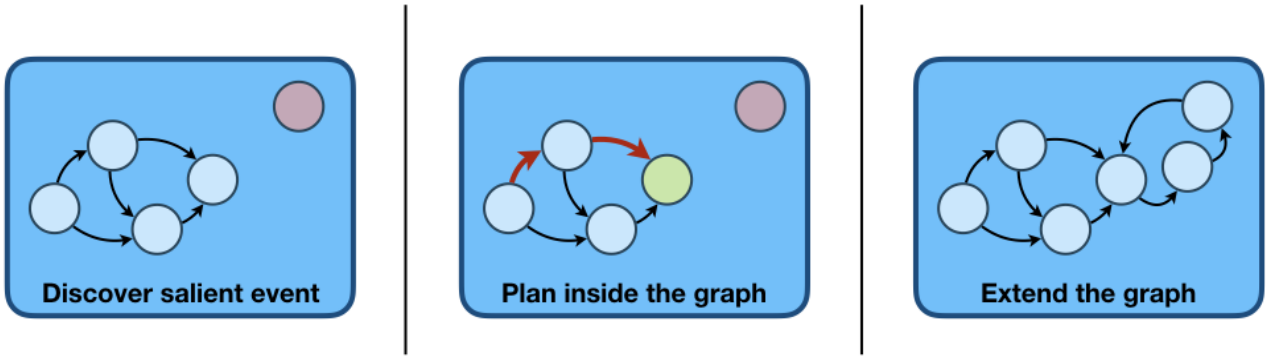

Figure 1. When the discovered salient event (red) is outside the graph, the agent uses planning inside the graph to reach the node closest to its goal (green). It then expands the graph by constructing a series of skills that connect the salient event to the graph.

rooted at the start state, proceeds in three steps: (a) randomly sample a state from the configuration space, (b) identify its nearest neighbor node in the tree and finally (c) use the robot's dynamics model to move one step in the direction of the sampled state. Despite its simplicity, RRTs have been wildly successful in solving motion planning problems involving high-dimensional dynamical systems. It owes its success in large part to the *Voronoi bias* of the tree construction procedure (Lindemann & LaValle, 2004).

The Voronoi bias of RRT refers to property that at each iteration of the algorithm, the direction in which tree expands is proportional to the volume of the Voronoi region of an edge node. The volume of the Voronoi region of a state can thus be seen as a measure of "how unexplored" the region around that state is; randomly selecting points to expand the tree therefore causes it to naturally grow in the direction of the unexplored frontiers of the state-space.

Since RRT assumes a given dynamics model, it cannot be directly used in the RL setting. However, as we will see in Section 3, our graph construction procedure is heavily inspired by RRT. Finally, although inspired by RRT, we do not learn a single-query tree (LaValle, 2006). Instead, much like the Probabilistic Road Map (PRM) algorithm (Kavraki et al., 1996), we learn a multiple-query graph, meaning that the discovered skill graph can be used to find solution paths between multiple start-goal pairs.

3. Deep Skill Graphs

A skill graph is a particular arrangement of acquired skills that represents a convenient discrete abstraction of the original continuous MDP. To construct this graph, the agent continually interleaves two steps: (a) discovery of target salient regions that will increase the coverage of the skill graph and (b) creation of skill chains that connect each new salient event to the graph. Once acquired, the agent may construct plans to reach any state that lies in the region spanned by its skill graph.

3.1. Definitions

Salient Events: A salient event $\beta_{salient}$ is a collection of states that are deemed important by an intrinsic motivation system (Barto et al., 2004). We formalize them as binary classifiers that take a state as input and return a boolean decision denoting whether that state triggered that salient event.

Skill Graph: A skill graph is a weighted and directed graph $G = (\mathcal{V}, E, \mathcal{W})$. Each vertex $i \in \mathcal{V}$ either corresponds to an option effect set or a salient event. There is an edge $e_{i \to j}$ between vertices i and j if and only if the effect set of vertex i is inside the initiation set of vertex j, i.e, $\mathcal{E}_i \subseteq \mathcal{I}_j$. The edge weight $w_{i,j} \in \mathcal{W}$ is the inverse of the fraction of times you can successfully get from vertex i to vertex j using a single option execution.

3.2. Constructing the Skill Graph

As illustrated in Figure 1, skill graph construction iteratively proceeds in three steps. In the first step, the DSG agent discovers a target/salient event it would like to reach. Since this target state lies outside the graph, the agent does not yet have the necessary skills to get there using planning. In the second step, the agent identifies the node in the graph closest to the target state identified in step 1. It then constructs a plan (shown with red arrows) inside the graph to reach this identified node (shown in green). Once there, it uses deep skill chaining to construct a series of skills that eventually connect the target state to graph. Now, the salient event from step 1 is now just another node in the graph that can be easily reached using planning. This process repeats when a new salient event is generated, until the skill-graph has achieved reasonable coverage of the state-space.

3.2.1. Salient Event Discovery

Task-agnostic skill acquisition requires the agent to identify salient regions of the state-space and construct skills that target such regions. Regions of the state-space covered by the graph represent regions where the agent has achieved mastery (Kaelbling, 1993; Veeriah et al., 2018). This is because the agent can plan using learned skills to reliably reach arbitrary states inside the graph. So that the agent may incrementally increase the portion of the world over which it has achieved mastery, our salient event discovery algorithm should extend our graph into largely unexplored regions of the state-space.

We propose two strategies for generating salient events:

Deep covering options (DCO): Given a buffer of n transitions $\mathcal{B} = (s, s')_{1:n}$, DCO learns a one-dimensional embedding called the Fiedler vector $f : s \to \mathbb{R}$. The states $s_1 = \arg\max_{s \in \mathcal{B}} f(s)$ and $s_2 = \arg\min_{s \in \mathcal{B}} f(s)$ represent the two states in $\mathcal{B}$ that are furthest apart in the agent's decision space. Jinnai et al. (2020) note that adding an edge between s_1, s_2 maximally increases the connectivity of the graph (makes it easier to explore the underlying MDP). They assume that such an edge corresponds to a single option execution. However, in general s_1, s_2 might be very far away from each other and hence require an arbitrary number of skills to reliably go from one to the other.

Algorithmically, DCO examines the replay buffer of the agent every K episodes (where K is a hyperparameter). It then generates two salient events: $\beta_{s_1} : s \to \{0, 1\}$ that targets state s_1 and $\beta_{s_2} : s \to \{0, 1\}$ that targets state s_2. We then call upon deep skill chaining to learn as many skills as it needs to connect β_{s_1} and β_{s_2} to the existing portion of the skill graph. Once these events are in the graph, the agent can then plan with its existing skills to get from one to the other—reflecting the increased connectivity of the resulting underlying SMDP.

Random sampling: Covering options is a principled approach to generating salient events for intrinsically motivated graph expansion. Its success in high-dimensional spaces however, is dependent on the quality of its approximation of the graph Laplacian (Wu et al., 2019). A far simpler alternative would be to generate salient events in the same way the RRT algorithm does—uniformly at random. This simple strategy can lead to the graph growing in the directions of the largest unexplored regions of the state-space. Its disadvantage, however, is that it assumes that one can sample feasible states from the environment. When such an oracle is available, random sampling of salient events may provide a competitive alternative to its more computationally expensive counterparts.

3.2.2. Picking a target salient event

Given the current list of salient events $\tilde{\beta} = \{\beta_1, \beta_2, ..., \beta_n\}$, the agent must first pick which one to target in the current episode. One could imagine using heuristics based on which which skills to practice and refine the most (Stout & Barto, 2010), but for simplicity, we pick a target randomly from this list. Once the agent successfully reaches β, it picks a new target from $\tilde{\beta}$, and the process continues.

3.2.3. Identifying the nearest node in the graph

Given a target salient event β that lies outside the graph, DSG must first identify its nearest neighbor in the graph. Let this vertex in the graph be v_{nn}. For the experiments presented in this paper, we chose the node with the lowest Euclidean norm. This measure is not applicable for all problems, but we leave discovering and using a more appropriate measure related to MDPs (Mahadevan & Maggioni, 2007; Taylor et al., 2011) for future work.

3.2.4. Acting inside the graph

Given target salient event β and its nearest neighbor inside the graph v_{nn}, the DSG agent must now figure out a way to get to that target node v_{nn} from its current state s_t. We can find the vertex in the graph corresponding to s_t by querying the initiation set classifiers of all the options in the graph. Let o_t be the option whose initiation set s_t is in, i.e, $\mathcal{I}_{o_t}(s_t) = 1, \exists o_t \in \mathcal{O}$. We now use graph search (Dijkstra, 1959) to find a plan that goes from o_t to v_{nn}. We then execute the first option in the plan, land in a new state $s_{t+\tau}$, and then re-plan to go from $s_{t+\tau}$ to v_{nn}. This process continues until we reach v_{nn}.

3.2.5. Extending the graph

Once DSG has used its planning based control loop to reach v_{nn}, which is usually a vertex in the boundary of the graph, it leaves the graph in an effort to reach the target event β. Here, it uses deep skill chaining to learn options that eventually chain back from β to v_{nn}. After that, the agent can go back to using its planner to reach β in the future.

3.2.6. Adding edges to the graph

Now we will describe how skill chains targeting different salient events may be connected together in the skill graph. For simplicity, we can break our discussion into two different cases:

Option-option edges: For plans in the abstract MDP to correspond to feasible solutions in the ground MDP, Konidaris et al. (2018) showed that any two options o_1 and o_2 can have an edge $e_{1,2}$ between them if and only if there is a guarantee that successful execution of o_1 will allow the agent to execute o_2, i.e, $e_{1,2}$ exists iff $\mathcal{E}_{o_1} \subseteq \mathcal{I}_{o_2}$. To implement this rule,

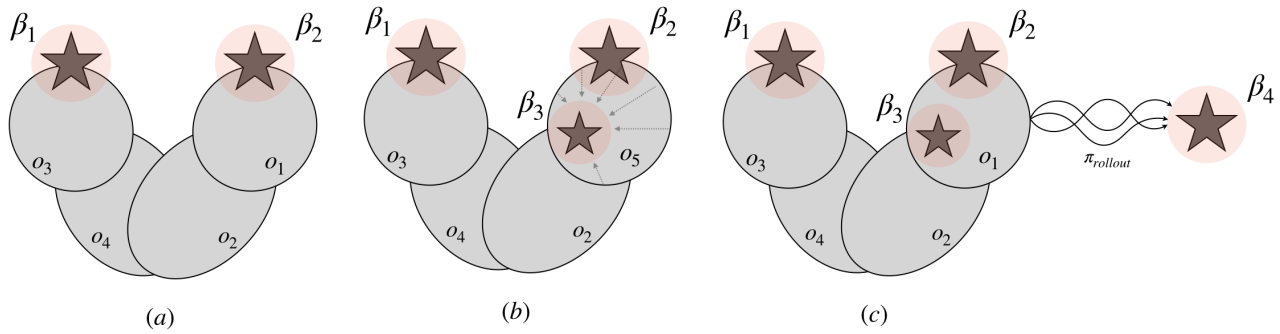

Figure 2. Illustration of how DSG handles achieving goal states at test time: (a) If the goal region contains the effect set of any option, DSG can use its planner to reach it without any additional learning (b) If the goal lies inside the initiation set of an option, it can create a new option o_5 to reach the goal with some learning and (c) If the goal happens to fall outside the graph, DSG uses planning to reach the nearest neighbor in the graph and does RL (deep skill chaining) for the rest of the way.

we store all the states in which o_1 execution successfully terminated and check if all of them lie inside $\mathcal{I}_{o_2}$. If at some point $\mathcal{E}_{o_1}$ expands and we get a sample from its effect set that lies outside $\mathcal{I}_{o_2}$, we must delete edge $e_{1,2}$ from the skill graph.

Option-event edges: An edge can exist from an option to a salient event if the option's effect set is a subset of the salient event—this typically only happens when the option is trained to target that salient event. For an edge to go from a salient event $\beta_{salient}$ to option o, the option's initiation set must contain all the points at which that salient event can be triggered, i.e, $\beta_{salient} \subseteq \mathcal{I}_o$. To implement this logic, we keep track of all the points at which any option has ever triggered β and ensure that $\mathcal{I}_o$ includes all of those points.

To develop greater intuition about when it is appropriate to connect options in the graph, readers may refer to figures 7 and 8 in the appendix.

3.2.7. Learning backward options

So far, we have only described how the skills in the graph take the agent from the start state to discovered salient events. However, to equip the agent with the ability to navigate *between* events, it must learn options that go backward from discovered salient events to the start state (and to other salient events). The logic for learning these "backward options" is similar to learning "forward options", but requires some additional care which is described in section A.3 of the appendix.

3.3. Using the Skill Graph

Once the agent has finished constructing the skill graph, it can use it to target goals not seen during training. Suppose that at test time, the agent is asked to reach a goal region β_{s_g} (in continuous domains, this is typically defined as a small region around a goal state s_g). As illustrated in figure 2, the goal region β_{s_g}, at test time can fall into one of the three possible scenarios:

- β_{s_g} completely contains the effect set of an option: Since reaching that effect set would imply reaching β_{s_g}, the agent may use its planning based control loop to reach s_g without any additional learning.
- β_{s_g} is inside the initiation set of some option in the graph: Suppose that the β_{s_g} does not contain the effect set of an option in the graph but is inside initiation set of option o_{train}. In this case, we have options that can take the agent *close to* the goal but do not have a controller that will drive us to s_g. As a result, we create a new option o_{test} having the same initiation set as option o_{train}. The agent then uses planning to get to $\mathcal{I}_{o_{test}}$ and then execute $\pi_{o_{test}}$ to reach s_g. Since existing skills reliably bring the agent close to s_g, it can learn $\pi_{o_{test}}$ fairly quickly [1].
- Goal is outside the graph: It is of course possible that s_g entirely lies outside our graph. In this case, DSG uses the same logic that it used to trigger salient events outside the graph during training, i.e, it follows the steps outlined in sections 3.2.3 – 3.2.6.

4. Experiments

We test our algorithm in a series of continuous control tasks adapted from the "Dataset for RL" benchmark (Fu et al., 2020; Todorov et al., 2012).[2] Chosen tasks either involve

[1] It is possible to pre-train $\pi_{o_{test}}$ off-policy with o_{train}'s replay buffer (Munos et al., 2016; Thomas & Brunskill, 2016), but for simplicity we train this policy starting from a random initialization.

[2] We use the maze-navigation tasks from this suite, without using their demonstration data.

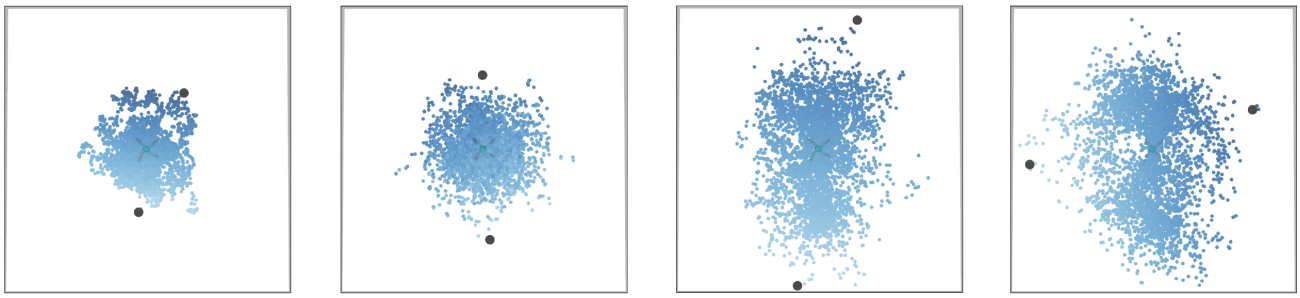

Figure 3. Identifying and expanding the frontier in the Ant Reacher domain: these plots visualize the position of the ant-robot at different times during training. The shade of the blue dots represent the values of the approximated Fiedler vector at that state. The black dots represent the two states that DCO identifies as being farthest away in the agent's transition buffer. The fact that the black dots always lie at the edge of the explored space shows DSG's ability to consistently *identify* the frontier. The fact that the area occupied by the blue dots is always increasing suggests DSG's ability to consistently *expand* that frontier.

the point agent from Nachum et al. (2018) or the quadruped ant robot from Brockman et al. (2016). These tasks are challenging for non-hierarchical methods, which make little-to-no learning progress in these tasks (Duan et al., 2016).

We evaluate two versions of the proposed algorithm: one using deep covering options (**DSG-DCO**) and the other using RRT-style random sampling for salient event discovery (**DSG-RRT**). We compare Deep Skill Graphs to goal-conditioned RL (Kaelbling, 1993; Schaul et al., 2015) and deep covering options (Jinnai et al., 2020).

Goal-conditioned RL algorithms learn policies that generalize across goals by taking the goal state as an additional input during training. Because of their relative simplicity and their widespread use in RL (Nachum et al., 2018; Levy et al., 2019; Eysenbach et al., 2019b; Nasiriany et al., 2019), we compare against them as a representative non-hierarchical method that works in the multi-task setting.

We choose DCO as our HRL baseline because (a) the DSG-DCO variant of our algorithm uses DCO for salience discovery and (b) DCO outperformed other methods that learn options for exploration (Eysenbach et al., 2019a; Machado et al., 2017).

4.1. Qualitative Evaluation

Exploration Property of DSG: We compare the states visited by the DSG agent and those visited under a random walk. Figure 4 shows that in the ant-maze environment, skill discovery can lead to temporally extended exploration as opposed to the dithering behavior of random walks, even in the absence of an extrinsic reward function.

Incremental Graph Expansion: Figures 3 and 6 provides some intuition on why DSG can effectively explore large regions of the state-space—the skill-graph begins at the start state and incrementally expands into unexplored regions of the state-space. By planning and executing learned skills

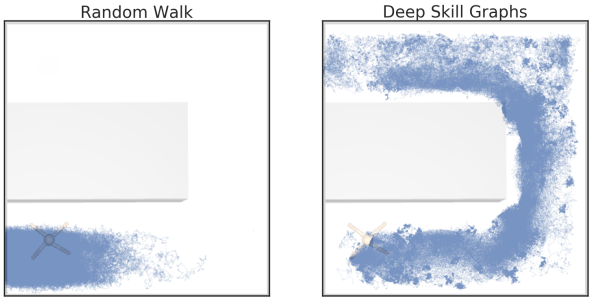

Figure 4. Exploration property of skill-graphs: while a random walk fails to visit the different parts of the maze, deep skill graphs incrementally expand the frontier of the agent's experiences.

inside the graph, the agent can reliably get back to the frontier of its knowledge (as represented by the outermost vertices in the graph). Exploration from the frontier in turn allows it to amass the experiences it needs to further expand the graph. By interleaving planning and chaining in this way, the DSG agent incrementally achieves mastery of ever increasing proportions of the state-space.

4.2. Quantitative Evaluation

After training each algorithm for the same number of episodes, we evaluate their ability to navigate from randomly chosen start states to randomly chosen goal states. As is common in the multi-task setting, we plot the "success rate" of each algorithm, which is the fraction of times it was able to successfully reach the goal in the given episodic budget. All learning curves in figure 5 are averaged over the same set of 20 random start and goal states.

4.2.1. Comparative Analyses

Comparison with goal-conditioned RL: Figure 5 shows that we comfortably outperform our flat goal-conditioned

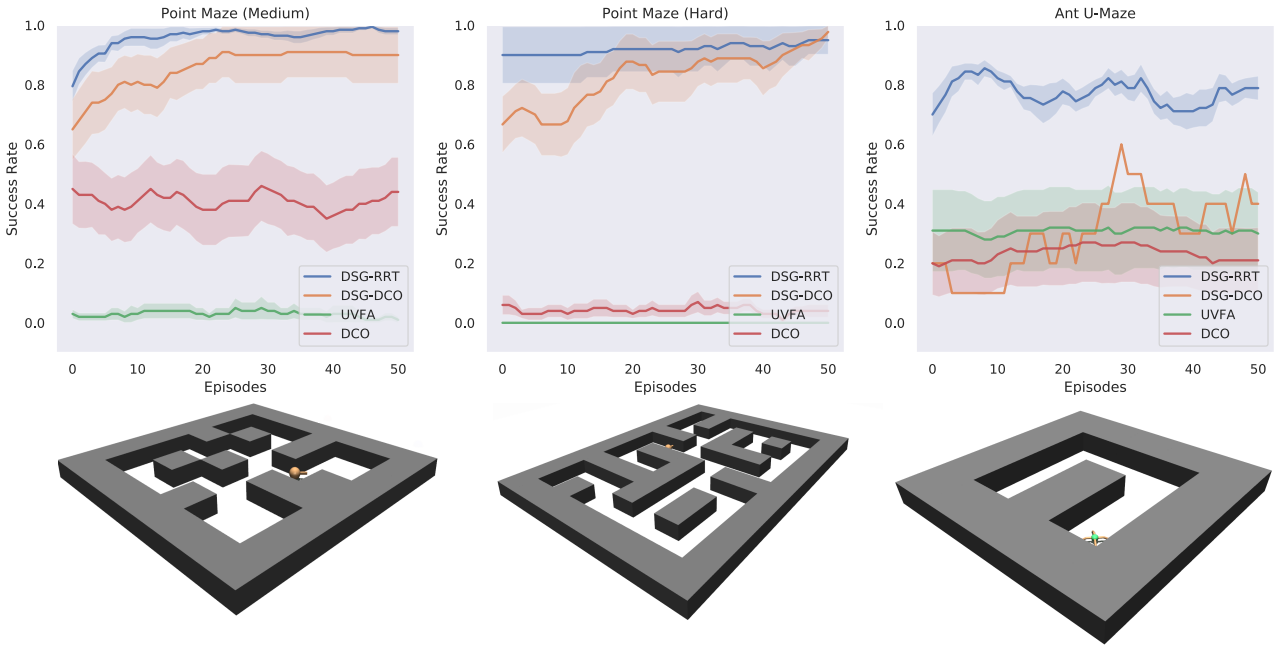

Figure 5. Test time learning curves comparing two variants of deep skill graphs (DSG-RRT and DSG-DCO) with goal-conditioned RL (UVFA) and deep covering options (DCO). The vertical axis represents the fraction of times the agent was able to reach a randomly sampled goal in the given episodic budget. Solid lines represent mean success rate and error bands represent standard error measured over 20 runs. For more details about hyperparameter settings and baseline implementations, please refer to section A.5 of the appendix.

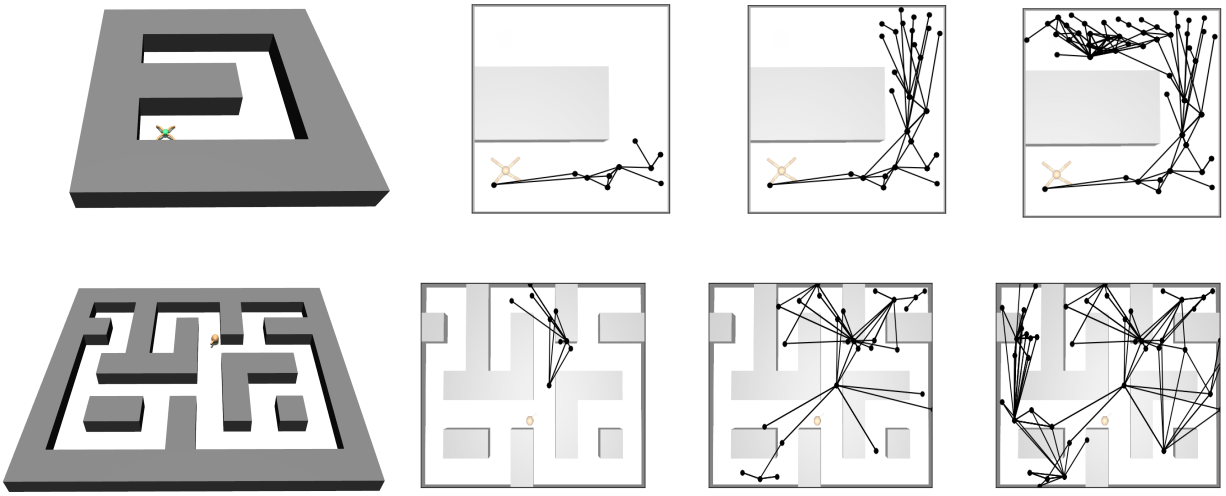

Figure 6. Incremental expansion of the skill graph: black dots represent salient events; a line connects two events if there exists option(s) that take the agent from one event to the other. Not shown here are the individual options that the agent constructs to go between each of the visualized events. These are graphs were constructed using DSG-RRT, similar graphs can be constructed using DSG-DCO.

RL baseline. Our results support the findings of Nasiriany et al. (2019) that while UVFAs can in principle be used to solve any goal-reaching task, their effectiveness seems to be limited to situations where the goal state is near the agent's start state. By contrast, DSG learns skills that specialize in different parts of the state-space, making them more effective at solving long-horizon problems.

Comparison with deep covering options: DCO learns options that allow it to effectively explore the MDP. However, these options are isolated, so the agent must use SMDP Q-learning (Bradtke & Duff, 1995) to compose options and primitive actions to reach a goal given at test-time. Learning an effective policy over options in this setting demands a lot of data, making DCO under-perform when being tested for few-shot generalization. DSG, on the other hand, uses planning to compose options at test time and hence requires little-to-no additional interactions with the environment.

Impact of salience discovery method: Figure 5 shows that DSG-RRT consistently outperforms the DSG-DCO variant of our algorithm. We found that the skill-graph produced by randomly sampling salient events was denser than the one constructed by using salient events generated by DCO. As a result, a test-time goal state tended to be inside the graph more often for DSG-RRT than for DSG-DCO. This meant that DSG-RRT could plan and execute skills it had already learned to move close to the goal, while DSG-DCO often had to extend the graph towards the goal state—a process that can be a relatively sample inefficient. An analysis of DSG-DCO's poor performance on ant-maze is presented in section A.1.3 of the appendix.

5. Related Work

Skill Discovery in high-dimensional spaces: Recent work on goal-directed skill acquisition in large continuous spaces can be broadly divided into three categories: (1) option-critic methods (Bacon et al., 2017; Harutyunyan et al., 2019; Khetarpal & Precup, 2019; Klissarov et al., 2017; Tiwari & Thomas, 2019; Riemer et al., 2018; Harb et al., 2018) which describe an end-to-end architecture for learning options in high-dimensional spaces, (2) feudal methods (Dayan & Hinton, 1993; Vezhnevets et al., 2017; Nachum et al., 2018; Levy et al., 2019; Li et al., 2019) in which a higher-level manager outputs goals for lower level workers to achieve and (3) skill chaining methods (Konidaris & Barto, 2009; Konidaris et al., 2012; Konidaris, 2016; Shoeleh & Asadpour, 2017; Metzen & Kirchner, 2013; Bagaria & Konidaris, 2020) which deconstruct the solution path of an MDP into a series of shorter horizon skills. While these methods demonstrate substantial progress on the skill discovery question, they optimize for a task-specific objective and hence cannot be directly used for the multitask setting.

Combining model-free RL and planning: Some recent works have sought to bridge model-free RL and planning. SoRB (Eysenbach et al., 2019b) uses graph search on the replay buffer (Lin, 1993) to find efficient paths between observations made by a model-free policy. PRM-RL (Faust et al., 2018) replaces the local planner used in PRM methods (LaValle, 2006) with an RL policy. Both SoRB and PRM-RL assume access either to a well trained value function or an RL policy that can be meaningfully queried in arbitrary parts of the state-space. By contrast, we learn skill policies that specialize in different parts of the state-space. In LEAP (Nasiriany et al., 2019), a planner generates subgoals for a low level model-free policy to meet. However, their planner uses a pre-trained generative model to generate high-dimensional goals in parts of the state-space the RL agent may not have seen. Our algorithm automatically extends the skill graph towards unexplored regions. SPTM (Savinov et al., 2018) is a memory augmented RL agent that plans over raw landmark observations in navigation problems. Our method bypasses the difficulty of planning in high-dimensional spaces by instead planning over abstract states. Most recently, DADS (Sharma et al., 2020) achieved impressive results by using Model Predictive Control to compose learned skills to reach novel goal states at test time. However, their algorithm does not interleave skill-discovery and high-level planning, making them unable to exploit the effectiveness of temporal abstraction to explore from states that are difficult to get to.

Exploration in RL: In the non-hierarchical setting, notions of novelty (Strehl & Littman, 2008; Bellemare et al., 2016) and prediction error (Oudeyer et al., 2008; Barto et al., 2004; Burda et al., 2019; Pathak et al., 2017; Badia et al., 2020) have been used to drive the agent to previously unseen regions of the state-space (Taïga et al., 2019). Recent work in HRL based intrinsic motivation has fallen into two broad categories: (1) empowerment (Klyubin et al., 2005) driven methods (Florensa et al., 2016; Hausman et al., 2018; Eysenbach et al., 2019a; Gregor et al., 2016; Heess et al., 2017; Sharma et al., 2020) which optimize an entropy based objective and (2) spectral methods (Mahadevan & Maggioni, 2007; Machado et al., 2017; Jinnai et al.; 2020) that exploit structure in the transition dynamics of the environment. While these methods generally focus on the exploration question alone, we show that additionally combining them with options for exploitation eventually facilitates better exploration.

6. Conclusion

We introduced a practical algorithm that compresses a large continuous MDP into a small abstract representation suitable for planning. Like all RL agents, DSG starts by exploring the environment since it does not know enough about it to

plan. It then discovers salient events and the skills it needs to reliably trigger those salient events. Together, these skills and target events enable high-level planning.

We showed that skill graphs grow outward from the start-state towards large unexplored regions, reflecting mastery over ever increasing portions of the state-space. Finally, we tested our algorithm on a series of maze navigation tasks and showed that DSG can be used to reach novel goals at test time in a small number of trials. We compared the few-shot generalization capability of our algorithm to that of popular flat and hierarchical alternatives and showed that DSG can significantly outperform them.

References

Abel, D. and Littman, M. *A Theory of Abstraction in Reinforcement Learning*. PhD thesis, Brown University, 2020.

Bacon, P.-L. *Temporal Representation Learning*. PhD thesis, McGill University Libraries, 2018.

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

Badia, A. P., Sprechmann, P., Vitvitskyi, A., Guo, D., Piot, B., Kapturowski, S., Tieleman, O., Arjovsky, M., Pritzel, A., Bolt, A., et al. Never give up: Learning directed exploration strategies. In *ICLR*, 2020.

Bagaria, A. and Konidaris, G. Option discovery using deep skill chaining. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=B1gqipNYwH.

Barto, A. G. and Mahadevan, S. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(1-2):41–77, 2003.

Barto, A. G., Singh, S., and Chentanez, N. Intrinsically motivated learning of hierarchical collections of skills. In *Proceedings of the 3rd International Conference on Development and Learning*, pp. 112–19, 2004.

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pp. 1471–1479, 2016.

Bradtke, S. J. and Duff, M. O. Reinforcement learning methods for continuous-time markov decision problems. In *Advances in neural information processing systems*, pp. 393–400, 1995.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Brunskill, E. and Li, L. Sample complexity of multi-task reinforcement learning. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, pp. 122–131, 2013.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1lJJnR5Ym.

Campbell, M., Hoane Jr, A. J., and Hsu, F.-h. Deep blue. *Artificial intelligence*, 134(1-2):57–83, 2002.

Chentanez, N., Barto, A. G., and Singh, S. P. Intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pp. 1281–1288, 2005.

Chung, F. R. and Graham, F. C. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

Dayan, P. and Hinton, G. E. Feudal reinforcement learning. In *Advances in neural information processing systems*, pp. 271–278, 1993.

Dijkstra, E. W. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pp. 1329–1338, 2016.

Even-Dar, E., Kakade, S. M., and Mansour, Y. Reinforcement learning in pomdps without resets. 2005.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019a. URL https://openreview.net/forum?id=SJx63jRqFm.

Eysenbach, B., Salakhutdinov, R. R., and Levine, S. Search on the replay buffer: Bridging planning and reinforcement learning. In *Advances in Neural Information Processing Systems 32*, pp. 15246–15257. 2019b.

Farebrother, J., Machado, M. C., and Bowling, M. Generalization and regularization in dqn. *arXiv preprint arXiv:1810.00123*, 2018.

Faust, A., Oslund, K., Ramirez, O., Francis, A., Tapia, L., Fiser, M., and Davidson, J. Prm-rl: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5113–5120. IEEE, 2018.

Fiedler, M. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2):298–305, 1973.

Florensa, C., Duan, Y., and Abbeel, P. Stochastic neural networks for hierarchical reinforcement learning. In *International Conference on Learning Representations*, 2016.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pp. 1582–1591, 2018.

Giunchiglia, F. and Walsh, T. A theory of abstraction. *Artificial intelligence*, 57(2-3):323–389, 1992.

Gregor, K., Rezende, D. J., and Wierstra, D. Variational intrinsic control. *ArXiv*, abs/1611.07507, 2016.

Ha, D. and Schmidhuber, J. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

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

Harutyunyan, A., Dabney, W., Borsa, D., Heess, N., Munos, R., and Precup, D. The termination critic. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2231–2240, 2019.

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

Heess, N., TB, D., Sriram, S., Lemmon, J., Merel, J., Wayne, G., Tassa, Y., Erez, T., Wang, Z., Eslami, S., et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

Jinnai, Y., Park, J. W., Abel, D., and Konidaris, G. Discovering options for exploration by minimizing cover time. In *Proceedings of the 36th International Conference on Machine Learning*. PMLR. URL http://proceedings.mlr.press/v97/jinnai19b.html.

Jinnai, Y., Park, J. W., Machado, M. C., and Konidaris, G. Exploration in reinforcement learning with deep covering options. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SkeIyaVtwB.

Kaelbling, L. P. Learning to achieve goals. In *IJCAI*, pp. 1094–1099. Citeseer, 1993.

Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., and Levine, S. Model-based reinforcement learning for atari. *International Conference on Learning Representations (ICLR)*, 2020.

Kakade, S. and Langford, J. Approximately optimal approximate reinforcement learning. In *ICML*, volume 2, pp. 267–274, 2002.

Kavraki, L. E., Svestka, P., Latombe, J.-C., and Overmars, M. H. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.

Khetarpal, K. and Precup, D. Learning options with interest functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 9955–9956, 2019.

Klissarov, M., Bacon, P.-L., Harb, J., and Precup, D. Learnings options end-to-end for continuous action tasks. *Hierarchical Reinforcement Learning Workshop (NeurIPS)*, 2017.

Klyubin, A. S., Polani, D., and Nehaniv, C. L. Empowerment: A universal agent-centric measure of control. In *2005 IEEE Congress on Evolutionary Computation*, volume 1, pp. 128–135. IEEE, 2005.

Konidaris, G. Constructing abstraction hierarchies using a skill-symbol loop. In *IJCAI: proceedings of the conference*, volume 2016, pp. 1648. NIH Public Access, 2016.

Konidaris, G. On the necessity of abstraction. *Current Opinion in Behavioral Sciences*, 29:1–7, 2019.

Konidaris, G. and Barto, A. Skill discovery in continuous reinforcement learning domains using skill chaining. In *Advances in Neural Information Processing Systems*, pp. 1015–1023, 2009.

Konidaris, G. and Barto, A. *Autonomous Robot Skill Acquisition*. PhD thesis, University of Massachusetts at Amherst, 2011.

Konidaris, G., Kuindersma, S., Grupen, R., and Barto, A. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, 31 (3):360–375, 2012.

Konidaris, G., Kaelbling, L. P., and Lozano-Perez, T. From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 61:215–289, 2018.

LaValle, S. M. Rapidly-exploring random trees: A new tool for path planning. 1998.

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

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

Levy, A., Platt, R., and Saenko, K. Hierarchical reinforcement learning with hindsight. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryzECoAcY7.

Li, S., Wang, R., Tang, M., and Zhang, C. Hierarchical reinforcement learning with advantage-based auxiliary rewards. In *Advances in Neural Information Processing Systems*, pp. 1409–1419. 2019.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Lin, L.-J. Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 1993.

Lindemann, S. R. and LaValle, S. M. Incrementally reducing dispersion by increasing voronoi bias in rrts. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 4, pp. 3251–3257. IEEE, 2004.

Liu, M., Machado, M. C., Tesauro, G., and Campbell, M. The eigenoption-critic framework. *arXiv preprint arXiv:1712.04065*, 2017.

Machado, M. C., Bellemare, M. G., and Bowling, M. A laplacian framework for option discovery in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 2295–2304. JMLR.org, 2017.

Machado, M. C., Rosenbaum, C., Guo, X., Liu, M., Tesauro, G., and Campbell, M. Eigenoption discovery through the deep successor representation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Bk8ZcAxR-.

Mahadevan, S. and Maggioni, M. Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. *Journal of Machine Learning Research*, 8(Oct):2169–2231, 2007.

Mcgovern, E. A. and Barto, A. G. *Autonomous discovery of temporal abstractions from interaction with an environment*. PhD thesis, University of Massachusetts at Amherst, 2002.

Metzen, J. H. and Kirchner, F. Incremental learning of skill collections based on intrinsic motivation. *Frontiers in neurorobotics*, 7:11, 2013.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529, 2015.

Munos, R., Stepleton, T., Harutyunyan, A., and Bellemare, M. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems 29*, pp. 1054–1062. 2016.

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

Nasiriany, S., Pong, V., Lin, S., and Levine, S. Planning with goal-conditioned policies. 2019.

Oudeyer, P.-Y., Kaplan, F., et al. How can we define intrinsic motivation. In *Proc. of the 8th Conf. on Epigenetic Robotics*, volume 5, pp. 29–31, 2008.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 16–17, 2017.

Precup, D. *Temporal abstraction in reinforcement learning*. PhD thesis, 2001.

Ravindran, B. and Barto, A. G. *An algebraic approach to abstraction in reinforcement learning*. PhD thesis, University of Massachusetts at Amherst, 2004.

Riemer, M., Liu, M., and Tesauro, G. Learning abstract options. In *Advances in Neural Information Processing Systems*, pp. 10424–10434, 2018.

Savinov, N., Dosovitskiy, A., and Koltun, V. Semi-parametric topological memory for navigation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SygwwGbRW.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. Universal value function approximators. In *International conference on machine learning*, pp. 1312–1320, 2015.

Sharma, A., Gu, S., Levine, S., Kumar, V., and Hausman, K. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations (ICLR)*, 2020.

Shoeleh, F. and Asadpour, M. Graph based skill acquisition and transfer learning for continuous reinforcement learning domains. *Pattern Recognition Letters*, 87:104–116, 2017.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Stout, A. and Barto, A. G. Competence progress intrinsic motivation. In *2010 IEEE 9th International Conference on Development and Learning*, pp. 257–262. IEEE, 2010.

Strehl, A. L. and Littman, M. L. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

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

Sutton, R. S. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Sutton, R. S., Koop, A., and Silver, D. On the role of tracking in stationary environments. In *Proceedings of the 24th International Conference on Machine Learning*, pp. 871–878, 2007.

Taïga, A. A., Fedus, W., Machado, M. C., Courville, A., and Bellemare, M. G. Benchmarking bonus-based exploration methods on the arcade learning environment. *arXiv preprint arXiv:1908.02388*, 2019.

Tanaka, F. and Yamamura, M. Multitask reinforcement learning on the distribution of mdps. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation*. IEEE, 2003.

Taylor, M. E., Kulis, B., and Sha, F. Metric learning for reinforcement learning agents. In *AAMAS*, pp. 777–784, 2011.

Thomas, P. and Brunskill, E. Data-efficient off-policy policy evaluation for reinforcement learning. In *International Conference on Machine Learning*, pp. 2139–2148, 2016.

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

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

Veeriah, V., Oh, J., and Singh, S. Many-goals reinforcement learning. *arXiv preprint arXiv:1806.09605*, 2018.

Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., and Kavukcuoglu, K. Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3540–3549. JMLR. org, 2017.

Wilson, A., Fern, A., Ray, S., and Tadepalli, P. Multi-task reinforcement learning: a hierarchical bayesian approach. In *Proceedings of the 24th international conference on Machine learning*, pp. 1015–1022, 2007.

Witty, S., Lee, J. K., Tosch, E., Atrey, A., Littman, M., and Jensen, D. Measuring and characterizing generalization in deep reinforcement learning. *arXiv preprint arXiv:1812.02868*, 2018.

Wu, Y., Tucker, G., and Nachum, O. The laplacian in RL: Learning representations with efficient approximations. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HJlNpoA5YQ.

A. Appendix

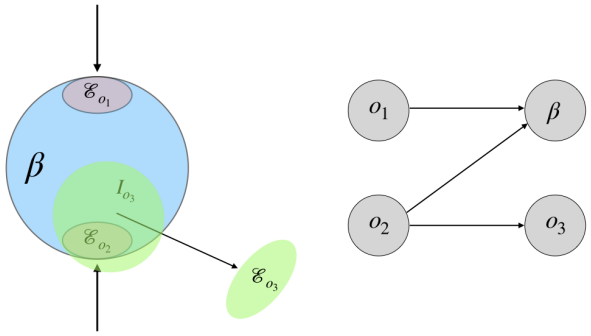

Figure 7. How initiation and effect sets of learned options correspond to connections in the abstract skill-graph: the blue sphere represents a salient event β. The red regions represent the effect set of two different options o_1, o_2 targeting β. The green regions represent the initiation and effect set of another option o_3. Because $\mathcal{E}_{o_2} \subseteq \mathcal{I}_{o_3}$, we can create an edge from $o_2 \rightarrow o_3$ in the abstract graph shown on the right.

A.1. Using Deep Covering Options for Salient Event Discovery

DCO approximates the Fiedler vector, which is a function that projects states into a one dimensional embedding. In DCO, all states that have a fiedler value lower than some pre-defined threshold are marked as the termination condition of an option. However, as noted by Jinnai et al., this scheme can often lead to termination sets that are either too small or those that are too big. If we were to directly use such termination conditions as salient events, the resulting salient events themselves could be too small or too big. As a result, we discard the termination condition discovered by DCO and simply pick the states with the highest and lowest values under the approximate Fiedler vector. We then construct an ϵ-ball around that state and treat it as a salient event. Creating this ϵ-ball assumes a meaningful metric, which is readily available in the domains considered here. However, scaling our method to domains with pixel observations would require a deeper understanding of how the size of DCO termination conditions can be better controlled (Harutyunyan et al., 2019).

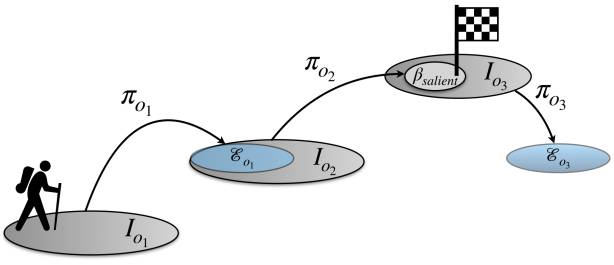

Figure 8. Conditions under which vertices in the skill-graph can be connected by an edge between them: An edge from o_1 to o_2 exists because $\mathcal{E}_{o_1} \subseteq \mathcal{I}_{o_2}$. An edge from o_2 to $\beta_{salient}$ exists because o_2 targets $\beta_{salient}$. Finally, an edge exists from $\beta_{salient}$ to o_3 because $\beta_{salient} \subseteq \mathcal{I}_{o_3}$.

A.1.1. Filtering salient events

So as to not create options that target redundant salient events, we may choose to reject a generated salient event before we add them to the list of target events. A generated event is rejected if it is either (a) already satisfied by an existing event or (b) the generated event is inside the initiation classifier of an option already in the graph.

A.1.2. When to generate new salient events

We attempt to generate a new pair of salient events when any of the following three conditions are satisfied:

1. The most recent generated salient event has been added to the graph
2. It has been $K = 30$ episodes since a salient event was generated and accepted
3. It has been $R = 10$ episodes since the last pair of salient events were generated, and both were rejected (according to criteria described in A.1.1).

A.1.3. DSG-DCO on Ant-Maze

Figure 9 shows that in the Ant-Maze domain, DCO fails to discover salient events in the top left portion of the state-space. As a result, DSG is unable to expand the graph into that region of the state space and it thus fails to reliably reach goal states there. DSG-DCO's failure to expand the skill-graph in the top part of the maze is principally responsible for its poor performance in Ant-Maze.

A.2. Setting edge weights in the skill graph

As described in section 3.1, the edge weight between vertices i, j is given by $w_{i,j} = \frac{1}{success\_rate(i,j)}$, where $success\_rate(i, j)$ is the fraction of times option i successfully executed its policy to reach option j's initiation set. The same formula works when vertex i corresponds to an option and vertex j corresponds to a salient event. However, when vertex i corresponds to a salient event and vertex j corresponds to an option, then the edge weight $w_{i,j} = 0$ because being inside the salient event implies being inside the initiation set of option o_j. As described in section 3.1, the edge weight between vertices i, j is given

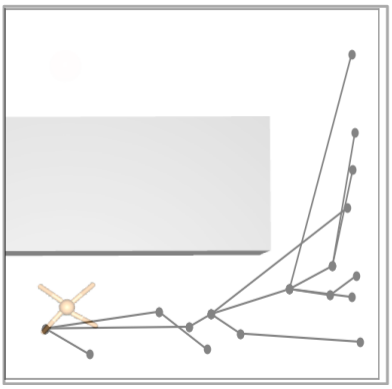

Figure 9. Failure mode of DSG-DCO: In ant-maze, DCO failed to generate salient events in the top left part of the maze—preventing DSG from extending its graph into that region of the state-space.

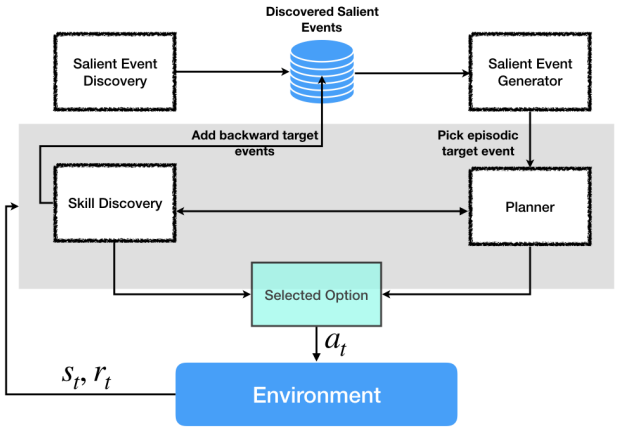

Figure 10. The Deep Skill Graphs system architecture

by $w_{i,j} = \frac{1}{success\_rate(i,j)}$, where $success\_rate(i,j)$ is the fraction of times option i successfully executed its policy to reach option j's initiation set. The same formula works when vertex i corresponds to an option and vertex j corresponds to a salient event. However, when vertex i corresponds to a salient event and vertex j corresponds to an option, then the edge weight $w_{i,j} = 0$ because being inside the salient event implies being inside the initiation set of option o_j.

A.3. More details on learning backward options

Consider a skill chain $\mathcal{C}$ that starts at β_0 and targets $\beta_{salient}$. $\mathcal{C}$ is considered "complete" when it learns an option o such that $\beta_0 \subseteq \mathcal{I}_o$. At this point, we start learning a new skill chain $\mathcal{C}_{back}$ that targets β_0 and chains until $\beta_{salient}$. As is typical with skill chaining, $\mathcal{C}_{back}$ learns as many options as it needs to reliably drive the agent from $\beta_{salient}$ to β_0.

A.4. Test environments

We evaluated our algorithm in four tasks that exhibit a strong hierarchical structure (Nachum et al., 2018; Fu et al., 2020; Brockman et al., 2016; Duan et al., 2016): (1) Point Maze (Medium), (2) Point Maze (Hard), (3) Ant U-Maze and (4) Ant Reacher. Learning curves in figure 5 are shown for environments 1-3. In ant reacher, there is no maze, and the ant is required to navigate an open area spanned by $[-10, 10]^2$. Only qualitative results are presented in figure 3 for the Ant Reacher domain. In each task, the agent is reset back to its start state (a small distribution around $(0, 0)$) after 1000 steps per episode.

These learning curves in figure 5 are generated after we train each algorithm for 2000 episodes in problems involving the point robot and for 4000 episodes in problems involving the more complex ant robot. Each episode lasts 1000 time steps.

A.5. Baseline Implementation Details

Following Bagaria & Konidaris (2020), we use the Deep Deterministic Policy Gradients (DDPG) algorithm (Lillicrap et al., 2015) to train option policies. As a result, we compare against a goal-conditioned version of DDPG as our goal-conditioned RL baseline. In the same way that our algorithm samples a salient event every 30 episodes, the goal-conditioned DDPG samples a goal state at random every 30 episodes. It gets a positive terminal reward for successfully reaching this goal.

All hyper-parameters for our implementation of DCO were taken from Jinnai et al. (2020). In the same way that DSG generates a new salient event every 30 episodes, so does DCO. Unlike DSG, DCO requires the number of skills to learn to be specified as an input to the algorithm. We evaluated DCO with $3, 4, 5, 10$ options and present the best performing ones in figure 5.

A.6. System Architecture

Figure 10 shows the system architecture of the deep skill graph agent. Both the salient event discovery module and the deep skill chaining agent (Skill Discovery module) may find salient events in the MDP, all of which are stored in a single list. When the agent is in a state inside the graph and there exists a path in the graph to the chosen target event, the planner selects the option to execute. If the agent is outside the graph or when there is no path from the current state to the selected goal state, the policy-over-options in the DSC module picks the option to execute. All options execute their closed loop policies until they reach their termination set or timeout at 200 steps.

A.7. The x-y prior

Option initiation sets are learned over the same state variables that factor in the reward function. This is the same assumption as related work (Bagaria & Konidaris, 2020; Levy et al., 2019; Nachum et al., 2018; Sharma et al., 2020; Eysenbach et al., 2019a).

A.8. DDPG Hyperparameters

We used DDPG to train option policies and as the basis for our goal-conditioned RL baseline. We did not try to optimize over the space of DDPG hyperparameters and used the ones reported in previous work (Lillicrap et al., 2015; Fujimoto et al., 2018).

Parameter	**Value**
Replay buffer size	$1e6$
Batch size	64
γ	0.99
τ	0.01
Number of hidden layers	2
Hidden size 1	400
Hidden size 2	300
Critic learning rate	$1e-3$
Actor learning rate	$1e-4$

Table 1. DDPG Hyperparameters

A.9. Compute Infrastructure

We used 1 NVIDIA GeForce 2080 Ti and 4 NVIDIA GeForce 2070 Ti GPUs to perform all experiments reported in this paper.