DEEP HIERARCHICAL LAPLACIAN SKILL DISCOVERY

Anonymous Authors

ABSTRACT

Temporally-abstracted actions, often referred to as skills or options, provide an appealing set of benefits for learning how to both explore environments and execute tasks. But many methods rely on extrinsic reward signal to build good options which may not be available at train-time or may be expensive to collect. We therefore focus on the problem of *discovering* skills via unsupervised environment interaction. Inspired by the compositionality theorized in biological learning, we iteratively learn a hierarchy of skills. Specifically, we learn deep Laplacian representations on exploration policies utilizing lower level skills. The learned Laplacian representations are then used to create new skills to update the exploration policy and repeat the process. The result is a feedback loop that learns representations from skills and skills from representations in order to continually extend the agent's capabilities. We show our method composes lower level skills and incorporates increasingly complex skills to better explore and plan. Our results display this method outperforms state-covering and state-novelty methods for solving long-horizon downstream tasks and exploring environments effectively.

1 INTRODUCTION

General task solving, autonomy, and learning from experience has long taken advantage of Reinforcement Learning (RL) (Sutton & Barto, 2018). The RL framework makes a few general assumptions and allows for sequential decision making with an agent that interacts with an environment and acts to maximize its reward. This reward can be designed and tends to capture what the intended task at hand is, and as a result, the agent can learn how to solve the task by optimizing for the cumulative reward, or return. A major problem in RL is the difficulty that arises when trying to correctly explore the environment to find a decision making sequence, or policy, that maximizes return in the sparse-reward case. Classically, the agent has to rely on random chance to run into a high-reward transition and then continue its search from there. This is especially difficult in long-horizon tasks where very little signal is given to the agent for it to know if its doing the right thing.

The problem of augmenting RL for better exploration and policy learning has long been studied. One line of work is the use of novelty metrics to intrinsically reward the agent for exploring new states in the environment, such as Random Network Distillation (Burda et al., 2018), curiosity, count-based metrics, or empowerment (Bellemare et al., 2016; Pathak et al., 2017; Klyubin et al., 2005). However, since these methods only operate as reward bonuses, they cannot transfer well to other areas of the state space and do not encourage the chaining of interesting behaviors together. As a result, they are often times only used to explore for accomplishing a single, specified task instead of providing primitives to solve many tasks. A more desirable approach would be a task-agnostic one that creates modular, re-usable behaviors. Consider the case of a robotic system for search and rescue. A system like this would require general exploration skills for reaching unknown parts of the area, not just for a single task it was trained on. Furthermore, to rapidly complete new task specifications, it must have general knowledge about the environment and its own capabilities.

We study the set of approaches focused on creating new, temporally extended actions that consist of low-level action sequences. These are commonly referred to as options or skills and are policies that can be run to help both exploration and downstream planning (Sutton et al., 1999). As opposed to exploration bonuses, these option learning methods learn explicit new policies for potential composition. Favorably, these could be used in a task-agnostic way to assist planning and exploration. But most methods for finding interesting options rely on a good extrinsic reward-signal and thus do not allow for creating a general skill library. (Bacon et al., 2016; Xu et al., 2022; Pertsch et al., 2020).

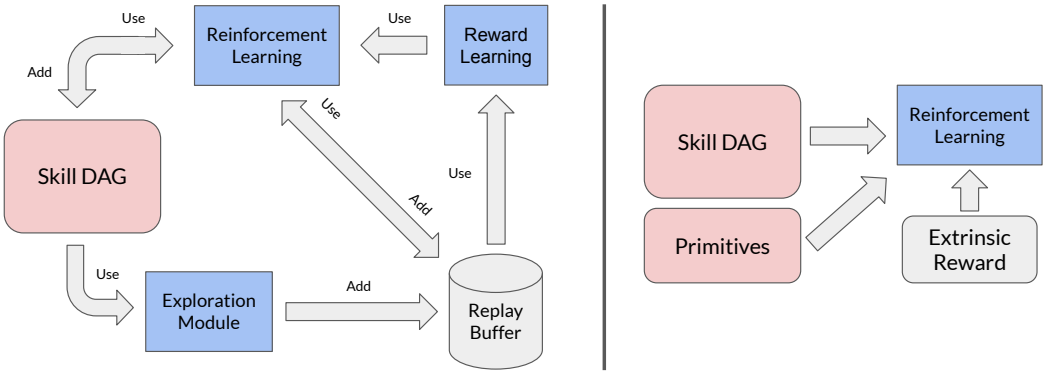

Figure 1: **Left:** the pretraining phase feedback loop that LaSH utilizes to build a Skill Directed Acyclic Graph (DAG). It is flexible to any RL algorithm, intrinsic reward, and exploration method. **Right:** the task solving phase of LaSH which involves using the DAG to learn a policy to solve a task.

As a result, we look towards the problem setting that utilizes intrinsic motivation bonuses to learn explicit general skills, thus affording us a reward-free way of getting composable policies for extended exploration and downstream planning. In this way, it's a combination of classic option learning and classic intrinsic motivation and thus enjoys multiple benefits. Furthermore, it allows for an agent to pretrain in an environment and be ready for fast adaptation to new tasks in that environment.This study is often referred to as skill discovery.

Many reward-free skill discovery approaches use the mutual information between states and skills to make distinct behaviors (Eysenbach et al., 2018), but these behaviors have no incentive to cover the environment well – which is imperative for sparse, long-horizon tasks. Furthermore, many skill discovery methods lack the ability to continually use existing capabilities to bootstrap learning of more complex behaviors, as a human would. If we seek to learn increasingly complex behaviors from current behaviors, we need corresponding higher level representations of the environment to learn from. Meanwhile these higher level representations are only helpful if they reflect the agent's current capabilities. Thus, an iterative approach to skill discovery that interweaves skill learning and representation learning is one that we desire (Machado et al., 2023). A well-studied approach for creating representations from current agent capabilities for defining options is the proto-value function (PVF) which can be acquired from the graph Laplacian eigenvectors (Mahadevan, 2005; Johns & Mahadevan, 2007; Mahadevan & Maggioni, 2007b).

We elect to utilize a deep graph Laplacian, a model of temporal diffusion in an environment, to create an intrinsic reward signal that incorporates the environment's structure and the agent's capabilities. We then utilize vanilla RL, specifically DQN (Mnih et al., 2013), to learn explicit, re-usable skills that maximize intrinsic return. We build upon existing work by running a feedback loop to chain the current level of skills learned thus far to re-learn a temporal diffusion graph Laplacian and, subsequently, new options that are more temporally extended. We show our method, LaSH, is capable of planning in long-horizon sparse-reward tasks when used with RL after pretraining. To our knowledge, it is the first deep Laplacian method for creating explicit skill hierarchies. Our work's main contributions are:

1. A reward-free iterative feedback loop to create a continually evolving skill hierarchy.
2. The usage of this hierarchy to augment the action space for extrinsic planning.
3. The greedy chaining of skills for learning incrementally more complex representations during exploration.

2 Related Work

Methods for solving temporally abstracted RL have mostly been introduced via the Hierarchical RL (HRL) framework where there are two controllers: one to choose options, and one to execute them (Sutton et al., 1999; Precup, 2000). Feudal RL is a similar technique for doing HRL from the perspective of subgoal commands (Dayan & Hinton, 1992; Vezhnevets et al., 2017; Nachum et al.,

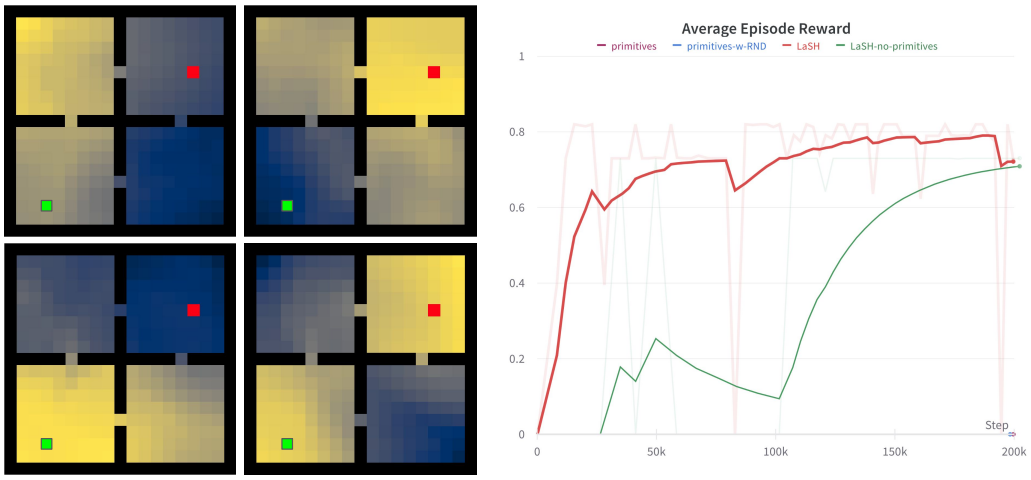

Figure 2: **Left:** We show the Laplacian representation as a heatmap overlaid on the 4-room environment. **Right:** We also show the reward curves over time as the models are run with extrinsic sparse reward access.

2018). The Hierarchy of Abstract Machines (HAM) (Parr & Russell, 1997; Bai & Russell, 2017) framework also tackles the HRL problem through encoding prior knowledge of the MDP structure to simplify policy learning. Central to most HRL methods is an extrinsic reward which is used to tune the low-level option controller and the high-level meta-controller (Bacon et al., 2016; Frans et al., 2017; Bagaria & Konidaris, 2020), some minimal supervision or proxy task (Florensa et al., 2017; Tessler et al., 2016), or the inclusion of demonstrations (Fox et al., 2017; Krishnan et al., 2017). The use of these task-specific signals can create degenerate behaviors requiring regularizers. (Harb et al., 2017; Alexander et al., 2016; Khetarpal et al., 2020; Kulkarni et al., 2016a). The unsupervised setting, on the other hand, remains agnostic to tasks allowing for general options.

Simple skill discovery methods have been proposed using state visitation statistics, RRT, (LaValle, 1998) or graph theory to determine bottleneck states (i.e. states that are essential to travel between connected components of the environment). However, these methods are difficult to scale and and may require special hand-engineered heuristics. Importantly, explicit state graphs hinder the ability to generalize across similar states which can then lead to requiring exhaustive search to get good bottlenecks (McGovern & Barto, 2001; Menache et al., 2002; Iba, 1989; Şimşek et al., 2005; Şimşek & Barto, 2008; Bagaria et al., 2020). Some methods use random options purely for exploration and not planning (Dabney et al., 2020), and as a result, do not learn with experience. We work towards a skill-discovery method scalable to modern deep learning approaches as opposed to these graph-based systems or other approaches (Kompella et al., 2017).

Deep unsupervised skill discovery methods chiefly differ in their skill learning objectives. Some have proposed using mutual information (MI) as an objective to learn maximally distinguishable skills (Eysenbach et al., 2018). This has been built upon by adding predictability as a implicit secondary objective to ensure the skills are well-suited for downstream planning (Sharma et al., 2019). These can ultimately be used in a hierarchical controller to solve tasks without demonstrations (Daniel et al., 2016). The inherent goal of these methods is to learn one level of behaviors that are maximally different from each other in order to cover the space of behaviors (Warde-Farley et al., 2018; Laskin et al., 2022). In this way, these methods attempt to incorporate all behaviors but only at one level of abstraction; as a result, during skill acquisition, we cannot learn increasingly complex behaviors. Furthermore, these methods may require learning a dynamics model (Sharma et al., 2019) on the skills which might be difficult to scale to pixel-level tasks.

Algorithm 1 LaSH Skill Discovery

skillDAG = $\mathcal{A}$
for each level **do**
  Fill $\mathcal{D}$ with LaSH- Explore(skillDAG)
  Train representation on $\mathcal{D}$ via 3
  Reward label $\mathcal{D}$ using 3.2
  Initialize new skill policies
  for learnStep **do**
    $a \sim$ random new skill action
    Update $\mathcal{D}$ with transition (s, a, s', r)
    Update skills via RL
  end for
  Add skills to skillDAG, empty $\mathcal{D}$
end for

Algorithm 2 LaSH-Explore

$\mathcal{D}$ = emptyBuffer()
$O \leftarrow$ skillDAG
for exploreStep **do**
  while episode is not done **do**
    $\pi_o \leftarrow \arg\max_O V^{\pi_o}(s)$
    Rollout π_o and add to τ
  end while
  $\mathcal{D} \leftarrow \mathcal{D} \cup \tau$
end for

Methods like these may not be well-suited for long-horizon exploration as they don't build increasingly complex options during training, and could thus generate a large amount of unhelpful basic options (Gregor et al., 2016). We therefore elect for a state-coverage based objective inspired by the graph Laplacian (Wu et al., 2018) as done by (Machado et al., 2017; Bar et al., 2020) and extended to deep function approximation by (Klissarov & Machado, 2023). These methods are highly similar to successor-based objectives whereby a successor representation (Dayan, 1993; Kulkarni et al., 2016b) is learned and is used to derive eigen-directions of state variation to act along (Machado et al., 2018). These methods are referred to as covering-option approaches (Jinnai et al., 2019b;a) and take advantage of the proto-value functions to generate new options (Jinnai et al., 2020; Mahadevan & Maggioni, 2007a) but are often used for exploration and not planning. In an ideal case, we should pursue learning skills that would be useful for accomplishing tasks though, not just for exploring. Learning a policy upon options has a great number of benefits, mainly arising from faster backups when running RL.

Our method, the Laplacian Skill Hierarchy (LaSH) is capable of capturing the benefits that covering-options have for state exploration and is also capable of planning hierarchically with these multi-level skills. LaSH is most similar to Deep Covering Eigen-options (DCEO) (Klissarov & Machado, 2023) but crucially creates multiple levels of abstraction in a skill-learning loop inspired by (Machado et al., 2023; Machado & Bowling, 2016) and furthermore learns a tunable policy on the skills as opposed to randomly triggering options for the purpose of exploration only as DCEO does. Additionally, we operate under the reward-free setting whereas DCEO functions under knowing the reward which violates our problem assumptions. We show that a method like this incrementally picks up more and more complex purposes to better cover the state space, significantly helping solve long-horizon, sparse-reward tasks.

3 BACKGROUND

An agent interacts in a Markov Decision Process (MDP) defined by state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $\mathcal{P}$, and reward function $\mathcal{R}$. The goal of the agent is to maximize the expected discounted return $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t]$, where $\gamma \in [0, 1)$ is a discount factor and r_t is the reward at time t.

We further augment this setup with the option framework thus making this a Semi-Markov Decision Process (SMDP) that has variable time action length. Classically, an option is a temporally extended action that is defined by a triple (I, o, β), where $I \subseteq \mathcal{S}$ is the initiation set, $o : \mathcal{S} \rightarrow \mathcal{A}$ is the option policy, and $\beta : \mathcal{S} \rightarrow [0, 1]$ is the termination function. This equips the agent to use both primitive single-step actions as well as extended multi-step actions encoded by policies $\pi : \mathcal{S} \rightarrow \mathcal{A}$. We use $I = \mathcal{S}$ and we use β as a fixed option length L, thus making β a piecewise constant function 1.

$$\beta(s) = \begin{cases} 1 & \text{if } t \geq L \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Our objective is to learn these options in a hierarchical manner via unsupervised environment interaction. We build off of work on covering eigen-options (CEO) and Deep CEO (DCEO) which rely on two main steps: (1) learning a representation and corresponding reward as covered in Section 3.1, and (2) using the reward to derive options from RL, as covered in Section 3.2. DCEO runs these two steps jointly assuming access to extrinsic reward, thus violating our problem assumptions and not explicitly composing a hierarchy of skills. Furthermore, DCEO does not use these skills for planning, but rather randomly triggers skills for exploration. We aim to modify the methods used in DCEO in order to create composable skills at multiple levels in the reward-free setting, in order to generally plan with these skills downstream.

3.1 Representation Learning

We aim to find a state representation to help us form an intrinsic reward that gives rise to effective skills. In our case, an effective skill is one that can (a) be composed to complete long horizon planning tasks and (b) help cover the state space well for exploration of sparse reward. Machado et al. (2017) showed that the Laplacian representation of temporal diffusion across an environment is an effective signal for learning state-covering options. The Laplacian representation, encoded with embedding function $f\colon \mathcal{S} \to \mathbb{R}^d$, uses the eigenvectors of the graph Laplacian, $L = D - A$, where D is the degree matrix and A is the adjacency matrix.

Specifically, $f(s) = [v_1(s), v_2(s), ..., v_d(s)]$ where each entry is the element in the i^{th} eigenvector, $v_i \in \mathbb{R}^{|\mathcal{S}|}$ that corresponds to state s for some $s \in \mathcal{S}$. Due to the nature of the graph Laplacian's discrete nature and large size, we use a spectral graph drawing objective as proposed by (Wang et al., 2021) to identify the unique d smallest eigenvalued eigenvectors amenable for neural architectures in Equation 2. As shown by Mahadevan & Maggioni (2007b), most cases of interest have value functions that may not be smooth in the state space, but are smooth on the manifold associated with it, and the smoothest Laplacian eigenvectors (i.e. PVFs) capture exactly this. In this way, our options are motivated by geometric and spatial characteristics of the MDP state space which may not be easily distinguishable in Euclidean space.

$$\begin{aligned} \min_{u_1,\dots u_T} \sum_{k=1}^{d}\sum_{i=1}^{k}(d-i+1)u_i^T L u_i \ \text{ s.t.} \\ u_i^T u_j = \delta_{ij} \ \ \forall (i,j) \in \mathbb{N}_{\leq d} \times \mathbb{N}_{\leq d} \end{aligned} \tag{2}$$

Using a neural approximation of this where each $u_i(s)$ is a scalar output $f_i(s)$ from a neural network, we get an optimizable expectation form in 3.

$$\mathbb{E}_{(s,s')\sim\mathcal{D}} \sum_{k=1}^{d}\sum_{i=1}^{k}(f_i(s) - f_i(s'))^2 + \mathbb{E}_{s\sim\mathcal{D},s'\sim\mathcal{D}} \sum_{l=1}^{d}\sum_{j=1}^{l}\sum_{k=1}^{l}(f_j(s)f_k(s) - \delta_{jk})(f_j(s')f_k(s') - \delta_{jk}) \tag{3}$$

This equation can be minimized with f parameterized by some neural network parameter set. The first expectation aims to keep consecutive states close in expectation when sampling from the graph. The second expectation serves as a regularizer to ensure different eigenvectors are orthogonal. Using this objective, we can train the Laplacian embedding function, f.

3.2 Option Learning

Option learning refers to when the agent is tasked with developing skills based upon the representation it has developed from exploration (e.g. the Laplacian model f). We can simply create these skills formalized as policies in the RL setting where the reward is the "eigen-purpose" reward as described by Machado & Bowling (2016). Every dimension of the Laplacian representation corresponds to an eigenvector which then corresponds to 1 skill. The goal of the i^{th} skill's policy is to ascend or descend the Laplacian representation's i^{th} dimension output which can be described by Equation 3.2

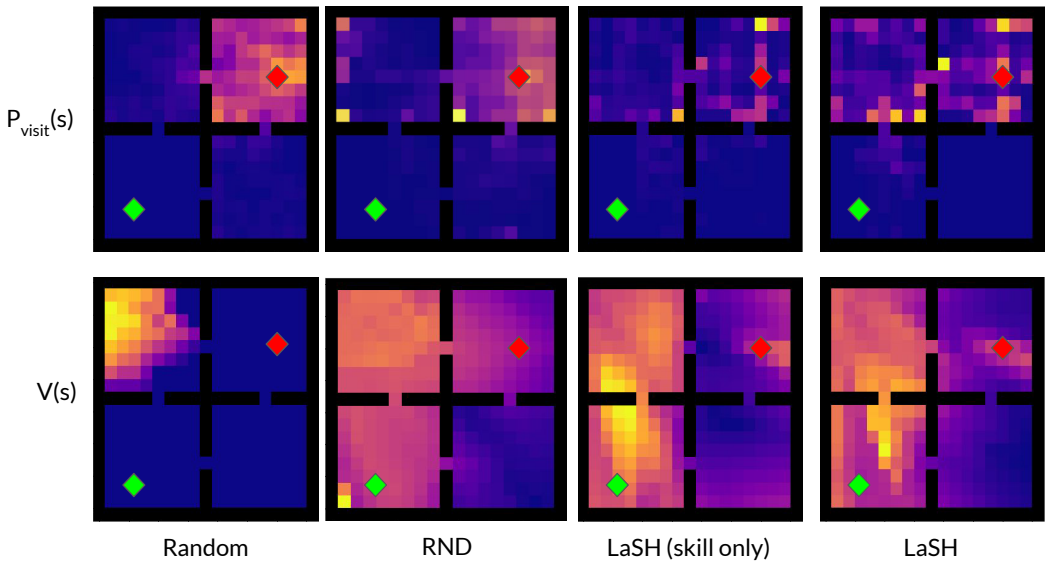

Figure 3: **Top:** Visitation histogram of states starting at the red with the goal displayed in green. LaSH methods use our greedy exploration while others are random or RND induced. **Bottom:** The value function when trained via DQN to reach the goal with sparse reward is shown.

$$r_{i,t} = f_i(s_{t+1}) - f_i(s_t)$$

4 Laplacian Skill Hierarchies (LaSH)

We now overview how to build and use a set of options for planning and exploration via LaSH. The algorithm first runs in an unsupervised setting to discover skills and explore the environment, then is task-supervised via a reward function and operates in a standard SMDP. During the unsupervised phase, the algorithm explores the environment with its current skill library and learns an intrinsic reward. Using the skill library as the action space, it then learns new options via RL on this intrinsic reward, updates the skill library, and returns to the exploration step to repeat. The proposed exploration depends on the skills learned thus far, and as a result, changes as we learn more options. As a result, the Laplacians learned are induced by continually more complex policies. Our exploration is structured and detailed in Algorithm 2 whereby we maximize the expected value of intrinsic reward. In practice, we choose the number of times we re-learn the Laplacian and the corresponding new options depending on the complexity of the environment.

Once these options are learned, we then construct an SMDP and run RL on an extrinsic task-specific reward while having the action space as the set addition between $\mathcal{A}$ and the learned options. We show the algorithm pseudocode in Algorithm 1 and the full algorithm in the Appendix 3.

Crucially, our algorithm works iteratively, continually creating new skills so knowledge is not overwritten over time but is rather saved for the agent to re-use. This allows LaSH to operate as a module that can generate a set of skills at different levels of abstraction which can be combined in new ways to solve a variety of new tasks. We also structure our exploration by greedily maximizing value from chaining together skills, since we do not have access to extrinsic reward as DCEO may. By composing skills to maximally cover the environment during exploration, we influence the next Laplacian representation to find new temporal correlation patterns that can introduce new skills.

5 EXPERIMENTS

Method	Success Rate	Success Steps
LaSH 3-level	**100** %	52980 $\pm$ 24381
LaSH 1-level	60 %	39950 $\pm$ 21813
DCEO	60 %	**3820** $\pm$ **2625**
Primitives	0 %	N/A

Figure 4: We show the rate of success and the steps it takes to succeed in opening the door for each of the methods.

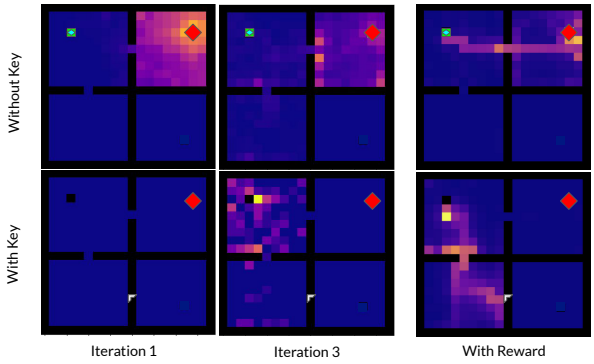

Figure 5: The visitation of random walks with the learned skill-augmented action space after 1 iteration and 3 iterations. We also show visitation induced by the learned policy after optimizing for extrinsic reward with LaSH. The visitations are separated into two rows for when the agent has the key (in green).

5.1 PLANNING

To assess the capabilities of LaSH as a module to assist in planning to reach a goal, we use a simple maze environment in the tabular setting with the coordinates as the state. We use the Minigrid (Chevalier-Boisvert et al., 2023) environment to create a 4-room environment with small doors connecting the 4 chambers. We test by running DQN on different action spaces; specifically, the primitive action space and the LaSH augmented action space after pretraining. We also experiment with just the options from LaSH and RND with primitives as well. Episodic rewards are shown Figure 2 along with Laplacian representations overlaid on the maze. This experiment shows that the augmented action space learned from LaSH significantly helps the learning process when given a downstream task. We display the visitation distribution with our greedy skill chaining as well as the final learned value functions in Figure 3.

5.2 EXPLORATION

We also experiment with more complex longer-horizon tasks that require multiple LaSH levels. Specifically, we assess whether LaSH assists in developing skills that can be composed to explore and create plans. We test this by measuring whether random walks with learned skills from LaSH actually result in completing a long-horizon task and how many training steps are required to do so. In our case, the task is to open the door and the only way to do it is to interact with the door while holding a key, which must be picked up in advance. We use the success rate and number of steps taken to reach the sparse goal to assess the planning and exploration capabilities for the given algorithm.

With prioritized replay (Schaul et al., 2016), standard techniques can eventually result in backing up rewards from explored goals for a usable policy so we don't consider episodic return as a metric. The results are shown in Table 4 for LaSH with 1 level and 3 levels and 12 options total. The first level of options often resulted in room-to-room navigation while later levels had object interaction. We also show results for random primitive noise and DCEO, which learns 1 level of skills jointly with the Laplacian representation and crucially has *extrinsic reward access*. Even without any extrinsic reward access, LaSH more consistently achieves the goal than all other methods. Although, it can be seen that DCEO achieves the reward very fast which corresponds to the runs where the algorithm ran into the reward in the pretraining phase, encouraging it to seek out the goal more and move it into the skills directly, unlike LaSH. Regardless, LaSH would remain more efficient if we wanted to re-use it for solving multiple other tasks because DCEO would need to be re-trained for each reward.

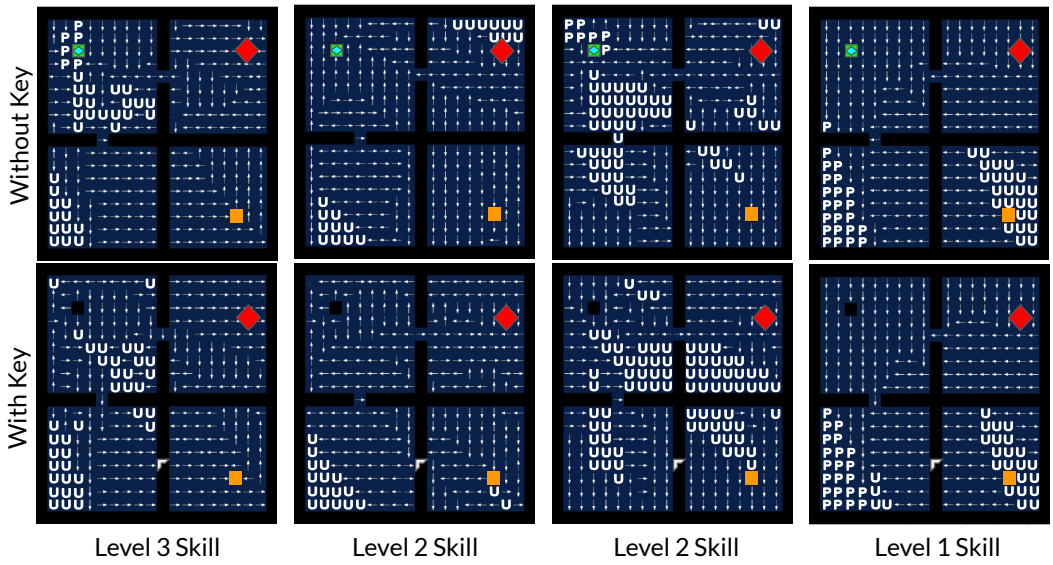

Figure 6: Every column in this figure is a selected skill with arrows or letters (P for pick up and U for unlock) as actions in a given state. The start is in red, a goal is in orange behind a locked door, and the key is in green.

Visualizations of skill policies at multiple levels are shown in Figure 6. Considering that these skills are run with some epsilon noise, we can roughly determine what they accomplish. We see that low level skills have fairly uniform actions such as constantly moving left or right to traverse inside rooms. As the LaSH iterations increase, we see more complex behaviors that are roughly combinations of uniform behaviors in level-one. In fact, one level-two skill navigates to the key, and the other executes a pickup action when close to the key. This is more complex than intra-room movement and shows inter-room movement as well as interaction with objects. Finally, in level-3, we again see composition of behaviors from earlier levels whereby the agent can roughly move to the key, pick it up, and move to the door. It is important to note that these skills are discovered without any extrinsic reward encouraging travel to the door or goal.

6 Discussion

We present a method, LaSH , which uses a feedback loop of skill learning and skill-chained exploration to build a library of skills. Importantly, the hierarchy formed is well-suited for gradually exploring the environment and preserving important options for downstream planning. Furthermore, the greedy state-covering skill-chaining policy for learning the next level Laplacian proves useful for incorporating a skill-composition inductive bias. Our method is capable of learning hierarchical skills for planning, and exploring environments well especially in the sparse-reward case. Specifically, we compared it against similar reward-accessing methods and novelty methods for completing sparse-reward tasks which it outperforms.

7 Future Work

There are limitations associated with this system, namely the high cost of using a full DQN model per skill. This could be ameliorated with a large latent-variable model to encode shared information between skills. Furthermore, exploring how skills can be fine-tuned to extrinsic rewards was not explored due to the potential instabilities of learning a policy on options while learning the options simultaneously. But, with an extrinsic reward, the skills in Figure 6 have the potential to be less noisy and accomplish reaching the goal better. There are also many scalability studies to conduct in order to move to continuous action spaces and high-dimension pixel input – we expect this to have potential for more complex tasks as shown with DCEO and the capability of the graph-drawing Laplacian objective to scale to visual tasks.

REFERENCES

Alexander, Vezhnevets, Volodymyr Mnih, John Agapiou, Simon Osindero, Alex Graves, Oriol Vinyals, and Koray Kavukcuoglu. Strategic attentive writer for learning macro-actions, 2016.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture, 2016.

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

Akhil Bagaria, Jason Crowley, Jing Wei Nicholas Lim, and George Konidaris. Skill discovery for exploration and planning using deep skill graphs. In *4th Lifelong Machine Learning Workshop at ICML 2020*, 2020. URL https://openreview.net/forum?id=-mvAo5hWNp.

Aijun Bai and Stuart Russell. Efficient reinforcement learning with hierarchies of machines by leveraging internal transitions. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pp. 1418–1424, 2017. doi: 10.24963/ijcai.2017/196. URL https://doi.org/10.24963/ijcai.2017/196.

Amitay Bar, Ronen Talmon, and Ron Meir. Option discovery in the absence of rewards with manifold analysis, 2020.

Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation, 2016.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation, 2018.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo de Lazcano, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *CoRR*, abs/2306.13831, 2023.

Özgür Şimşek and Andrew Barto. Skill characterization based on betweenness. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou (eds.), *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2008. URL https://proceedings.neurips.cc/paper_files/paper/2008/file/934815ad542a4a7c5e8a2dfa04fea9f5-Paper.pdf.

Özgür Şimşek, Alicia P. Wolfe, and Andrew G. Barto. Identifying useful subgoals in reinforcement learning by local graph partitioning. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pp. 816–823, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595931805. doi: 10.1145/1102351.1102454. URL https://doi.org/10.1145/1102351.1102454.

Will Dabney, Georg Ostrovski, and André Barreto. Temporally-extended $\epsilon -$ $greedyexploration$, 2020.

Christian Daniel, Gerhard Neumann, Oliver Kroemer, and Jan Peters. Hierarchical relative entropy policy search. *Journal of Machine Learning Research*, 17(93):1–50, 2016. URL http://jmlr.org/papers/v17/15-188.html.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5:613–624, 1993. URL https://api.semanticscholar.org/CorpusID:12559116.

Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. In S. Hanson, J. Cowan, and C. Giles (eds.), *Advances in Neural Information Processing Systems*, volume 5. Morgan-Kaufmann, 1992. URL https://proceedings.neurips.cc/paper_files/paper/1992/file/d14220ee66aeec73c49038385428ec4c-Paper.pdf.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning diverse skills without a reward function. 2018.

Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning, 2017.

Roy Fox, Sanjay Krishnan, Ion Stoica, and Ken Goldberg. Multi-level discovery of deep options, 2017.

Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. Meta learning shared hierarchies, 2017.

Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control, 2016.

Jean Harb, Pierre-Luc Bacon, Martin Klissarov, and Doina Precup. When waiting is not an option : Learning options with a deliberation cost, 2017.

Glenn A. Iba. A heuristic approach to the discovery of macro-operators. *Mach. Learn.*, 3(4): 285–317, mar 1989. ISSN 0885-6125. doi: 10.1023/A:1022693717366. URL https://doi.org/10.1023/A:1022693717366.

Yuu Jinnai, David Abel, D Ellis Hershkowitz, Michael Littman, and George Konidaris. Finding options that minimize planning time, 2019a.

Yuu Jinnai, Jee Won Park, David Abel, and George Konidaris. Discovering options for exploration by minimizing cover time, 2019b.

Yuu Jinnai, Jee Won Park, Marlos C. Machado, and George Konidaris. Exploration in reinforcement learning with deep covering options. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SkeIyaVtwB.

Jeff Johns and Sridhar Mahadevan. Constructing basis functions from directed graphs for value function approximation. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pp. 385–392, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595937933. doi: 10.1145/1273496.1273545. URL https://doi.org/10.1145/1273496.1273545.

Khimya Khetarpal, Martin Klissarov, Maxime Chevalier-Boisvert, Pierre-Luc Bacon, and Doina Precup. Options of interest: Temporal abstraction with interest functions, 2020.

Martin Klissarov and Marlos C. Machado. Deep laplacian-based options for temporally-extended exploration, 2023.

Alexander S. Klyubin, Daniel Polani, and Chrystopher L. Nehaniv. All else being equal be empowered. In Mathieu S. Capcarrère, Alex A. Freitas, Peter J. Bentley, Colin G. Johnson, and Jon Timmis (eds.), *Advances in Artificial Life*, pp. 744–753, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-31816-3.

Varun Raj Kompella, Marijn Stollenga, Matthew Luciw, and Juergen Schmidhuber. Continual curiosity-driven skill acquisition from high-dimensional video inputs for humanoid robots. *Artificial Intelligence*, 247:313–335, 2017. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2015.02.001. URL https://www.sciencedirect.com/science/article/pii/S000437021500017X. Special Issue on AI and Robotics.

Sanjay Krishnan, Roy Fox, Ion Stoica, and Ken Goldberg. Ddco: Discovery of deep continuous options for robot learning from demonstrations, 2017.

Tejas D. Kulkarni, Karthik R. Narasimhan, Ardavan Saeedi, and Joshua B. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation, 2016a.

Tejas D. Kulkarni, Ardavan Saeedi, Simanta Gautam, and Samuel J. Gershman. Deep successor reinforcement learning, 2016b.

Michael Laskin, Hao Liu, Xue Bin Peng, Denis Yarats, Aravind Rajeswaran, and Pieter Abbeel. Cic: Contrastive intrinsic control for unsupervised skill discovery, 2022.

Steven M. LaValle. Rapidly-exploring random trees : a new tool for path planning. *The annual research report*, 1998. URL https://api.semanticscholar.org/CorpusID:14744621.

Marlos C. Machado and Michael Bowling. Learning purposeful behaviour in the absence of rewards, 2016.

Marlos C. Machado, Marc G. Bellemare, and Michael Bowling. A laplacian framework for option discovery in reinforcement learning, 2017.

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

Marlos C. Machado, Andre Barreto, Doina Precup, and Michael Bowling. Temporal abstraction in reinforcement learning with the successor representation, 2023.

Sridhar Mahadevan. Proto-value functions: Developmental reinforcement learning. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pp. 553–560, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595931805. doi: 10.1145/1102351.1102421. URL https://doi.org/10.1145/1102351.1102421.

Sridhar Mahadevan and Mauro Maggioni. Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. *Journal of Machine Learning Research*, 8(74):2169–2231, 2007a. URL http://jmlr.org/papers/v8/mahadevan07a.html.

Sridhar Mahadevan and Mauro Maggioni. Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. *Journal of Machine Learning Research*, 8:2169–2231, 10 2007b.

Amy McGovern and Andrew G. Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pp. 361–368, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1558607781.

Ishai Menache, Shie Mannor, and Nahum Shimkin. Q-cut—dynamic discovery of sub-goals in reinforcement learning. In Tapio Elomaa, Heikki Mannila, and Hannu Toivonen (eds.), *Machine Learning: ECML 2002*, pp. 295–306, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-36755-0.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning, 2018.

Ronald Parr and Stuart Russell. Reinforcement learning with hierarchies of machines. In M. Jordan, M. Kearns, and S. Solla (eds.), *Advances in Neural Information Processing Systems*, volume 10. MIT Press, 1997. URL https://proceedings.neurips.cc/paper_files/paper/1997/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction, 2017.

Karl Pertsch, Youngwoon Lee, and Joseph J. Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on Robot Learning (CoRL)*, 2020.

Doina Precup. *Temporal abstraction in reinforcement learning*. PhD thesis, 2000.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay, 2016.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181–211, 1999. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(99)00052-1. URL https://www.sciencedirect.com/science/article/pii/S0004370299000521.

Chen Tessler, Shahar Givony, Tom Zahavy, Daniel J. Mankowitz, and Shie Mannor. A deep hierarchical approach to lifelong learning in minecraft, 2016.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning, 2017.

Kaixin Wang, Kuangqi Zhou, Qixin Zhang, Jie Shao, Bryan Hooi, and Jiashi Feng. Towards better laplacian representation in reinforcement learning with generalized graph drawing. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11003–11012. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/wang21ae.html.

David Warde-Farley, Tom Van de Wiele, Tejas Kulkarni, Catalin Ionescu, Steven Hansen, and Volodymyr Mnih. Unsupervised control through non-parametric discriminative rewards, 2018.

Yifan Wu, George Tucker, and Ofir Nachum. The laplacian in rl: Learning representations with efficient approximations, 2018.

Mengda Xu, Manuela Veloso, and Shuran Song. ASPire: Adaptive skill priors for reinforcement learning. In *Thirty-Sixth Conference on Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=sr0289wAUa.

A APPENDIX

Algorithm 3 LaSH Skill Discovery

skillDAG $= \{\}$
Append primitive actions to DAG
for $level \leq M$ **do**
  currentSkills $= \{\}$
  $\mathcal{D} \leftarrow$ emptyBuffer()
  for $exploreStep \leq E_{level}$ **do**
    if skill not yet chosen or running skill has ended **then**
      $\pi_o \leftarrow$ explorationPolicy(skillDAG) ▷ Sample a skill for exploration
    end if
    $a \sim \pi_o$ ▷ Run the selected skill
    Execute a in state s and observe s'
    Update $\mathcal{D}$ with transition (s, a, s')
    Train Laplacian Representation on $\mathcal{D}$ via 3
    Reward label $\mathcal{D}$ using eigen-purpose objective
  end for
  Initialize N_{level} skill policies into currentSkills
  for $learnStep \leq S_{level}$ **do**
    if skill not yet chosen or running skill has ended **then**
      if $\mathcal{N}(0,1) < \epsilon$ **then**
        $\pi_o \leftarrow$ explorationPolicy(SkillDAG + currentSkills)
      else
        $\pi_o \leftarrow$ random element from currentSkills
      end if
    end if
    $a \sim \pi_o$ ▷ Run the selected skill
    Execute a in state s and observe s' as well as intrinsic reward
    Update $\mathcal{D}$ with transition (s, a, s', r)
    Update Options via 3.2
  end for
  Append learned currentSkills to skillDAG
end for