Breadth-First Exploration on Adaptive Grid for Reinforcement Learning

Youngsik Yoon [1] **Gangbok Lee** [2] **Sungsoo Ahn** [1 2] **Jungseul Ok** [1 2]

Abstract

Graph-based planners have gained significant attention for goal-conditioned reinforcement learning (RL), where they construct a graph consisting of confident transitions between *subgoals* as edges and run shortest path algorithms to exploit the confident edges. Meanwhile, identifying and avoiding unattainable transitions are also crucial yet overlooked by the previous graph-based planners, leading to wasting an excessive number of attempts at unattainable subgoals. To address this oversight, we propose a graph construction method that efficiently manages all the achieved and unattained subgoals on a grid graph adaptively discretizing the goal space. This enables a breadth-first exploration strategy, grounded in the local adaptive grid refinement, that prioritizes broad probing of subgoals on a coarse grid over meticulous one on a dense grid. We conducted a theoretical analysis and demonstrated the effectiveness of our approach through empirical evidence, showing that only BEAG succeeds in complex environments under the proposed fixed-goal setting. [1]

1. Introduction

Many real-world sequential decision-making problems can be framed as the task of targeting a given goal, e.g., the navigation of walking robots (Schulman et al., 2015; Nachum et al., 2018; Haarnoja et al., 2019) and the manipulation of objects using robotic arms (Levine et al., 2016; Andrychowicz et al., 2017; Rajeswaran et al., 2018). Goal-conditioned reinforcement learning (RL) (Schaul et al., 2015; Nasiriany et al., 2019) aims to solve these problems using a goal-conditioned policy designed to maximize the return with respect to the target goal. This offers a versatile policy for a variety of distinct problems, described by corresponding goals, whereas other RL frameworks often require separate policies for different tasks. Besides the inherent versatility, it also enables the hierarchical RL (Vezhnevets et al., 2017; Nachum et al., 2018; Zhang et al., 2020) which decomposes a daunting long-horizon goal into a series of manageable short-horizon subgoals so that the agent can exploit more confident and learnable transitions between subgoals than the direct transition to the ultimate goal.

Recent advancements in goal-conditioned RL have witnessed the emergence of *graph-based RL* equipped with graph-based planners for this subgoal-based decomposition (Eysenbach et al., 2019; Huang et al., 2019; Bagaria et al., 2021; Kim et al., 2021; Lee et al., 2022). At a high level, these planners construct a graph that encapsulates confident transitions between subgoals as edges. They employ shortest-path algorithms to leverage the confident edges. However, there is a significant caveat: these graphs comprise only subgoals in the replay buffer, which the agent has achieved in prior, i.e., the previous planners disregard unattained subgoals consisting of failed ones in history and unexplored ones over the goal space. The oversight of such unattained subgoals leads to wasteful expenditure of samples to repeatedly attempt similar unexplored or even impossible subgoals rather than to explore novel ones, as exemplified in Figure 1.

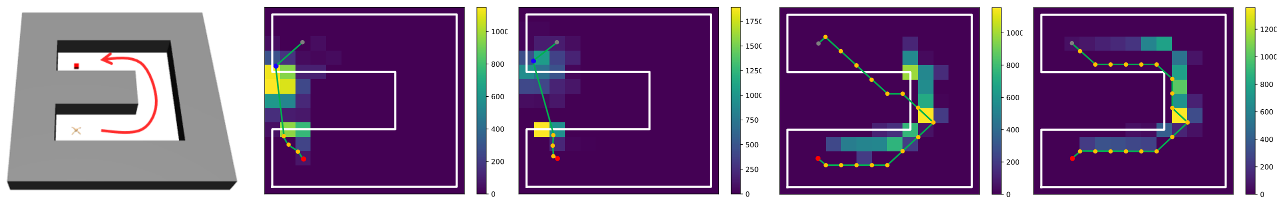
In response to this challenge, we propose **B**readth-first **E**xploration on **A**daptive **G**rid (BEAG) for graph-based RL. Our key idea is to manage both achieved and unattained subgoals on a grid graph, adaptively discretizing the goal space during the training process. Our subgoal management assesses the achievability of all the subgoals on the grid, including unattained ones, and then probes the subgoals in a planned order. This systemically prevents consecutive attempts to unachievable subgoals, whereas the previous method, e.g., (Lee et al., 2022), expend attempts to randomly ransack unexplored subgoals, as shown in Figure 1.

Specifically, we devise a breadth-first exploration strategy grounded on local adaptive grid refinement. This strategy first explores subgoals on a coarse grid and then refines the grid selectively around the local of unattained subgoals. The benefits are twofold: (i) it prioritizes broad probing on a coarse grid over extensive searching on a dense grid;

[1]Department of CSE, POSTECH, Pohang, Republic of Korea [2]Graduate School of AI, POSTECH, Pohang, Republic of Korea. Correspondence to: Jungseul Ok <jungseul@postech.ac.kr>.

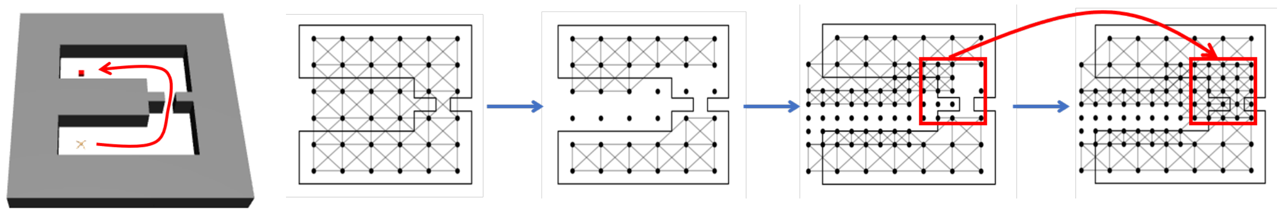
Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

[1]https://github.com/ml-postech/BEAG



(a) *U-maze* task (b) DHRL (Lee et al., 2022) (c) BEAG (ours)

Figure 1: **Illustration of breadth-first exploration.** We compare the subgoal exploration strategies of DHRL (state-of-the-art) and BEAG (ours) for *U-maze* task, depicted in Figure 1a. In Figure 1b and 1c, each plot summarizes the statistics on the attempted subgoals at 5-th and 10-th training epochs, corresponding to 174K and 219K environment steps, respectively. DHRL expends a substantial number of attempts on impossible subgoals (on and over the wall), whereas BEAG spends virtually zero attempts on them. This demonstrates the efficiency of BEAG conducting the breadth-first exploration.



(a) *bottleneck-maze* task (b) The proposed procedure of adaptive grid refinement

Figure 2: **Local adaptive grid refinement.** In Bottleneck-maze task of Figure 2a, we visualize how BEAG adaptively refines the grid of subgoals over training epochs (0, 8, 24, 25), corresponding to environment steps (174K, 246K, 390K, 399K) respectively. As shown in Figure 2b, our approach adaptively refines around informative subgoals, previously unattainable and thus disconnected.

and (ii) adaptively refines the grid around more demanding parts, as elucidated in Figure 2. Our extensive experiments demonstrate the efficacy of our method, in which BEAG identifies adaptive grids tailored to various environments requiring heterogeneous grid resolutions, both at the level of individual environments and within different parts of a single environment.

Our main contributions are summarized as follows:

- We show that the previous graph-based planners overlook unattained subgoals in their graphs, and thus frequently misguide the agent to wastefully attempt to impossible subgoals.
- To address this oversight, we introduce a grid-based RL, called BEAG, which efficiently identifies both achieved and unattained subgoals through the breadth-first exploration based on the local adaptive grid refinement over the goal space.
- We provide a geometrical analysis showing the adaptability and efficiency in various environments with differing grid resolution requirements, and an extensive set of experiments in which our method remarkably outperforms state-of-the-art methods.

2. Related works

Graph-based RL and graph management Graph-based RL has emerged as a promising framework for solving complex tasks (Huang et al., 2019; Eysenbach et al., 2019; Kim et al., 2021; Lee et al., 2022), where deep RL agents plan a path of manageable subgoals from a challenging goal while leveraging graph algorithms such as shortest path algorithms on a subgoal graph. Despite their remarkable advancements, they discard unattained subgoals and construct graphs of achieved ones, randomly sampled from the replay buffer, inheriting the practice originated by (Huang et al., 2019; Eysenbach et al., 2019). This restricted and random construction of graphs inevitably omits records of unattainable subgoals, which are beneficial to avoid them in the future. Consequently, this omission can lead deep RL agents to fruitlessly pursue impossible subgoals, thereby diluting the benefits offered by graph algorithms. To address this issue, we introduce a grid-based method to embrace both attained and unattained subgoals on a grid, systematically exploring them. Specifically, we introduce the breadth-first exploration on a local adaptive grid refinement, where the refinement mechanism is analogous to the principle in (Berger & Oliger, 1984).

Structured exploration in RL For efficient RL, exploration strategies have been structured to exploit the prior knowledge on the environment. This body of work encompasses both theoretical (Combes et al., 2017; Ok et al., 2018) and empirical (Pathak et al., 2017; Tam et al., 2022) studies. In hierarchical RL, the prior methods to expand the set of confident subgoals (Zhang et al., 2020; Kim et al., 2021; Lee et al., 2022) are structured under the assumption that more distant goals are inherently more challenging. They prioritize subgoals adjacent to previously identified confident ones. However, they overlook another valuable piece of prior knowledge that the achievabilities of similar goals are likely to be similar. Hence, they are prone to expend similar attempts around an impossible subgoal even after experiencing consecutive failures, rather than exploring more broadly for novel attempts. In contrast, our proposed breadth-first exploration strategy takes into account the prior knowledge of the similarity, thereby promoting a more efficient exploration. While similar approaches to broaden exploration ranges have been explored in various RL methods, e.g., (Pathak et al., 2017; Tam et al., 2022), our work is, to the best of our knowledge, the first one tailored for goal exploration. In addition, our adaptive refinement provides further improvements.

Finally, we underscore that a number of the previous methods (Eysenbach et al., 2019; Huang et al., 2019; Zhang et al., 2021; Gieselmann & Pokorny, 2021; Kim et al., 2023) have bypassed the subgoal exploration problems by randomly sampling *initial state or goal over the feasible space*, where such a random sampling can provide direct information on the area of achievable subgoals. In our experiment, when we fix the initial state and goal, only our BEAG can quickly identify achievable subgoals and learn a set of complex tasks, whereas the previous methods show extremely poor performance.

3. Preliminaries

Problem formulation We consider a standard goal-conditioned Markov decision process (Schaul et al., 2015; Andrychowicz et al., 2017) $\langle \mathcal{S}, \mathcal{G}, \mathcal{A}, p, r, \gamma, H \rangle$ consisting of state space $\mathcal{S}$, goal space $\mathcal{G} \subset \mathbb{R}^K$, action space $\mathcal{A}$, transition dynamics $p(s'|s,a)$ from state s to state s' given action a, reward function $r(s,a,s',g)$ for transition (s,a,s') with respect to goal g, discount factor γ, and horizon H. Then, the agent's policy $\pi(a|s,g)$, conditioned on goal g, is trained to minimize the time to achieve the goal within a prescribed tolerance $\delta_r > 0$, such that $\|\phi(s') - g\|_2 \leq \delta_r$, where $\phi : \mathcal{S} \mapsto \mathbb{R}^K$ is a goal-feature function. To be specific, we consider a sparse reward function defined as:

$$r(s,a,s',g) = -\mathbb{1}\{\|\phi(s') - g\|_2 > \delta_r\} \ , \tag{1}$$

which gives 0 only when the agent has successfully reached goal g and -1 otherwise. Then, the expected number of time steps to reaching state s to goal g given policy π can be approximated by a distance function $d(s,g)$:

$$d(s,g) := \log_\gamma(1 + (1-\gamma)V_\pi(s|g)) \ , \tag{2}$$

where $V_\pi(s|g)$ is the value function given goal g (Lee et al., 2022).

Graph-based RL A general framework of graph-based RL (Eysenbach et al., 2019; Kim et al., 2021; 2023; Lee et al., 2022) maintains a directed graph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} \subset \mathcal{G}$ is a set of nodes and $\mathcal{E}$ is a set of edges. Each edge (u,v) is associated with a weight function $w(u,v)$ estimating the (approximated) time $d(s,v)$ to reach v from some state s such that $\phi(s) = u$. For the agent at state s given an ultimate goal g, a graph-based planner runs a shortest path algorithm such as Dijkstra's algorithm (Dijkstra, 1959) to find a path $P = (\phi(s), v_1, v_2, ..., v_n, g)$ such that $v_1, ..., v_n \in \mathcal{V}$. Then, the agent first attempts at subgoal v_1 with $\pi(\cdot|\cdot, v_1)$, rather than the ultimate goal g with $\pi(\cdot|\cdot, g)$, and proceed along the path P. Such a decomposed guidance can accelerate the time to reach goal g as targeting each subgoal v_i along path P is more confident than attempting the ultimate goal g at once.

Previous methods for graph construction To include as many confident edges as possible in the graph, graph-based RL methods commonly adopt the graph construction approach, called Search on the Replay Buffer (SoRB) (Eysenbach et al., 2019). It begins with the node set $\mathcal{V}$ by sampling visited state s's uniformly at random from the replay buffer and adding $\phi(s)$'s as a node. Then, the edge set $\mathcal{E}$ is constructed by connecting all nodes in a neighbor of a certain distance τ, i.e., for each $u, v \in \mathcal{V}$, edge (u,v) is assigned weight $w(u,v)$:

$$w(u,v) = \begin{cases} d(s,v) & \text{if } d(s,v) \leq \tau \\ \infty & \text{otherwise} \end{cases} \ , \tag{3}$$

where s is the sampled state associated with $u = \phi(s)$, and we denote disconnected subgoal pairs by assigning infinite weights.

Limitation of previous graph-planning Given the aforementioned graph $\mathcal{H}$, for goal g and the agent at state s, the graph-based planner constructs an augmented graph $\mathcal{H}' = (\mathcal{V}', \mathcal{E}')$ by adding $\phi(s), g$ to the aforementioned graph $\mathcal{H}$ and run the shortest path algorithm to obtain path $P = (\phi(s), v_1, ..., v_n, g)$ such that $v_1, ..., v_n \in \mathcal{V}$. To maximize the confidence of the first and last edges, the augmented graph applies the same edge cut-off of threshold τ in (3) if possible. However, every distance from subgoals in $\mathcal{V}$ to goal g (or from $\phi(s)$ to subgoals in $\mathcal{V}$) can be cut off by τ. In this case, the shortest path is computed by just connecting g (or $\phi(s)$) to the subgoal with the smallest distance. When goal g is impossible to achieve in a short time, such a direct connection is frequent but problematic as it can hinder the planner from exploring detour paths and lead to wasteful attempts. Indeed, as depicted in Figure 1, the previous state-of-the-art, DHRL (Lee et al., 2022), plans paths, exemplified in orange color, including an edge over the wall and lack of path planning for proper exploration mainly due to the absence of unattained subgoals in the graph $\mathcal{H}$. To address this limitation, we propose a grid-based method described in Section 4.

4. Proposed method: BEAG

To endue graph-planners with systemic exploration, we propose **B**readth-first **E**xploration on **A**daptive **G**rid (BEAG), which constructs and maintains a grid graph including both attained and unattained subgoals. In what follows, we describe three key procedures of BEAG: (i) initializing grid with extended distance function; (ii) identifying unattainable subgoals; and (iii) adaptively refining grid. We also visually illustrate them with an example in Figure 2. For other details, we provide the pseudo-codes of BEAG in Appendix A and our implementation in the supplementary material.

Initializing grid with extended distance function Our graph management begins with an initial grid $\mathcal{H}$ covering the entire feasible goal space, even including impossible subgoals (e.g., ones inside walls of the maze in Figure 2). For ease of presentation, we describe an instance of grid initialization given lower bound L and upper bound U of goal space $\mathcal{G} \subset \mathbb{R}^K$ so that we set node set $\mathcal{V} = \left\{L, L+\delta_0, \dots, L+\lfloor\frac{U-L}{\delta_0}\rfloor\delta_0\right\}^K$ with an initial interval $\delta_0 > 0$, and edge set $\mathcal{E}$ consisting of (u,v)'s such that $\|u-v\|_\infty \le \delta_0$ for $u, v \in \mathcal{V}$. We note that the node set of the grid is formed independently of the replay buffer. Hence, extending the distance function d from a visited state to a goal in (2), we devise a distance function $\tilde{d}(u,v)$ from u to v as follows:

$$\tilde{d}(u,v) = \log_\gamma(1 + (1-\gamma)\tilde{V}_\pi(u|v)), \quad (4)$$

where $u, v \in \mathcal{G}$ and $\tilde{V}_\pi$ is an estimate of the value function at a set of state s such that $\phi(s) = u$ given goal v. By estimating the distance between nodes in the goal space in (4), it can be possible to estimate the distance between visited nodes in situations where transformation from goal to space is nontrivial (e.g., Reacher3D-wall environment in Figure 3h). This construction of grid graph $\mathcal{H}$ uniformly cover the entire goal space so that each goal $g \in \mathcal{G}$ has nearby subgoals on the grid.

Identifying unattainable subgoals Assuming that the length of edges between attainable subgoals remains bounded by the Euclidean distance with a Lipschitz continuity, we can deduce that nodes failing to reach within specified timesteps are deemed unattainable. However, marking subgoals as unattainable after a single failure may cause problems. For instance, failures generated from the unstable policy during the early stage of training can occur. For this, we employ two hyperparameters τ_t, τ_n which serve as thresholds for the failure condition and count, respectively. To be specific, in our grid, weight $w(u,v)$ for $(u,v) \in \mathcal{E}$ is defined as follows:

$$w(u,v) = \begin{cases} \tilde{d}(u,v) & \text{if } n_v^s > 0 \text{ or } n_v^a \le \tau_n \\ \infty & \text{otherwise} \end{cases}, \quad (5)$$

where n_v^s is the number of successes to actually reach v and n_v^a is the number of attempts v as the next node, i.e. n_v^a increases with each success or failure within a timestep of τ_t. By modifying the weight of edges, the planner can find the shortest path without containing marked unattainable nodes as illustrated in Figure 1.

Adaptive grid refinement Determining the initial interval δ_0 used in initializing the grid graph is a critical hyperparameter that significantly influences the performance of the planner. If δ_0 is set small, it results in an exponential number of subgoals in the graph, leading to high computational costs for graph management and path planning. On the contrary, if δ_0 is set large, it might fail to generate paths consisting of attainable nodes, depending on the environment. To ensure the algorithm operates robustly even in such situations, we propose an adaptive grid refinement technique that reduces the interval of the grid, creating a dense graph. To be specific, when the planner fails to plan a path, we choose one subgoal that requires further exploration. To prioritize exploration in regions close to unattainable nodes, which are deemed more valuable to explore than already successful regions, we opt to refine the unattainable node. To enhance exploration, we select one of the unattainable subgoals, prioritizing the less selected subgoals for breadth-first exploration. Specifically, to refinement, we add to $\mathcal{V}$ all subgoals v near the selected subgoal $u = (u_1, u_2, \dots, u_K)$:

$$v \in \bigtimes_{k=1}^{K} \left\{u_k - \delta, u_k - \frac{\delta}{2}, u_k, u_k + \frac{\delta}{2}, u_k + \delta\right\}, \quad (6)$$

where $\times$ is the Cartesian product of sets and δ is the current interval around u, which is initialized as δ_0 and halved with each refinement trial. Subsequently, for each v, we initialize both n_v^a and n_v^s as 0 and update $\mathcal{E}$ by connecting edges with nodes in $\mathcal{V}$ where L_∞ distance between them is equal to $\frac{\delta}{2}$.

5. Theoretical analysis of BEAG

In this section, we present a theoretical formalization of the advantages of our algorithm by analyzing how the path found by our algorithm closely approximates the ground-truth path in the K-dimensional goal space. Precisely, we prove that (i) *any point in the output path of our algorithm is guaranteed to be near the ground-truth path* and (ii) *the length of the output path is bounded by a constant factor of the length of the ground-truth path.*

To begin with, we establish definitions to indicate the validity of paths based on the existence of actions to follow the given path.

Definition 5.1. A goal $g \in \mathcal{G}$ is reachable from state s if there exists a sequence of actions $a_1, \dots, a_T$ with the terminal state s_T satisfying $\phi(s_T) = g$.

Definition 5.2. Consider a continuous path $f : [0, T] \to \mathcal{G}$ as a mapping from time to a compact goal space $\mathcal{G}$. An ϵ-path is a path when every goal is reachable if it is an element of the ϵ-covering $C(f, \epsilon)$ defined as:

$$C(f,\epsilon) = \bigcup_{t\in[0,1]} \{g : \|g - f(t)\|_\infty \le \epsilon\}.$$

Next, since the ground-truth path is continuous while our algorithm outputs a discrete path, we define a mapping from a discrete path given by our algorithm to a continuous path.

Definition 5.3. Given a grid graph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, the continuous grid path $P = (g_0, \dots, g_n)$ is a continuous path $f(t)$ satisfying:

$$f\left(\frac{i+\lambda}{n}T\right) = (1-\lambda)g_i + \lambda g_{i+1}, \quad \lambda \in [0,1],$$

for $i = 0, \dots, n-1$. The sequence $g_1, \dots, g_{n-1}$ forms a (discrete) path in the grid graph $\mathcal{H}$. Finally, the starting goal g_0 and the final goal g_n satisfies

$$\|g_0 - g_1\|_\infty \le \delta_0, \quad \|g_{n-1} - g_n\|_\infty \le \delta_0,$$

for the initial interval δ_0 of the grid graph $\mathcal{H}$.

We now provide our main theoretical result of which proof is presented in Appendix B.

Theorem 5.4. *Consider a compact goal space $\mathcal{G}$ and a ϵ-path $f(\cdot)$. (i) Our algorithm outputs an adaptive refined grid graph $\mathcal{H}$ with initial interval δ_0 and minimum interval $\delta \le \epsilon$ with a grid path $P = (g_0, \dots, g_n)$ that is the subset of the ϵ-path cover $C(f, \epsilon)$. (ii) The length of the grid path P is at most $(\lceil \frac{l}{\delta_0} \rceil + 1)\delta_0\sqrt{K}$ where l is the length of the ϵ-path $f(\cdot)$.*

The first statement implies that given a pair of source and destination points connected by a continuous path f with sufficient distance ϵ to obstacles, the adaptive refinement of the sufficiently small minimum interval $\delta \le \epsilon$ can always find a grid path P from the source to the destination. The second statement on the suboptimality ratio of the grid-based planning means that a small δ_0 assists in generating more precise paths, thereby reducing the upper bound on the length of the grid path. However, there exists a trade-off where the number of nodes increases proportionally to ${\frac{1}{\delta_0}}^K$, resulting in an escalation of computational costs for node management and path planning.

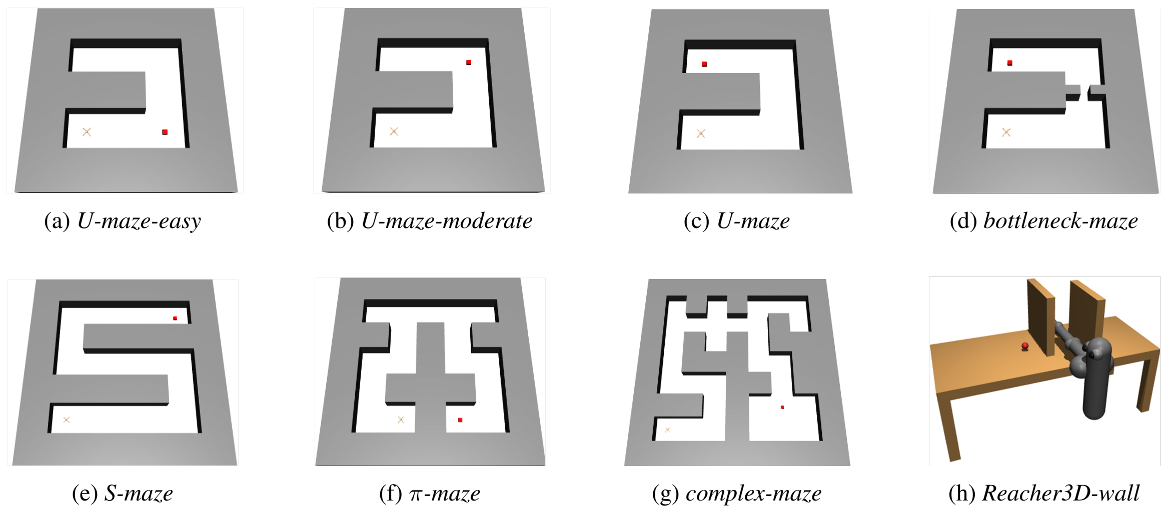
6. Experiments

6.1. Experimental setup

Baselines We compare our method, BEAG, with the state-of-the-art graph-based RL algorithms in the following:

- HIRO (Nachum et al., 2018): HIRO is a vanilla HRL method that has a 2-level learnable hierarchical policy and uses no graphs for training or inference.
- HIGL (Kim et al., 2021): HIGL constructs a graph from the replay buffer by considering both coverage and novelty. In addition, it employed a shortest-path algorithm to find a path and obtained a restricted subgoal using an adjacency network.
- DHRL (Lee et al., 2022): DHRL constructs a graph from the replay buffer based on farthest point sampling (FPS) algorithm (Arthur et al., 2007), which aids in generating a uniform graph. Then, it follows the shortest path from the current state to the subgoal generated by the high-level policy.
- PIG (Kim et al., 2023): PIG also constructs a graph based on the FPS algorithm and employs imitation learning to ensure that the same action is taken for each subgoal along the shortest path.

Training setups Goal-conditioned RL has various training setups depending on the initial state space and goal space, each of which serves a unique purpose. For instance, there can be a random initial state space and a random goal space, where the objective is to be able to reach anywhere from anywhere. However, especially in graph-based RL, utilizing a random initial state and goal space allows for data collection across the entire environment without the need for exploration. Also, in an environment where either the initial state or goal is randomly generated, the assumption of awareness regarding reachable goal spaces is necessary. This makes it a less challenging environment, and the scope of its utilization becomes limited. Therefore, we conducted the overall experiments in a fixed initial state and goal. Nevertheless, in Section 6.2, we also provide experiments using the training environment where a fixed initial state with randomly generated goals.



(a) *U-maze-easy* (b) *U-maze-moderate* (c) *U-maze* (d) *bottleneck-maze*

(e) *S-maze* (f) π*-maze* (g) *complex-maze* (h) *Reacher3D-wall*

Figure 3: **AntMaze and Reacher3D environments.** We evaluate graph-based RL methods in the set of MuJoCo environments depicted above, in challenging setups with sparse rewards over a long horizon. We note that the goal space for the AntMaze and the Reacher3D utilized 2-dimensional space and 3-dimensional space, respectively.

Evaluation Our primary evaluation metric for RL algorithms is the *success rate*, a widely adopted metric in prior works (Lee et al., 2022; Kim et al., 2021; 2023). Specifically, the success rate is the ratio of successfully achieving the most challenging goal (marked in Figure 3) in 10 trials.

Environments Our experiments were conducted on a set of challenging tasks with sparse rewards over a long horizon, by configuring the MuJoCo environments (AntMaze and Reacher). Specifically, the following environments are considered, and visual representations are provided in Figure 3.

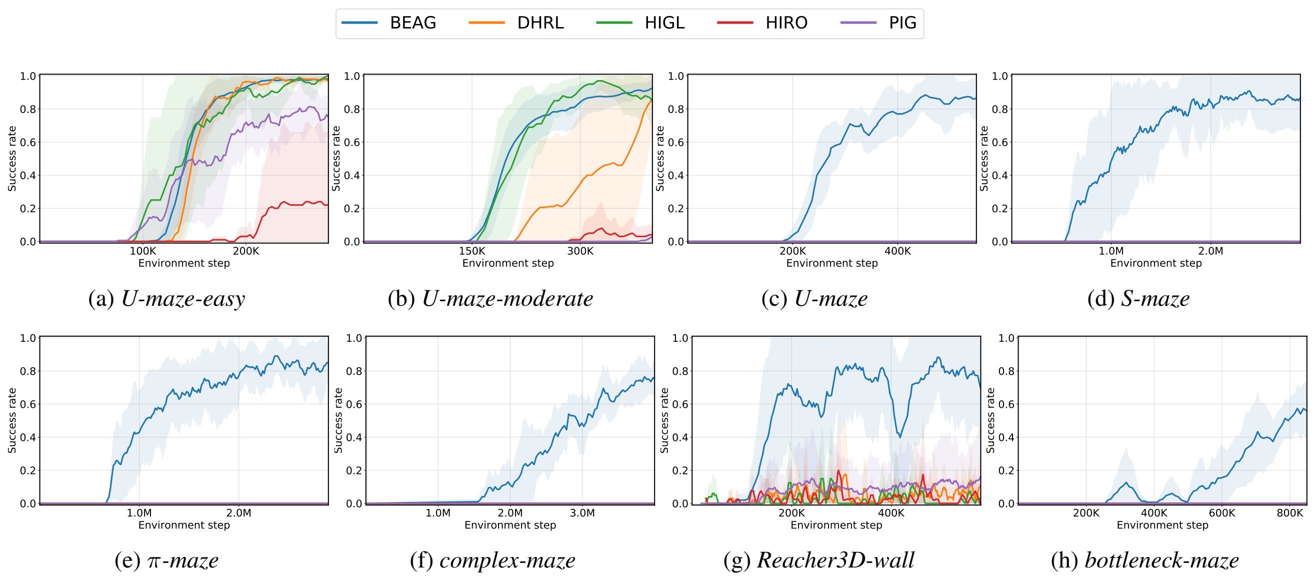
- $\{$*U, bottleneck, S,* π*, complex*$\}$*-maze*: We consider a variety of configurations of AntMaze environments with maps and ultimate goals, as illustrated in Figure 3. The size of maps ranges from 24 $\times$ 24 to 56 $\times$ 56 (*U, bottleneck*: 24 $\times$ 24, *S,* π: 40 $\times$ 40, *complex*: 56 $\times$ 56). The agent is basically tested for one of the most challenging goals, spotted by red dots in Figure 3. For further analysis, we also consider *U-maze-easy* and *U-maze-moderate* with ultimate goals with different difficulties shown in Figure 3a, b, respectively. It is worth noting that BEAG used the same hyperparamaters ($\delta_0 = 4, \tau_t = 200, \tau_n = 3$) for all experiments in the AntMaze environment.

- *Reacher3D-wall*: The robotic arm simulates moving the tip of the hand to the target position in 3D environment with obstacles, depicted in Figure 3h. The arm movement cannot pass through the obstacles.

6.2. Comparative results

Exploration in fixed goal setting (Figure 4) As shown in Figure 4, BEAG demonstrates significant performances across all environments. While previous baselines generally fail to achieve the target goal in most environments, they exhibit comparable performance in easy scenarios, *U-maze-easy* and *U-maze-moderate*. To delve deeper into why baselines are failing, we utilized the previous state-of-the-art method DHRL. We examine how the graph-based planner is providing subgoals (Figure 1) and plot the coverage (Figure 6a, b) and generated graphs (Figure 7a, b) as learning progresses. This demonstrates that over time, the success area of BEAG expands, while DHRL persistently attempts unattainable subgoals over the wall and continues to succeed only in the vicinity of the initial state. These findings underscore the superiority of breadth-first search compared to the consecutive failures observed in previous graph-based RL methods.

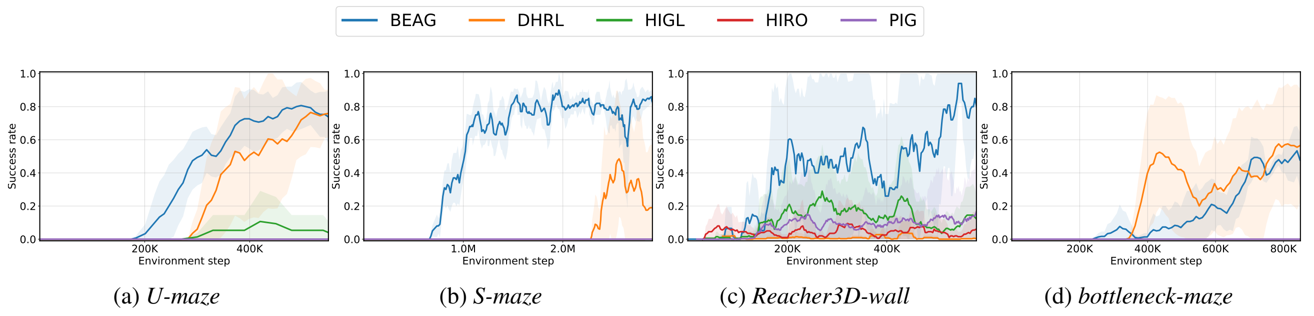
Exploration in random goal setting (Figure 5) In Figure 5, we evaluate the baselines and BEAG in random goal settings, where the training goals are randomly selected over only the feasible goal space. Previous methods (Nachum et al., 2018; Kim et al., 2021; Lee et al., 2022) have focused on these settings. However, our BEAG shows clear superiority compared to the baselines in most environments even in these settings. Interestingly, BEAG exhibits consistent performance regardless of the training goal setting, while DHRL shows a significant increase in performance on the random goal setting. This is because it is possible to identify feasible subgoals by simply observing sampled goals,



(a) *U-maze-easy* (b) *U-maze-moderate* (c) *U-maze* (d) *S-maze*

(e) π*-maze* (f) *complex-maze* (g) *Reacher3D-wall* (h) *bottleneck-maze*

Figure 4: **Success rates in various environments (fixed goal).** We report the average success rate as a solid line and the standard deviation as a shaded region. Both BEAG and all other baselines are trained with a fixed initial state and goal setting. We note that certain baselines may not be visible in specific environments due to overlapping values, especially at zero success rates.



(a) *U-maze* (b) *S-maze* (c) *Reacher3D-wall* (d) *bottleneck-maze*

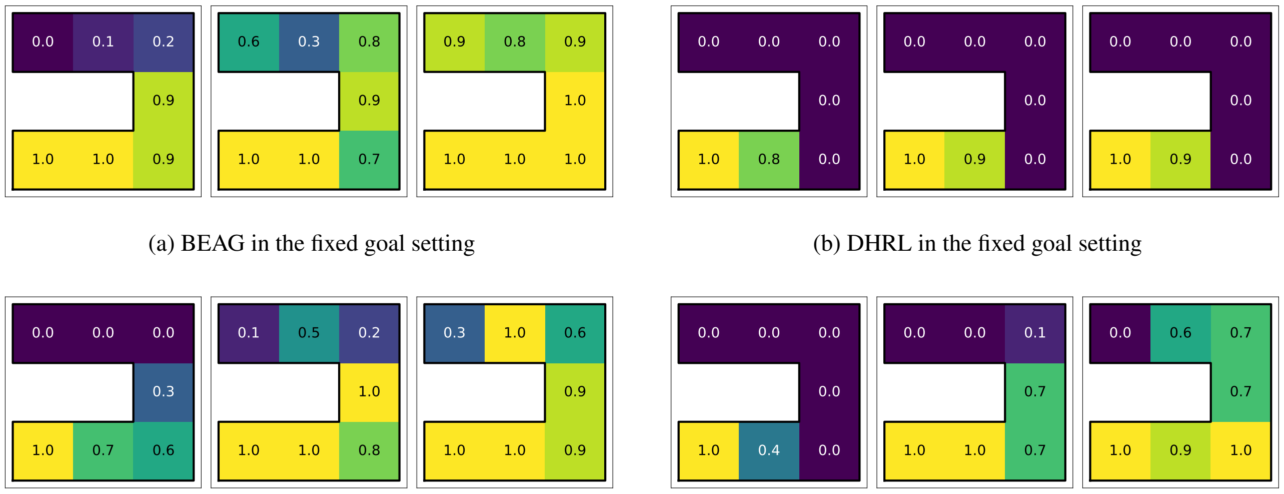
Figure 5: **Success rates in random goal setup.** We report the average success rate of BEAG and all other baselines, which are trained with a fixed initial state and a randomly assigned goal setting.

although the baselines do not explicitly exploit the observed goal distribution. For example, in *U-maze*, if sampled goals are sequentially provided as easy, moderate, and challenging goals over a few epochs, the graph would progressively expand, enabling the baselines to generate paths leading to the challenging goal. On the other hand, BEAG can explore without the assistant for an exploration coming from randomly sampled goals, demonstrates faster exploration performance compared to DHRL, as evident in Figure 6c, d and Figure 7c, d.

Adaptivity of BEAG (Figure 4h, 5d) The structure of the *bottleneck-maze*, presented in Figure 3d, involves a narrow gate dividing the maze into two parts, requiring the agent to recognize the bottleneck for effective navigation. Thanks to adaptive grid refinement, a sparse grid graph facilitates the discovery of successful paths, as illustrated in Figure 2. As evident in Figure 4h, BEAG can succeed in *bottleneck-maze*. However, despite using the same size of goal space, there exists a performance difference with *U-maze* due to the need for adaptive grid refinement in *bottleneck-maze*. We would like to emphasize that we intentionally maintained consistency in hyperparameters across different environments, refraining from cherry-picking hyperparameters for the *bottleneck-maze*.

6.3. Ablation study

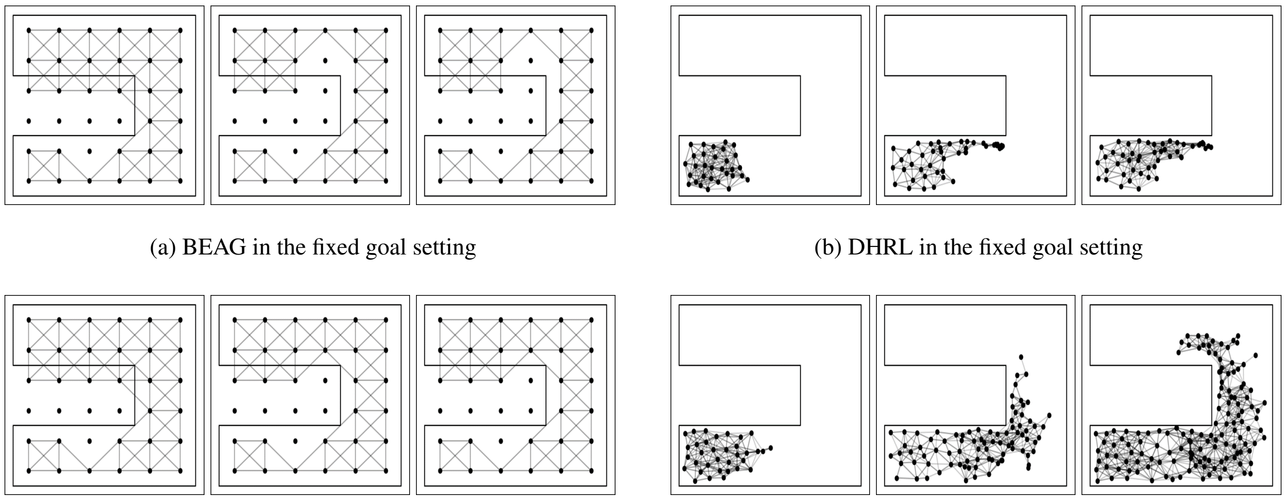
Initial interval of grid (Figure 8a) As shown in Figure 8a, BEAG demonstrates a slightly delayed increase at interval 1 than 2. That's because the number of subgoals for which



(a) BEAG in the fixed goal setting (b) DHRL in the fixed goal setting

(c) BEAG in the random goal setting (d) DHRL in the random goal setting

Figure 6: **Success rates for each region in *U-maze*.** We visualize the success rates of randomly sampled 10 goals in each square tile at 5, 10, and 15 epochs (174K, 219K, and 264K environment steps) for BEAG and DHRL, which are trained in (a,b) fixed goal setting and (c,d) random goal setting.



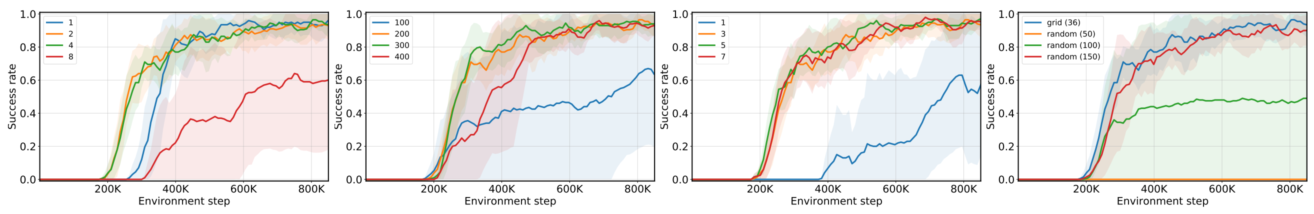
(a) BEAG in the fixed goal setting (b) DHRL in the fixed goal setting

(c) BEAG in the random goal setting (d) DHRL in the random goal setting

Figure 7: **Graphs generated in *U-maze*.** We visualize the graphs generated at 5, 10, and 15 epochs (174K, 219K, and 264K environment steps) for BEAG and DHRL, which are trained in (a,b) fixed goal setting and (c,d) random goal setting.

the planner needs to determine attainability doubles, leading to an increase in experiment steps. Nonetheless, BEAG demonstrates comparable performance at intervals 2 and 4, owing to the agent's requirement for a larger number of samples to accurately navigate a distant subgoal. At interval 8, before the policy learns to achieve a distant subgoal, refinement occurs to interval 4. However, adaptive grid refinement proceeds with a random order among failed subgoals within the same interval level, leading to a significant standard deviation in the results due to the randomness.

Failure condition and count (Figure 8b, c) To identify failure, we utilize a count-based method with two thresholds: failure condition τ_t and failure count τ_n. Primarily, we conduct experiments by varying τ_t to 100, 200, 300, and 400 steps. As illustrated in Figure 8b, with τ_t set to 100, a significant number of false negatives (attainable subgoals but identified as unattainable) occurred, resulting in a notably low success rate. Moreover, at τ_t of 400, there is a delay in exploration, causing a slightly slower increase in the success rate. Additionally, we conduct experiments



(a) Initial interval δ_0 (b) Failure condition τ_t (c) Failure count τ_n (d) Random graph

Figure 8: **Ablation studies.** We report the average success rate by varying the parameters of (a) initial intervals of the grid, (b) failure condition, and (c) failure count. Additionally, we report the average success rate (d) based on the graph structure. We note that all experiments are conducted in *U-maze* task.

by varying τ_n to 1, 3, 5 and 7. Similar to τ_t, choosing an appropriate τ_n is crucial, as a small τ_n may lead to false negatives. A notable performance decrease is observed at $\tau_n = 1$. To ensure stability, we employed the thresholds ($\tau_t = 200$, $\tau_n = 3$) for BEAG in the other experiments.

Grid vs. random graph (Figure 8d) To construct the graph, we utilize a grid graph for its uniform distribution, ensuring the presence of a nearby subgoal even with a limited number of nodes, regardless of the given goal. However, even without a grid, graphs that uniformly distributed within a given goal space, and we can consider a random graph constructed by uniformly randomly sampled subgoals. In Figure 8d, we conduct an experiment comparing with our method on the random graph. Despite a comparable number of nodes (50), outcomes from randomly sampled nodes resulted in failures. With a sufficiently larger number of nodes (100), we observe a significant standard deviation in the results, indicating instances of failure depending on the random seed. Achieving a success rate similar to the grid (36) requires approximately 4 times the number of nodes (150). Consequently, in complex environments, choosing the randomly initialized graph may not be favorable when considering computational costs.

7. Conclusion

In this work, we introduce BEAG, a novel approach that places a strong emphasis on sample-efficient exploration to enhance goal achievement. It is noteworthy that we propose a more constrained experimental setup where a fixed goal is provided during training, and only BEAG achieved strong performance in this setting as well. Our experimental findings showcase the rapid expansion of exploration, facilitated by predictive techniques applied to nodes previously considered unreachable within a grid graph framework. BEAG not only surpasses the performance of prior state-of-the-art methods but also introduces a fresh perspective to the realm of graph-based reinforcement learning. We also expect that the breadth-first exploration, which avoids repeated failures and enables a more diverse range of exploration, can be applied not only to goal-conditioned RL but also to other RL frameworks. While we have applied a straightforward grid refinement approach to a grid graph in this study, we anticipate the existence of more sophisticated heuristics for adaptive grid graph construction and the selection of candidate refinements in the future.

Limitation While BEAG has demonstrated significant performance improvements in the context of sparse and long-horizon tasks, several limitations remain to be addressed. First, in environments with high-dimensional goal spaces, the number of subgoals to construct graphs increases exponentially with respect to the dimension K. Accordingly, notable time and computational resources are required to manage graphs not only for BEAG but also for graph-based RL methods (Lee et al., 2022; Kim et al., 2021; Nachum et al., 2018). This limitation can be mitigated by reducing the dimensionality of the goal space, such as by utilizing latent state space (Li et al., 2021). Furthermore, for the sake of simplicity, we identify a part to be refined using hyperparameters τ_t and τ_n. However, this design choice can be problematic in stochastic environments, as it can inevitably misclassify attainable nodes as unattainable. To address this limitation, future research could explore methods based on probabilistic modeling.

Acknowledgments

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.RS-2019-II191906, Artificial Intelligence Graduate School Program (POSTECH); No.RS-2021-II212068, Artificial Intelligence Innovation Hub; No.RS-2024-00436680, Collaborative Research Project with MSRA), Samsung Research Funding & Incubation Center of Samsung Electronics (SRFC-TF2103-02), and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2022R1A6A1A03052954).

Impact statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., and Zaremba, W. Hindsight experience replay. *Advances in Neural Information Processing Systems*, 30, 2017.

Arthur, D., Vassilvitskii, S., et al. k-means++: The advantages of careful seeding. In *Soda*, volume 7, pp. 1027–1035, 2007.

Bagaria, A., Senthil, J. K., and Konidaris, G. Skill discovery for exploration and planning using deep skill graphs. In *International Conference on Machine Learning*, pp. 521–531. PMLR, 2021.

Berger, M. J. and Oliger, J. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of computational Physics*, 53(3):484–512, 1984.

Combes, R., Magureanu, S., and Proutiere, A. Minimal exploration in structured stochastic bandits. *Advances in Neural Information Processing Systems*, 30, 2017.

Dijkstra, E. W. A note on two problems in connexion with graphs. *Numerische Mathematik*, pp. 269–271, 1959.

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

Gieselmann, R. and Pokorny, F. T. Planning-augmented hierarchical reinforcement learning. *IEEE Robotics and Automation Letters*, 6(3):5097–5104, 2021.

Haarnoja, T., Ha, S., Zhou, A., Tan, J., Tucker, G., and Levine, S. Learning to walk via deep reinforcement learning. In *Proceedings of Robotics: Science and Systems*, 2019.

Huang, Z., Liu, F., and Su, H. Mapping state space using landmarks for universal goal reaching. *Advances in Neural Information Processing Systems*, 32, 2019.

Kim, J., Seo, Y., and Shin, J. Landmark-guided subgoal generation in hierarchical reinforcement learning. *Advances in Neural Information Processing Systems*, 34: 28336–28349, 2021.

Kim, J., Seo, Y., Ahn, S., Son, K., and Shin, J. Imitating graph-based planning with goal-conditioned policies. *International Conference on Learning Representations*, 11, 2023.

Lee, S., Kim, J., Jang, I., and Kim, H. J. Dhrl: A graph-based approach for long-horizon and sparse hierarchical reinforcement learning. *Advances in Neural Information Processing Systems*, 35:13668–13678, 2022.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

Li, S., Zhang, J., Wang, J., Yu, Y., and Zhang, C. Active hierarchical exploration with stable subgoal representation learning. *International Conference on Learning Representations*, 9, 2021.

Nachum, O., Gu, S. S., Lee, H., and Levine, S. Data-efficient hierarchical reinforcement learning. *Advances in Neural Information Processing Systems*, 31, 2018.

Nasiriany, S., Pong, V., Lin, S., and Levine, S. Planning with goal-conditioned policies. *Advances in Neural Information Processing Systems*, 32, 2019.

Ok, J., Proutiere, A., and Tranos, D. Exploration in structured reinforcement learning. *Advances in Neural Information Processing Systems*, 31, 2018.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, pp. 2778–2787. PMLR, 2017.

Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., and Levine, S. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *Proceedings of Robotics: Science and Systems*, 2018.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. Universal value function approximators. In *International Conference on Machine Learning*, pp. 1312–1320. PMLR, 2015.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897. PMLR, 2015.

Tam, A., Rabinowitz, N., Lampinen, A., Roy, N. A., Chan, S., Strouse, D., Wang, J., Banino, A., and Hill, F. Semantic exploration from language abstractions and pretrained representations. *Advances in Neural Information Processing Systems*, 35:25377–25389, 2022.

Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., and Kavukcuoglu, K. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pp. 3540–3549. PMLR, 2017.

Zhang, L., Yang, G., and Stadie, B. C. World model as a graph: Learning latent landmarks for planning. In *International Conference on Machine Learning*, pp. 12611–12620. PMLR, 2021.

Zhang, T., Guo, S., Tan, T., Hu, X., and Chen, F. Generating adjacency-constrained subgoals in hierarchical reinforcement learning. *Advances in Neural Information Processing Systems*, 33:21579–21590, 2020.

A. Pseudo Algorithm

Algorithm 1 Overview of BEAG

```
Input: total training episode $n_{\text{epi}}$, initial random episode $n_{\text{rand}}$, refinement frequency $n_{\text{ref}}$, max timesteps per episode $T$,
  failure count threshold $\tau_n$, failure condition threshold $\tau_t$, environment Env, policy $\pi$, goal-feature function $\phi$
  for $ep \leftarrow 1$ to $n_{\text{epi}}$ do
    Env.reset()
    if $ep = n_{\text{rand}}$ then
      $\mathcal{H}(\mathcal{V}, \mathcal{E}) \leftarrow$ GridGraphConstruction
      unattained subgoal set $\mathcal{V}_u \leftarrow \emptyset$
      $\mathcal{H}' \leftarrow \mathcal{H}$
    end if
    for $t \leftarrow 1$ to $T$ do
      if $ep < n_{\text{rand}}$ then
        $a_t$ is randomly sampled from action space $\mathcal{A}$
      else
        if $t = 0$ then
          $P, \mathcal{H}', \mathcal{V}_u \leftarrow \texttt{FindPath}(\mathtt{s_0}, \mathtt{g}, \mathcal{H}', \mathcal{V}_\mathtt{u})$
          waypoint index $p \leftarrow 1$; tracking time $t_{tr} \leftarrow 0$
        end if
        if achieved goal $g$ then
          $P, \mathcal{H}', \mathcal{V}_u \leftarrow \texttt{FindPath}(\mathtt{s_t}, \mathtt{g}, \mathcal{H}', \mathcal{V}_\mathtt{u})$
          $p \leftarrow 1$; $t_{tr} \leftarrow 0$
        else if achieved waypoint $v_p$ then
          $n^a_{v_p} \leftarrow n^a_{v_p} + 1$
          $n^s_{v_p} \leftarrow n^s_{v_p} + 1$
          $p \leftarrow p + 1$; $t_{tr} \leftarrow 0$
        else if $t_{tr} > \tau_t$ then
          $n^a_{v_p} \leftarrow n^a_{v_p} + 1$
          $\mathcal{H}' \leftarrow \texttt{RemoveSubgoal}(\mathcal{H}', \mathtt{v_p})$
          if $n^a_{v_p} > \tau_n$ and $n^s_{v_p} = 0$ then
            $\mathcal{H} \leftarrow \texttt{RemoveSubgoal}(\mathcal{H}, \mathtt{v_p})$
            $\mathcal{V}_u \leftarrow \mathcal{V}_u \cup \{v_p\}$
          end if
          $P, \mathcal{H}', \mathcal{V}_u \leftarrow \texttt{FindPath}(\mathtt{s_t}, \mathtt{g}, \mathcal{H}', \mathcal{V}_\mathtt{u})$
          $p \leftarrow 1$; $t_{tr} \leftarrow 0$
        end if
        $a_t \leftarrow \pi(s_t, v_p)$
      end if
      Env.step($a_t$)
      TrainingPolicy
      $t_{tr} \leftarrow t_{tr} + 1$
    end for
  end for
```

Algorithm 2 Training policy

```
Input: replay buffer $\mathcal{B}$, maximum her step $h_{max}$
  sample $D \leftarrow (s_t, sg_t, a_t, r(s_{t+1}, sg_t), s_{t+1}) \in \mathcal{B}$
  relabel $sg_t \leftarrow \hat{sg}_t = \phi(s_{t+t_h})$ where $t_h \sim$ Uniform($[0, \min(t_{total} - t, h_{max})]$)
  update $\pi$ and $\tilde{V}_\pi$ using $D$
```

Algorithm 3 Remove Subgoal

```
Input: graph $\mathcal{H}$, subgoal $v$
Output: modified graph $\mathcal{H}$
  for $(u, v) \in \mathcal{E}$ do
    $w(u, v) \leftarrow \infty$
  end for
  for $(v, u) \in \mathcal{E}$ do
    $w(v, u) \leftarrow \infty$
  end for
  return $\mathcal{H}$
```

Algorithm 4 Grid Graph Construction

```
Input: grid interval $\delta_0$, goal space dimension $K$, lower bound of goal space $L$, upper bound of goal space $U$
Output: graph $\mathcal{H}$
  vertex set $\mathcal{V} \leftarrow \emptyset$; edge set $\mathcal{E} \leftarrow \emptyset$
  for $m \leftarrow 1$ to $K$ do
    $\mathcal{V}_m \leftarrow \emptyset$
    $x \leftarrow L$
    while $x < U$ do
      $\mathcal{V}_m \leftarrow \mathcal{V}_m \cup \{x\}$
      $x \leftarrow x + \delta_0$
    end while
  end for
  $\mathcal{V} \leftarrow \mathcal{V}_1 \times \mathcal{V}_2 \times ... \times \mathcal{V}_K$
  for $v \in \mathcal{V}$ do
    interval $\delta_v \leftarrow \delta_0$
    $n_v^a \leftarrow 0$
    $n_v^s \leftarrow 0$
  end for
  for $u \in \mathcal{V}$ do
    for $v \in \mathcal{V}$ do
      if $\|u - v\|_\infty = \delta_0$ then
        $\mathcal{E} \leftarrow \mathcal{E} \cup \{(u, v)\}$
      end if
    end for
  end for
  $\mathcal{H} \leftarrow (\mathcal{V}, \mathcal{E})$
  return $\mathcal{H}$
```

Algorithm 5 Find Path

```
Input: current state $s$, goal $g$, graph $\mathcal{H}$, unattained subgoal set $\mathcal{V}_u$, discount factor $\gamma$
Output: generated path from current $s$ to $g$ $(\phi(s), v_1, v_2, \cdots, g)$, refined graph $\mathcal{H}_r$, unattained subgoal set $\mathcal{V}_u$
  for $(u, v) \in \mathcal{E}$ do
    $\tilde{d}(u, v) = \log_\gamma(1 + (1 - \gamma)\tilde{V}_\pi(u|v))$
    $w(u, v) = \tilde{d}(u, v)$
  end for
  $\mathcal{H}_r \leftarrow \mathcal{H}$
  $P \leftarrow$ Dijkstra's Algorithm$(\mathcal{H}_r, \phi(s), g)$: $(\phi(s), v_1, v_2, ..., g)$
  for $v \in P$ do
    if $v \in \mathcal{V}_u$ then
      $\mathcal{H}_r \leftarrow \texttt{AdaptiveGridRefinement}(\mathcal{H}_\mathtt{r}, \mathcal{V}_\mathtt{u})$
      for $(u, v) \in \mathcal{E}$ do
        $\tilde{d}(u, v) = \log_\gamma(1 + (1 - \gamma)\tilde{V}_\pi(u|v))$
        $w(u, v) = \tilde{d}(u, v)$
      end for
      $P \leftarrow$ Dijkstra's Algorithm$(\mathcal{H}_r, \phi(s), g)$
    end if
  end for
  return $P, \mathcal{H}_r, \mathcal{V}_u$
```

Algorithm 6 Adaptive Grid Refinement

```
Input: graph $\mathcal{H}$, unattained subgoal set $\mathcal{V}_u$
Output: refined graph $\mathcal{H}_r$, unattained subgoal set $\mathcal{V}_u$
  $\delta \leftarrow \max_{v \in \mathcal{V}_u} \delta_v$
  $u' \sim \mathcal{V}_u$, where $\delta_{u'} = \delta$
  $\mathcal{V}_u \leftarrow \mathcal{V}_u \setminus \{u'\}$
  $\delta_{u'} \leftarrow \delta_{u'}/2$
  $\mathcal{V}_r \leftarrow \bigtimes_{i=1}^{K} \left\{ u'_i - \delta, u'_i - \frac{\delta}{2}, u'_i, u'_i + \frac{\delta}{2}, u'_i + \delta \right\}$
  $\mathcal{E}_r \leftarrow \emptyset$
  for $u \in \mathcal{V}_r$ do
    for $v \in \mathcal{V}_r$ do
      if $\|u - v\|_\infty = \delta_{u'}$ then
        $\mathcal{E}_r \leftarrow \mathcal{E}_r \cup \{(u, v)\}$
      end if
    end for
  end for
  $\mathcal{V}_r \leftarrow \mathcal{V} \cup \mathcal{V}_r$
  $\mathcal{E}_r \leftarrow \mathcal{E} \cup \mathcal{E}_r$
  $\mathcal{H}_r \leftarrow (\mathcal{V}_r, \mathcal{E}_r)$
  return $\mathcal{H}_r, \mathcal{V}_u$
```

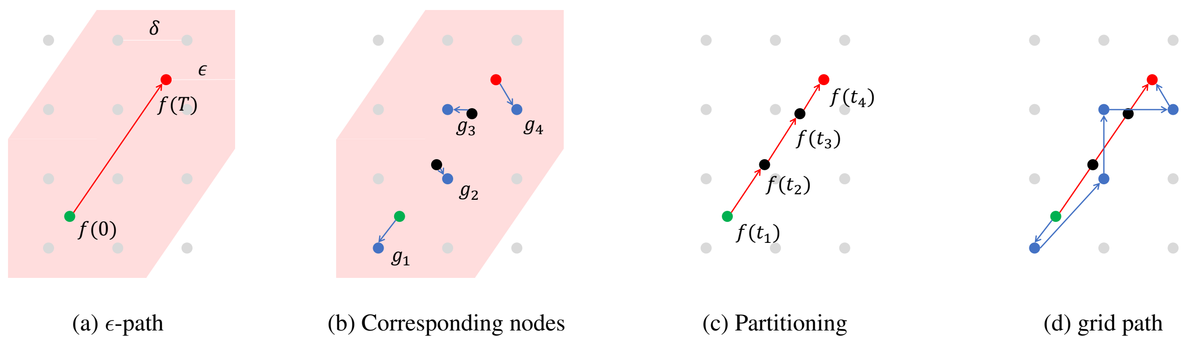


Figure 9: **Illustration for understanding notation.** (a) ϵ-path f (solid red line) and reachable region (shaded area) (b) obtaining nearby graph node g_i from $f(t_i)$ (c) partitioning ϵ-path by the length of δ (d) generated grid path P

B. Proofs of Theorems

Lemma B.1. *Given a compact goal space $\mathcal{G}$ and a ϵ-path $f(\cdot)$, for a grid graph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ with interval δ that satisfies $\delta \leq \epsilon$ and includes a grid path $P = (g_0, g_1, ..., g_n)$ that is subset of the ϵ-path cover $C(f, \epsilon)$. The length of the grid path is at most $(\lceil \frac{l}{\delta} \rceil + 1)\delta\sqrt{K}$ where l is the length of the ϵ-path $f(\cdot)$.*

Proof of Lemma B.1

Proof. Let $g_0 = \phi(f(0))$ and $g_n = \phi(f(T))$. For any time t, there exists a node $g \in \mathcal{V}$ such that $\|f(t) - g\|_\infty \leq \frac{\delta}{2} \leq \epsilon$, and these nodes are also reachable goals as illustrated in Figure 9b.

As depicted in Figure 9c, we partition the trajectory f into $f(t_1) = f(0), f(t_2), \dots f(t_{n-1}) = f(T)$, where the length of each partition from $f(t_i)$ to $f(t_{i+1}) \leq \delta$. For all $i \in 1, 2, \dots, n-1$, we find $g_i \in \mathcal{V}$ such that $\|g_0 - g_1\|_\infty \leq \delta$ and $\|g_{n-1} - g_n\|_\infty \leq \delta$. Additionally,

$$\forall i \in 1, 2, ..., n-2, \|g_i - g_{i+1}\|_\infty \leq \|g_i - f(t_i)\|_\infty + \|f(t_i) - f(t_{i+1})\|_\infty + \|f(t_{i+1}) - g_{i+1}\|_\infty < \frac{\delta}{2} + \delta + \frac{\delta}{2} = 2\delta,$$

implying (g_i, g_{i+1}) belongs to $\mathcal{E}$.

To prove that $\forall i, \forall \lambda \in [0, 1], (1 - \lambda)g_i + \lambda g_{i+1}$ is a reachable goal, we assume the opposite and derive a contradiction. Let's suppose that $\exists i, \exists \lambda \in [0, 1], g' = (1 - \lambda)g_i + \lambda g_{i+1}$ is not a reachable goal. Since $\|g_i - g_{i+1}\|_\infty = \delta$, it follows that $\|g_i - g'\|_\infty = \lambda\delta$ and $\|g_{i+1} - g'\|_\infty = (1 - \lambda)\delta$. Then, either $\|g_i - g'\|_\infty$ or $\|g_{i+1} - g'\|_\infty$ less than $\frac{\delta_0}{2}$, which implies either $\|f(t_i) - g'\|_\infty$ or $\|f(t_{i+1}) - g'\|_\infty$ less than δ by triangular inequality. Since $\delta \leq \epsilon$, g' is a reachable goal, which derive a contradiction. Thus, $\forall i, \forall \lambda \in [0, 1], (1 - \lambda)g_i + \lambda g_{i+1}$ is a reachable goal.

As depicted in Figure 9d, we have established that we can find a grid path denoted as $P(g_0, g_1, ..., g_n)$, which is covered by the ϵ-ball of the given ϵ-path. In the process of finding this grid path, we can bound its length as follows:

$$(g_0, g_1) \leq \sqrt{K}\|g_0 - g_1\|_\infty \leq \frac{\delta}{2}\sqrt{K},$$

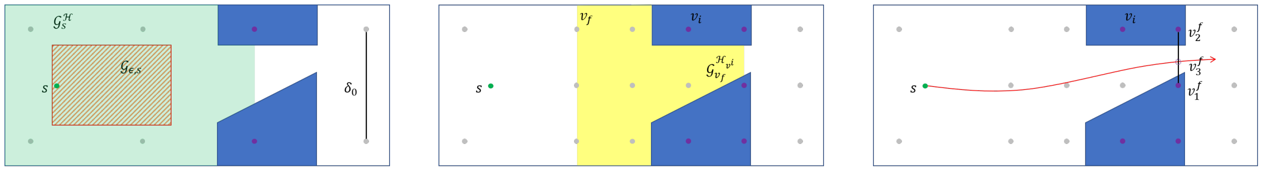
$$(g_{n-1}, g_n) \leq \sqrt{K}\|g_{n-1} - g_n\|_\infty \leq \frac{\delta}{2}\sqrt{K},$$

$$\forall i \in 1, 2, ..., n-2, (g_i, g_{i+1}) \leq \delta\sqrt{K},$$

since $(g_i, g_{i+1}) \in \mathcal{E}$. Additionally, given the length of each partition from $f(t_i)$ to $f(t_{i+1}) \leq \delta$, $n = \lceil \frac{l}{\delta} \rceil + 2$. Thus, the length of path P is upper bounded by $(\lceil \frac{l}{\delta} \rceil + 1)\delta\sqrt{K}$.

□

Definition B.2. Let $\mathcal{G}_s^{\mathcal{H}} \subseteq \mathcal{G}$ is a coverage of $\mathcal{G}$ on the grid graph $\mathcal{H}(\mathcal{V}, \mathcal{E})$ if $\forall g \in \mathcal{G}_s^{\mathcal{H}}$ there exists an grid path from s to g and $\mathcal{G}_{\epsilon,s} \subseteq \mathcal{G}$ is an ϵ-coverage of $\mathcal{G}$ if $\forall g \in \mathcal{G}_{\epsilon,s}$ there exists an ϵ-path from s to g.


$\mathcal{G}_s^{\mathcal{H}}$
$\mathcal{G}_{\epsilon,s}$
s
δ_0
v_f
v_i
$\mathcal{G}_{v_f}^{\mathcal{H}_{v^i}}$
s
v_i
v_2^f
v_3^f
v_1^f
s

(a) Grid graph coverage and ϵ-coverage (b) Coverage on the refined grid graph (c) ϵ-path and intersecting point

Figure 10: **Illustration for understanding notation.** (a) Inclusion relationship between $\mathcal{G}_{\epsilon,s}$ and $\mathcal{G}_s^{\mathcal{H}}$ (b) Expansion of $\mathcal{G}_s^{\mathcal{H}'}$ from adaptive refinement (c) ϵ-path has to cross edge between two unattainable nodes

Proof of Theorem 5.4

Proof. Let δ_0 is the initial intervals of $\mathcal{H}$, after breadth first exploration without grid refinement, we can find path on the $\mathcal{H}$ for all $g \in \mathcal{G}_{\delta_0,s}$ from Lemma B.1, i.e. $\mathcal{G}_{\delta_0,s} \subseteq \mathcal{G}_s^{\mathcal{H}}$ as illustrated in Figure 10a.

To select an infeasible node v^i for performing refinement, it is necessary to attempt distinguishing the feasibility of v^i. For such attempts, to initiate the process, it is crucial that at least one connected node v^f must be feasible, i.e. $\exists v^f \in \mathcal{V}, (v^i, v^f) \in \mathcal{E}$. After refinement v^i, our grid graph exchanged from $\mathcal{H}$ to $\mathcal{H}'(\mathcal{V}', \mathcal{E}')$, in other word $\mathcal{H}' = \mathcal{H} \cup \mathcal{H}_{v^i}$ where $\mathcal{H}_{v^i}$ is refined grid near the v^i. Since $v^f \in \mathcal{H}$, BEAG can explore the $\mathcal{H}_{v^i}$ by extending the path $s \to v^f$. After breadth first search on the $\mathcal{H}'$, we get $\mathcal{H}'_{v^i}$ which consist of the feasible nodes in $\mathcal{H}_{v^i}$. Then, as depicted in Figure 10b a coverage on the refined grid graph $\mathcal{H}'$ is extended:

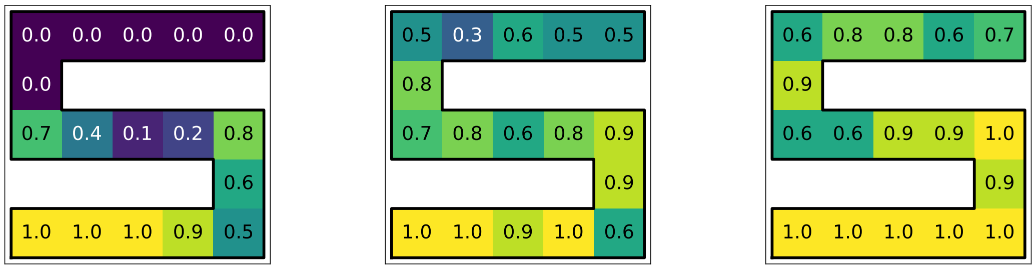
$$\mathcal{G}_s^{\mathcal{H}'} = \mathcal{G}_s^{\mathcal{H}} \cup \mathcal{G}_{v_f}^{\mathcal{H}_{v^i}} \cup \bigcup_{v}^{\mathcal{H}'_{v^i}} \mathcal{G}_{\delta_0,v}. \tag{7}$$

This process continues until all infeasible nodes are already refined with the intervals of $\frac{\delta_0}{2}$, and we claim that $\mathcal{G}_s^{\mathcal{H}'} \supseteq \mathcal{G}_{\frac{\delta_0}{2},s}$ at that time.

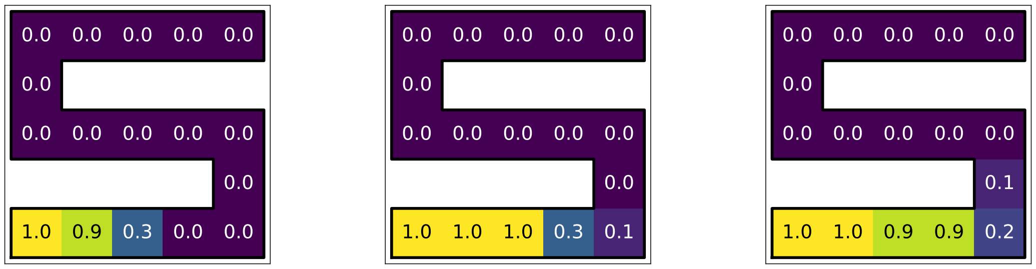
Suppose, for the sake of contradiction, that there exists a goal $g \in \mathcal{G}_{\frac{\delta_0}{2},s}$, i.e. $\exists \frac{\delta_0}{2}$-path from s to g, while $g \notin \mathcal{G}_s^{\mathcal{H}'}$. Since $g \notin \mathcal{G}_s^{\mathcal{H}'}$, g is not connected with feasible nodes in $\mathcal{H}'$, i.e. $\frac{\delta_0}{2}$-path must cross edge between infeasible nodes as illustrated in Figure 10c. Since all infeasible nodes have already been refined to have intervals of $\frac{\lambda}{2}$, any edge crossed by a $\frac{\delta_0}{2}$-path has an interval of $\frac{\delta_0}{2}$. Let the two nodes on this edge be denoted as v_1^f and v_2^f, so that $\|v_1^f - v_2^f\|_\infty = \frac{\delta_0}{2}$. If v_3^f is the node where the $\frac{\delta_0}{2}$-path crosses, then $\|v_1^f - v_3^f\|_\infty + \|v_3^f - v_2^f\|_\infty = \frac{\delta_0}{2}$. Thus, either $\|v_1^f - v_3^f\|_\infty \leq \frac{\delta_0}{2}$ or $\|v_3^f - v_2^f\|_\infty \leq \frac{\delta_0}{2}$ is true. Because both v_1^f and v_2^f are infeasible, leading to a contradiction with the definition of a $\frac{\delta_0}{2}$-path. By contradiction, we conclude that $\mathcal{G}_s^{\mathcal{H}'} \supseteq \mathcal{G}_{\frac{\delta_0}{2},s}$.

Applying the same approach, we can demonstrate that, for a graph $\mathcal{H}'$ refined up to intervals of δ, the relationship $\mathcal{G}_s^{\mathcal{H}'} \supseteq \mathcal{G}_{delta,s}$ holds true. As δ becomes sufficiently small, $s.t. \delta < \epsilon$, then $\mathcal{G}_{\epsilon,s} \subseteq \mathcal{G}_s^{\mathcal{H}'}$ which implies that the existence of the grid path P on the $\mathcal{H}'$. And, thanks to the Lemma B.1, we can also conclude that the length of the path P is upper bounded by $(\lceil \frac{l}{\delta_0} \rceil + 1)\delta_0\sqrt{K}$.

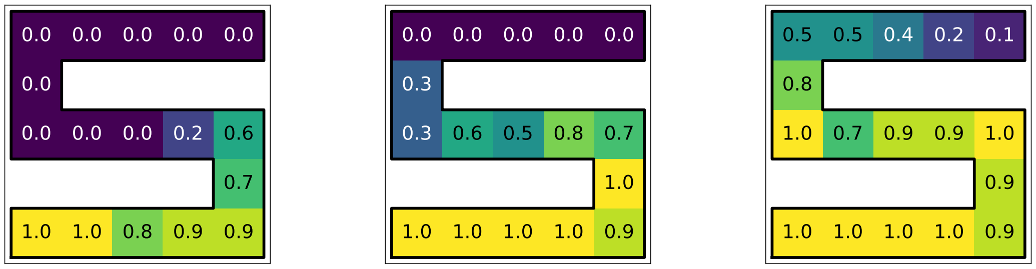
□



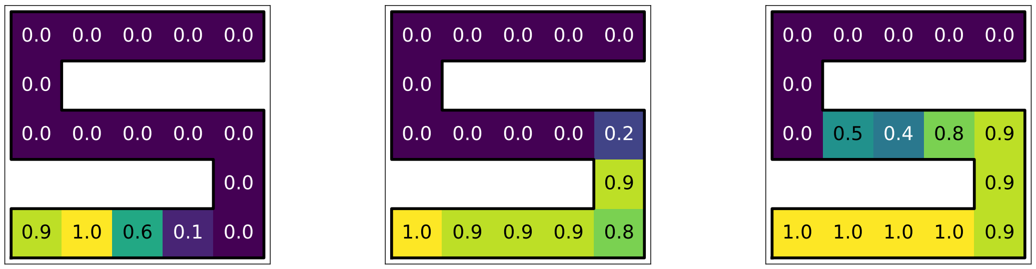
(a) BEAG on fixed goal setting



(b) DHRL on fixed goal setting



(c) BEAG on random goal setting

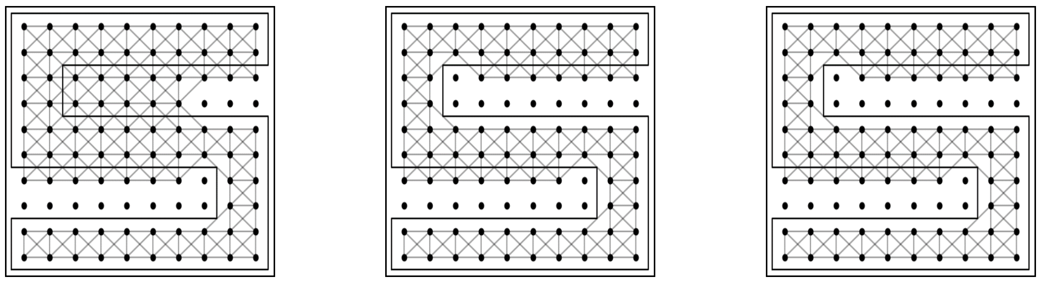


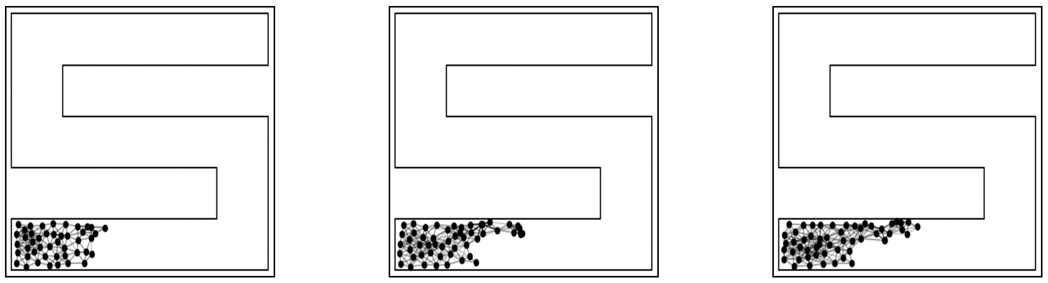
(d) DHRL on random goal setting

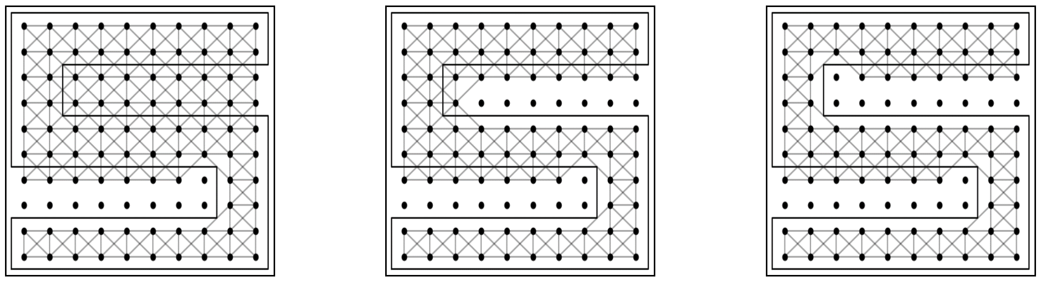
Figure 11: **Success rates for each region in *S-maze*.** We visualize the success rates of randomly sampled 10 goals in each region at 10, 20, and 30 epochs for BEAG and DHRL.

C. Coverage of the graph planner as learning progresses (S-maze)

To illustrate the exploration performance of algorithms, we visualize the average success rates for each region as learning progresses. Specifically, we compare the exploration capabilities of BEAG and DHRL in *S-maze* environments, both with fixed initial states at the bottom-left corners. Interestingly, BEAG even displays faster exploration performance in the fixed goal setting. This can be attributed to the absence of wasted episodes, as attainable goals can be derived from random goals, thereby mitigating unnecessary setbacks in exploration.


(a) BEAG on fixed goal setting


(b) DHRL on fixed goal setting


(c) BEAG on random goal setting

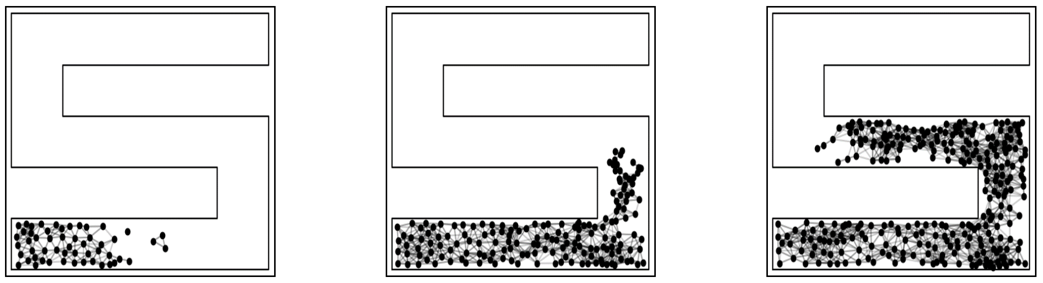

(d) DHRL on random goal setting

Figure 12: **Graphs generated in *S-maze*.** We visualize the graphs generated at 10, 20, and 30 epochs for BEAG and DHRL, which are trained in each goal setting.

D. Generated graphs from the graph planner as learning progresses (S-maze)

Similar to the coverage visualization depicted in Figure 11, we also visualize the graphs generated as learning progresses in *S-maze* environments to illustrate the exploration performance of algorithms. Interestingly, as depicted in Figure 12b, DHRL struggle with exploration in the fixed goal setting due to walls. In contrast, BEAG efficiently progresses exploration by eliminating impossible nodes within the walls.

Table 1: **Hyperparameters for DHRL and BEAG.**

	DHRL	Ours
initial episodes without graph planning	75	-
gradual penalty	1.5-5.0	-
high-level train freq	10	-
Frontier-based Goal Shifting	$\{\pi$, Complex$\}$-maze	-
number of landmarks	300-600	36-196 (initial)
hidden layer	(256, 256)	(256, 256)
actor lr	0.0001	0.0001
critic lr	0.001	0.001
τ	0.005	0.005
γ	0.99	0.99
batch size	1024	1024
target update freq	10	10
actor update freq	2	2

Table 2: **Hyperparameters for PIG.**

	Reacher	U-maze	π-maze	Complex-maze
Initial random trajectories	20k	100k	400k	800k
Number of nodes in a graph	80	400	500	500
Balancing coefficient λ	0.0001	0.001	0.001	0.001
Skipping temperature α	10.0	10.0	10.0	10.0
Hindsight relabelling range	50	200	200	200
Action L2	0.01	0.5	0.5	0.5
Action noise	0.1	0.2	0.2	0.2
clipping threshold for distances	4.0	38.0	38.0	38.0

Table 3: **Hyperparameters for HIGL and HIRO.**

	HIGL	HIRO
higl-level τ	0.005	0.005
π^{hi} lr	0.0001	0.0001
Q^{hi} lr	0.001	0.001
high-level γ	0.99	0.99
low-level τ	0.005	0.005
π^{lo} lr	0.0001	0.0001
Q^{lo} lr	0.001	0.001
low-level γ	0.95	0.95
hidden layer	(128, 128)	(128, 128)
number of coverage landmarks γ	20-100	-
number of novelty landmarks γ	20-400	-
batch size	128	128

E. Hyperparameter choice

When evaluating the previous Graph-based RL method, we used the same hyperparameters as used in their papers. And, we conducted additional tuning the number of landmarks for a fair comparison.