Enhancing Exploration and Exploitation in Hierarchical Reinforcement Learning with Subgoal Graph Learning

Yibo Zhang[1 2], **Dengpeng Xing**[1 2*]

[1]Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
zhangyibo2023@ia.ac.cn, dengpeng.xing@ia.ac.cn

Abstract

Goal-conditioned hierarchical reinforcement learning has demonstrated effectiveness in addressing complicated decision-making tasks by providing "temporal extraction", which decomposes tasks into smaller and more manageable "subgoals". This enables agents to plan over a longer time scale. However, achieving optimal exploration and exploitation still remains a challenge, especially for long-horizon or sparse-reward scenarios. In this paper, we introduce **A**ctive exploration and hierarchical **S**elf-**I**mitation (ASI), an effective scheme to enhance exploration and exploitation based on subgoal representation learning. The key point of ASI is to utilize temporal adjacency information in the representation space. We construct and dynamically update an adjacency graph that captures the relationships between subgoals. Based on the adjacency information provided by the graph, we design two mechanisms: active "frontier-reaching" exploration that faster expands the explored area by targeting boundary regions, and hierarchical self-imitation learning that leverages historical experience to facilitate both frontier reaching and policy training. Experimental results show that our method accelerates exploration and outperforms existing baselines in challenging long-horizon continuous control tasks.

Code — https://github.com/zzzyybbb/ASI

Introduction

Deep reinforcement learning (DRL) has achieved remarkable success in decision-making tasks, ranging from locomotion (Schulman et al. 2015; Zhu et al. 2017; Nachum et al. 2018), manipulation (Levine et al. 2016; Kalashnikov et al. 2018; Zhang et al. 2019), to gaming (Mnih et al. 2015; Lample and Chaplot 2017; Silver et al. 2018). However, DRL methods often struggle to maintain their effectiveness when faced with complex scenarios, including complicated manipulation, long-horizon planning, or sparse-reward tasks.

Hierarchical reinforcement learning (HRL) (Pateria et al. 2021) considers these challenges by decomposing complex tasks into simpler sub-tasks through hierarchical policy structures (Kulkarni et al. 2016; Bacon, Harb, and Precup 2017). Among the prevailing paradigms, goal-conditioned hierarchical reinforcement learning (GCHRL) has shown considerable potential in tackling complicated goal-reaching tasks (Dayan and Hinton 1992; Schmidhuber and Wahnsiedler 1992; Vezhnevets et al. 2017; Nair and Finn 2020; Nasiriany et al. 2019). By breaking down challenges into subgoal-conditioned sub-tasks, it enables agents to efficiently learn and achieve intermediate objectives, accelerating the whole learning process.

Typical GCHRL frameworks employ a two-level hierarchy: a high-level manager policy that periodically produces subgoals, and a low-level controller policy that executes primitive actions to achieve these subgoals. The subgoals can be represented either as explicit states, or learned representations. After receiving the subgoal, the low-level controller policy interacts with the environment and selects action sequences conditioned on the current subgoal.

In the above framework, a significant problem is effective subgoal representation (Dwiel et al. 2019). While early approaches mainly used the state space directly as the subgoal space (Levy et al. 2017; Eysenbach, Salakhutdinov, and Levine 2019; Liu et al. 2020), this proved inefficient for high-dimensional problems due to the exponential growth of the subgoal space. Most existing methods alleviate this issue either through learned representations (Nachum et al. 2019; Ghosh, Gupta, and Levine 2019; Rafati and Noelle 2019; Wang et al. 2024) or pre-defined domain-specific spaces (Zhang et al. 2020; Kim, Seo, and Shin 2021; Luo et al. 2024). However, learned subgoal representations sometimes lack stability, as exploration and the representation learning processes occur simultaneously, making the learned latent space adapts poorly to newly discovered states. This instability can severely impact training effectiveness. Several studies have attempted to alleviate this issue. For instance, recent work (Li et al. 2022) mitigates this by introducing penalties for dramatic representation changes.

Subgoal selection presents another challenge, requiring a comprehensive consideration of exploration and exploitation. Existing approaches mainly choose subgoals based on criteria such as reachability, novelty, and pre-defined intrinsic rewards (Pitis et al. 2020; Röder et al. 2020; Li et al. 2022; Luo et al. 2024), or choose subgoals through graph-based planning (Huang, Liu, and Su 2019; Zhang, Yang, and Stadie 2021; Lee et al. 2022). However, choice of subgoal selection strategies leads to a trade-off: exclusively pursuing

*Corresponding author.

novel subgoals may hinder exploitation, while overemphasizing reachability can lead to inefficient exploration, which is also called the exploration-exploitation dilemma.

In this paper, we propose a scheme to enhance both exploration and exploitation in GCHRL. Our approach leverages temporal adjacency information of latent state representations to construct a dynamic adjacency graph, enabling access to near-optimal transitions between explored latent states. Building upon the adjacency graph, we introduce an active "frontier-reaching" (FR) mechanism to efficiently guide agents toward unexplored regions, accelerating exploration. Additionally, we develop a novel hierarchical self-imitation (SI) approach to refine exploitation. Our main contributions include:

• **Faster exploration with frontier-reaching mechanism**: We take advantage of the adjacency graph to identify "frontier regions" of the explored area and select subgoals that efficiently guide agents toward them. After approaching frontier regions, we select subgoals according to visit counts, accelerating the exploration of unseen states.

• **Better exploitation with hierarchical self-imitation learning:** We design a hierarchical self-imitation learning scheme that enhances subgoal reachability within explored regions while ensuring reliable frontier-reaching processes. Leveraging latent transitions from the graph, we guide the training of both policy hierarchies, enabling faster convergence while maintaining near-optimality of policies.

Related Work

Goal-Conditioned Hierarchical Reinforcement Learning

Goal-conditioned hierarchical reinforcement learning is a framework designed to address complex goal-oriented tasks through hierarchical structures (Vezhnevets et al. 2017; Bacon, Harb, and Precup 2017). The hierarchical architecture of policies substantially enhances the learning efficiency. However, these structures are often prone to instability, as updates within one level of the hierarchy may have unexpected consequences on others. Previous works mitigated this issue by introducing off-policy correction methods (Nachum et al. 2018), or employing Hindsight Experience Replay (Andrychowicz et al. 2017) to achieve more stable training dynamics (Levy et al. 2017). Another critical element of GCHRL is Universal Value Function Approximator (UVFA) (Schaul et al. 2015), which is typically implemented as a neural network. UVFA enables estimation of value functions under diverse subgoals, thereby facilitating generalization in goal-directed tasks.

Subgoal Representation Learning

Learning a subgoal space is crucial for enhancing training efficiency in GCHRL, as it allows for a significant reduction in the dimensionality of the high-level action space. Existing approaches either rely on pre-defined subspaces of the state spaces or learn a latent space. Some previous works directly used *xy* spaces (Zhang et al. 2020; Kim, Seo, and Shin 2021; Kim et al. 2023). However, in unknown or unstructured environments, they may become infeasible. Variational Autoencoder (VAE) (Kingma 2014) was employed to learn representations in early works (Péré et al. 2018; Nair and Finn 2020). Nonetheless, such approaches may still suffer from redundancy in the representation space, and the use of VAEs may lead to underfitting, particularly for high-dimensional state spaces. Representation learning based on features with slow dynamics was proposed to capture task-related information and enhance exploration (Li et al. 2021). To mitigate the instability issue of representation learning, a probabilistic perspective was considered, leading to a probabilistic approach (Wang et al. 2024). Additionally, a penalty on changes between old and new representations was incorporated to prevent dramatic shifts in representations during the training process (Li et al. 2022).

Self-Imitation Learning in GCHRL

Self-imitation learning effectively enhances the alignment of policies with past good experiences. Utilization of these experiences can lead to faster policy learning and higher performance, especially in sparse-reward environments. Early work leveraged expert data along with Generative Adversarial Imitation Learning (Ho and Ermon 2016) to learn goal-conditioned policies (Ding et al. 2019). Another work employed a non-parametric high-level policy to guide the training of low-level policy (Chane-Sane, Schmid, and Laptev 2021). Besides, graph planning based on farthest point sampling was utilized by (Kim et al. 2023) to generate a series of landmarks for self-imitation fine-tuning. In contrast to these approaches, our work integrates self-imitation within a hierarchical architecture, where both levels of policies are trained based on a dynamically updated adjacency graph.

Preliminaries

Framework of GCHRL

A GCHRL task is commonly modeled as a Markov Decision Process (MDP) (Puterman 1990) defined by $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mathcal{G})$, where $\mathcal{S}, \mathcal{A}$ denote the state space and the action space, $\mathcal{P} = p(s'|s,a)$ describes the state transition dynamic, $\mathcal{R} = r(s,a)$ is the reward, $\mathcal{G}$ is the goal space and γ is the discount factor. In GCHRL, the high-level policy is expressed as $\pi_{hi}(sg|s,g;\theta_{hi})$, where $s \in \mathcal{S}, g \in \mathcal{G}$, θ_{hi} are parameters of the high-level network, and sg denotes a subgoal in the subgoal space $\mathcal{SG}$. The low-level policy is represented as $\pi_{lo}(a|s,sg;\theta_{lo})$, where $s \in \mathcal{S}, a \in \mathcal{A}, sg \in \mathcal{SG}$, and θ_{lo} are the parameters of the low-level network.

The high-level policy selects a new subgoal every c steps. At each time step t, when $t \equiv 0 \pmod c$, high-level policy generates a new subgoal $sg_t \sim \pi_{hi}(s_t, g; \theta_{hi})$. For $t \not\equiv 0 \pmod c$, the subgoal is determined by the subgoal transition process $h(sg_{t-1}, s_{t-1}, s_t)$, i.e., $sg_t = sg_{t-1} + s_{t-1} - s_t$ for relative subgoals and $sg_t = sg_{t-1}$ for absolute subgoals. The low-level policy then produces a primitive action $a_t \sim \pi_{lo}(s_t, sg_t; \theta_{lo})$ based on current subgoal.

The reward function for the high-level policy is defined as cumulative external rewards over c steps:

$$r_{hi,t'} = \sum_{t=0}^{c-1} r\big(s_{ct'+t}, a_{ct'+t}\big), \tag{1}$$

where t' is the extended time step for high-level policy, i.e. the high-level timestep t' corresponds to the low-level timestep $c \cdot t'$. The low-level reward is the distance between current state and the subgoal. In absolute subgoal settings,

$$r_{lo,t} = -\|\phi(s_{t+1}) - sg_t\|_2, \tag{2}$$

where $\phi : \mathcal{S} \to \mathcal{SG}$ maps states to the subgoal space.

Universal Value Function Approximator

The UVFA is a neural network designed to estimate action values conditioned on goals. UVFA is typically integrated into the Actor-Critic framework and serves as the critic network, evaluating state-action pairs conditioned on a given goal. In GCHRL, for the high-level policy, UVFA is represented as $Q_{hi}(s, sg, g; \psi_{hi})$, while for the low-level policy, it is defined as $Q_{lo}(s, a, sg; \psi_{lo})$, where ψ_{hi}, ψ_{lo} denote parameters of high-level and low-level networks, respectively.

UVFAs are learned by minimizing the temporal difference error. The losses for the high-level and low-level can be written as:

$$\begin{aligned}\mathcal{L}_{cri}(\psi_{hi}) = \mathbb{E}_{(s_{ct'}, sg_{ct'}, r_{hi,t'}, g)\sim\mathcal{B}_{hi}} \\ \left[(Q_{hi}(s_{ct'}, sg_{ct'}, g) - y_{hi,t'})^2\right], \\ y_{hi,t'} = r_{hi,t'} + \gamma Q_{hi}(s_{c(t'+1)}, \pi_{hi}(s_{c(t'+1)}, g), g),\end{aligned} \tag{3}$$

$$\begin{aligned}\mathcal{L}_{cri}(\psi_{lo}) = \mathbb{E}_{(s_t, a_t, r_{lo,t}, sg_t)\sim\mathcal{B}_{lo}}\left[(Q_{lo}(s_t, a_t, sg_t) - y_{lo,t})^2\right] \\ y_{lo,t} = r_{lo,t} + \gamma Q_{lo}(s_{t+1}, \pi_{lo}(s_{t+1}, sg_t), sg_t),\end{aligned} \tag{4}$$

where $\mathcal{B}_{hi}, \mathcal{B}_{lo}$ denote high- and low-level replay buffers.

During training, UVFA estimates are generally more accurate for nearby goals than for distant ones. Consequently, leveraging these more reliable estimates can help stabilize policy training. This forms one of the main motivations for incorporating self-imitation learning into policy training.

Method

In this section, we propose our framework, Active exploration and hierarchical Self-Imitation (ASI), for better exploration and exploitation. Our framework comprises two key modules: the frontier-reaching exploration module and the hierarchical self-imitation module. Both modules are built upon the adjacency latent graph, so we first introduce the representation learning process and the construction of the graph. Then these two modules and the learning process are introduced successively. An overview of the entire framework is illustrated in Figure 1.

Constructing Latent Graph

Directly constructing a graph in the original state spaces is impractical, as state spaces are typically too large. Thus, effective representation learning is essential for reducing the subgoal space. Let ϕ be a function that maps states to latent subgoal space: $\phi : \mathcal{S} \to \mathcal{SG}$. Following (Li et al. 2022), we employ a triplet loss and a stable loss to train it.

Triplet loss encourages states with small temporal distances to be closer, while separating states with large temporal distances:

$$\begin{aligned}\mathcal{L}_{tri}(\phi) = \mathbb{E}_{(s_t, s_{t+1}, s_{t+c})\sim\mathcal{B}} \\ \left[\|\phi(s_{t+1}) - \phi(s_t)\|_2 + \max\{0, \delta - \|\phi(s_{t+c}) - \phi(s_t)\|_2\}\right],\end{aligned} \tag{5}$$

where $\mathcal{B}$ is the replay buffer and δ represents a margin.

To achieve higher stability, the stable loss penalizes dramatic changes of representations:

$$\mathcal{L}_s(\phi) = \mathbb{E}_{s\sim\mathcal{B}}[\lambda(s)\|\phi(s) - \phi_{old}(s)\|_2], \tag{6}$$

where $\phi_{old}(s)$ is the representation network before update and $\lambda(s)$ is a regularization weight. For states with smaller triplet loss, $\lambda(s)$ is set to a larger value, making them more important in representation learning, enhancing stability.

The overall representation loss is the combination of the above two equations:

$$\mathcal{L}(\phi) = \mathcal{L}_{tri}(\phi) + \mathcal{L}_s(\phi). \tag{7}$$

With triplet loss and stable loss, stable subgoal space can be learned. Building upon this space, we construct an adjacency graph $\mathcal{M} = (\mathcal{V}, \mathcal{E}, w)$, where $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ represents the set of edges, and w refers to the set of weights. Since the subgoal space remains continuous, it is natural to discretize it. Suppose all subgoals within a square region with edge length d correspond to a single node, we can represent them by the vertex of the square. As we only consider temporal distance, the edge weight w_{ij} is defined as the minimum transition steps needed from node i to j:

$$\begin{aligned}w_{ij} = \min_{\tau\sim\mathcal{B}, (s_p, s_q)\sim\tau} |p - q|, \\ s.t.\ I_i[\phi(s_p)] = I_j[\phi(s_q)] = 1,\end{aligned} \tag{8}$$

where τ is the trajectory within the buffer, s_p, s_q are states in the original state space, p, q are timesteps, and $I_n(x)$ is the function indicating whether a point $x \in \mathcal{SG}$ belongs to a node n. If x belongs to n, then $I_n(x) = 1$; otherwise, $I_n(x) = 0$. Once the graph is constructed, distances between any two nodes can be calculated via any shortest-path algorithm.

Exploration with Frontier-Reaching Episodes

To enhance exploration, it is crucial to minimize time consumption within explored areas. One intuitive scheme is to identify the boundary of the explored area and encourage the agent to approach it, after which more unseen states can be explored. Motivated by this, we take advantage of $\mathcal{M}$ to facilitate this process. To determine the "frontier" area, we can identify the node with maximum temporal distance from the starting point, and consider its neighborhood as the target. This can be achieved by searching within $\mathcal{M}$.

To successfully guide agents to the desired area, it is essential to determine an appropriate route from the starting point to the target region. Since $\mathcal{M}$ already contains the adjacency information between nodes, this can also be achieved via graph search. In practice, by employing Dijkstra's algorithm (Dijkstra 1959) with path backtracking, both the frontier node and the appropriate route can be obtained.

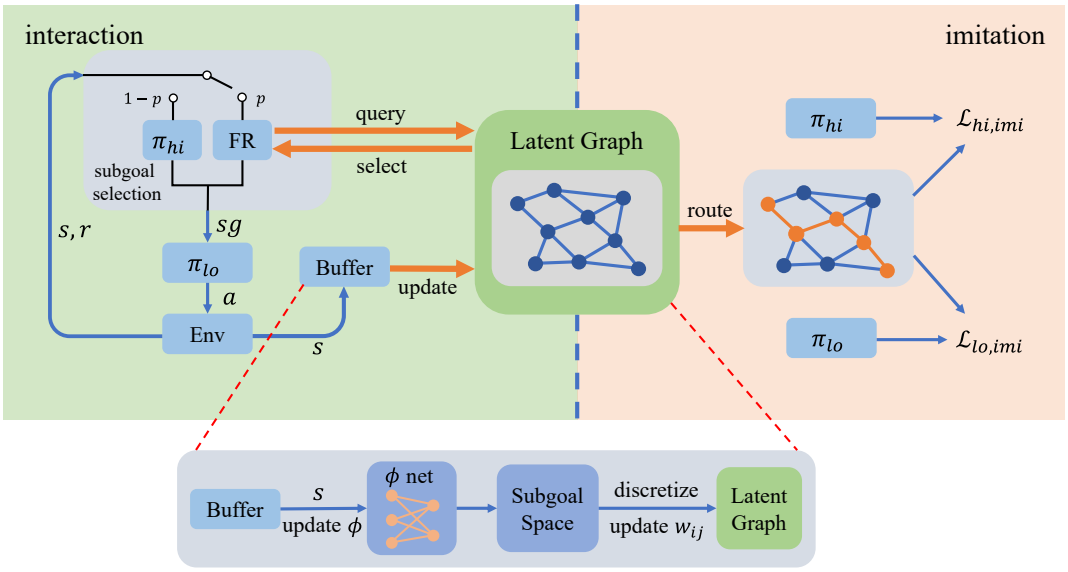


Figure 1: Framework of ASI. The left side illustrates the interaction process, while the right side depicts the hierarchical self-imitation process. The "FR" rectangle on the left denotes the frontier-reaching module, which selects subgoals from the latent graph; ϕ denotes the subgoal representation network. $\mathcal{L}_{hi,imi}$ and $\mathcal{L}_{lo,imi}$ on the right represent the high- and low-level self-imitation losses. Both two losses are computed based on the graph, which is continuously updated throughout training.

Specifically, at the beginning of a frontier-reaching episode, we sample the starting state s_0 and map it to the subgoal space through ϕ. Taking $\phi(s_0)$ as the starting point, we determine the frontier point $g_F \in \mathcal{SG}$ and the route $(l_1, ..., l_f)$, where $l_i \in \mathcal{SG}, i = 1, ..., f$, f denotes the number of points within the route, and l_1 and l_f denote the starting point and the frontier point, i.e., $l_1 = \phi(s_0)$ and $l_f = g_F$. After obtaining g_F and the route, the agent starts to interact with the environment. At timestep t, assume the agent's position is $\phi(s_t)$. When $t \equiv 0 (\text{mod } c)$, we select the subgoal through the following steps (as illustrated in Figure 2):

1) find the point l_n that is nearest to the current position,

$$n = \arg\min_{n \in \{1,...,f\}} \|\phi(s_t) - l_n\|_2. \tag{9}$$

2) select the point k steps after l_n in $(l_1, ..., l_f)$, where k is randomly selected from a fixed range.

By constantly selecting subgoals using the above way, the agent has a higher probability of reaching the frontier region. Upon approaching the desired region, the agent can select subgoals within nearby area using any exploratory strategy. Here a count-based method is employed. We record visit counts of nodes during interactions:

$$n(v) = \sum_{s \sim \mathcal{B}} I_v(\phi(s)), \tag{10}$$

where $v \in \mathcal{V}$ is the ID of node, and I denotes the indicator function, as described in Equation (8). Furthermore, we assume that subgoals belonging to the same node share the same visit count. When selecting subgoals, we first sample a set of candidates near $\phi(s_t)$, then we choose the one with the minimum visit count as subgoal sg_t. Through this method, more rarely visited subgoals can be explored.

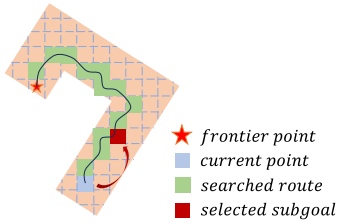


Figure 2: An illustration of subgoal selection of frontier-reaching episode. The blue grid is the current latent state. The red one is the selected subgoal.

An additional benefit of utilizing $\mathcal{M}$ is that the routes are near-optimal. Through discretization, sub-optimal trajectories are divided into smaller segments, many of which are locally near-optimal. By graph search, these segments are "stitched" together to form longer trajectories, thereby ensuring the resulting routes are globally near-optimal.

Hierarchical Self-imitation Learning

Inspired by the fact that the agent must traverse a series of subgoals before reaching the final goal, we devise a hierarchical self-imitation scheme aimed at enhancing exploitation and stabilizing the training process.

Our motivation is twofold. First, agents can utilize information not only within the same episode but also across episodes to refine themselves. Leveraging $\mathcal{M}$, transitions from different episodes can be stitched together, enabling higher probability of discovering viable trajectories toward goals. These transitions can guide the selection of subgoals or primitive actions. Second, UVFA estimates conditioned on nearby goals are generally more accurate than those con-

ditioned on distant goals. Therefore, when nearer goals are selected, better decisions are more likely to be made. Exploiting this, action choices conditioned on distant goals can be refined using those conditioned on nearer goals within the same viable transition sequence. This enables agents to approximately follow viable trajectories even under distant goal conditions, enhancing the reachability of distant goals.

For the high-level policy, we encourage the alignment of subgoals with targets along the path provided by $\mathcal{M}$, while for the low-level policy, actions conditioned on distant subgoals are expected to exhibit higher consistency with those conditioned on nearby subgoals. To be specific, suppose $R_I(s,g) = (l_1, l_2, ..., l_m)$ represents a route from the starting point $l_1 = \phi(s)$ to goal point $l_m = \phi(g)$ searched from $\mathcal{M}$, the high-level self-imitation loss can be expressed as:

$$\mathcal{L}_{hi,imi}(\theta_{hi}) = \mathbb{E}_{(s,g)\sim\mathcal{B}_{imi}}\big[\|\pi_{hi}(s,g;\theta_{hi}) - l_k(s,g)\|_2^2\big], \tag{11}$$

where $\mathcal{B}_{imi}$ denotes the imitation learning buffer and $l_k(s,g)$ is the k-th point in $R_I(s,g)$. In practice, we select k randomly within a range to ensure robustness.

While for the low-level policy, the loss can be written as:

$$\mathcal{L}_{lo,imi}(\theta_{lo}) = \mathbb{E}_{(s,g)\sim\mathcal{B}_{imi}}\Bigg[\sum_{i=0}^{|u|-1}\|\pi_{lo}(s,l_k(s,g);\theta_{lo}) - \mathbf{NG}(\pi_{lo}(s,l_{u[i]}(s,g);\theta_{lo}))\|_2^2\Bigg], \tag{12}$$

where u represents the index array of randomly selected, nearer subgoals within $R_I(s,g)$, and $|u|$ is the number of points within u. For instance, if we choose nearer subgoals (l_2, l_4), then $u = (2,4)$ and $|u| = 2$. The operator $\mathbf{NG}(\cdot)$ denotes the "no gradient" operation. Similar to the high-level self-imitation, the faraway index k is also chosen from a fixed range, while the indices $u[i], i = 0, ..., |u|-1$ are selected within a smaller range prior to k. The framework of the hierarchical self-imitation is illustrated in Figure 3.

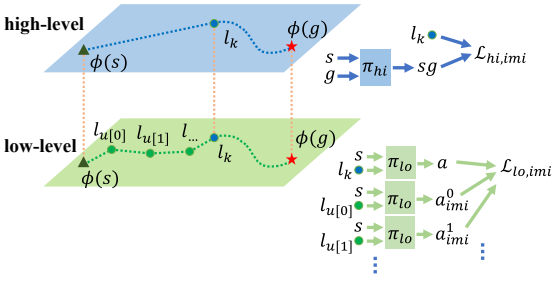


Figure 3: Illustration of the hierarchical self-imitation scheme. $\phi(s)$ and $\phi(g)$ are the representations of the starting state and the goal. $l_{...}$ are nodes on the route. For π_{hi}, l_k is the imitation target of $\pi_{hi}(s,g)$, while for π_{lo}, actions given by $\pi_{lo}(s,l_{u[i]}), i = 0, ..., |u|-1$ are targets of $\pi_{lo}(s,l_k)$.

As we previously noted, routes selected in $\mathcal{M}$ are near-optimal, implying that the self-imitation targets for both hierarchies are also near-optimal. For π_{hi}, targets are located near the subsequent waypoints leading to the final goal, while for π_{lo}, targets are primitive actions corresponding to locally near-optimal transitions. So the imitation losses encourage the agent to move along near-optimal transitions.

Training with ASI

The frontier-reaching module and the hierarchical self-imitation module are both independent of the current GCHRL framework, and they operate independently of one another, so they can be deployed separately. For policy training, the self-imitation loss can be incorporated as an additional term into the original loss. High-level and low-level losses can be written as follows, respectively:

$$\mathcal{L}_{hi}(\theta_{hi}) = -\mathbb{E}_{\pi_{hi}}\Big[\sum_{t'=0}^{\lfloor T/c\rfloor}\gamma^{t'}r_{hi,t'}\Big] + \alpha\mathcal{L}_{hi,imi}(\theta_{hi}), \tag{13}$$

$$\mathcal{L}_{lo}(\theta_{lo}) = -\mathbb{E}_{\pi_{lo}}\Big[\sum_{t=0}^{T-1}\gamma^{t}r_{lo,t}\Big] + \beta\mathcal{L}_{lo,imi}(\theta_{lo}), \tag{14}$$

where T is the number of timesteps in each episode, and α, β are balancing coefficients.

The pseudo-code for the entire learning process is presented in Algorithm 1 in the supplementary material, which can be found in our code repository. To enhance exploration, we assign an episode as a frontier-reaching episode with probability p, which is gradually increased throughout training. In each frontier-reaching episode, the primary purpose of the agent is to reach the frontier area quickly and subsequently explore surrounding regions. The agent searches a route from the starting point to the frontier area at the beginning and selects proper waypoints as subgoals. Upon approaching the desired area, the subgoal selection method switches to the count-based one. To enhance exploitation, both high- and low-level policies are fine-tuned by self-imitation with a fixed time interval. As the foundations of the above two modules, ϕ and $\mathcal{M}$ are also updated with a larger interval.

Experiments

Experimental Setup

Environments We evaluate ASI on a set of challenging long-horizon continuous control tasks base on Mujoco (Todorov, Erez, and Tassa 2012) simulator, including various Ant locomotion tasks. Specifically, we conduct our experiments in the following environments:

- AntMaze (U/S/W-shape): A series of maze tasks. An ant starts at a fixed point and aims to reach different goals.
- AntFourRooms: The ant starts at the bottom left room and aims to reach the top right room.
- HalfCheetahHurdle: A HalfCheetah robot tries to jump over a hurdle of height 0.35 and then reach the goal.

More details about the environments can be seen in Appendix A in the supplementary material (in our code repository).

Implementation We use SAC (Haarnoja et al. 2018) as the underlying algorithm for training two hierarchies of policies, following the approach of (Li et al. 2022). Additionally, the choice of $\lambda(s)$ in Equation (6) also aligns with (Li et al. 2022). For the U-shaped AntMaze task, we set the

maximum number of timesteps to 500; for S- and W-shaped AntMaze tasks, the maximum number of timesteps is set to 1000; for AntFourRooms and HalfCheetahHurdle, the maximum number of timesteps is also 1000. For Ant tasks, the probability of assigning a frontier-reaching episode is 0.2, which is gradually increased to 0.4 during training. As for HalfCheetahHurdle, this probability is 0.1. The balancing coefficients α and β of high- and low- level self-imitation learning are set to 0.01 and 0.001, respectively. For each environment, we run 5 parallel experiments using 5 different random seeds. We provide more implementation details in Appendix B in the supplementary material (in our code repository).

Evaluation For each task, we run 10 episodes in the corresponding test environment and calculate the success rate after every 50 training episodes. We report the average success rate over 5 runs and the corresponding 95% confidence interval as the performance metric.

Baselines We compare ASI with the following baselines:

• LESSON (Li et al. 2021): A GCHRL method that learns a subgoal space based on states with slow dynamics.

• HESS (Li et al. 2022): A GCHRL method that learns a representation space with stable regularization and uses an active exploration strategy.

• HRAC (Zhang et al. 2020): A GCHRL method that directly uses the pre-defined *xy* space as the subgoal space, which is a very strong prior.

• SAC (Haarnoja et al. 2018): A typical non-HRL approach that enhances policy learning with entropy.

For all tasks, we report results of ASI combined with HESS. For convenience of expression, we use the label "ASI" to denote "ASI+HESS". Indeed, ASI is a flexible framework and can be integrated with any GCHRL method, as the frontier-reaching and self-imitation modules are both independent of current approaches.

Comparative Analysis

As shown in Figure 4, ASI achieves good results in all tasks. In the U-shaped AntMaze environment, ASI converges more quickly than LESSON and HESS, and learns faster than HRAC at the early stage. In the complex S- and W-shaped AntMaze environments, ASI significantly outperforms all the baselines. This is closely related to the exploration mechanism of ASI. With the active frontier-reaching module, the agent can explore more unseen states within the same number of timesteps. Moreover, the hierarchical self-imitation learning module assists the agent in successfully approaching frontier regions, leading to higher exploration efficiency and faster policy learning. In the AntFourRooms task, ASI has higher exploration efficiency and outperforms HESS and LESSON. With a pre-defined subgoal space, the learning process of HRAC is more stable and a bit faster. ASI does not use such a pre-defined subgoal space, so it takes more time to learn a good subgoal space and align it with policies. But it can still achieve similar results. In the HalfCheetahHurdle task, ASI also exhibits competitive performance.

In comparison, LESSON only reduces the subgoal space but lacks effective exploration methods, making it less advantageous for dealing with complex tasks. HRAC utilizes an adjacency network to measure the reachability between states and introduces an adjacency loss to curtail subgoal selection. This overemphasizes exploitation but ignores exploration, resulting in relatively poor learning efficiency in complicated environments. HESS introduces an active subgoal selection method based on novelty and reachability, making itself more advantageous for exploring the environment. However, this kind of exploration is less efficient than the frontier-reaching approach. The agent spends more timesteps within the explored area, leaving fewer opportunities for exploring unseen states. Limited by non-HRL architectures, SAC lacks the capacity for tackling complex and sparse-reward tasks, so it almost cannot learn effective policies with several millions of timesteps.

Ablation Study

Ablation Study on FR and SI Components To verify the effectiveness of the proposed frontier-reaching and hierarchical self-imitation modules, we conduct ablation studies on these two modules. For comprehensive comparison, we measure exploration efficiency using the average coverage ratio of the whole *xy* space, while assessing exploitation performance using the average success rate.

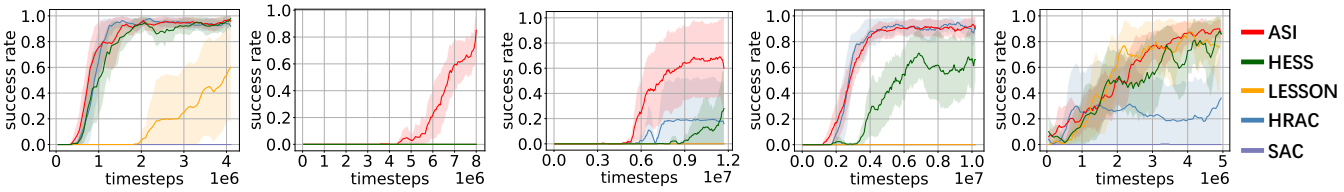
The exploration performance among HESS, ASI without FR, ASI without SI, and ASI in the S-shaped AntMaze environment is shown in Figure 5(a). Comparing "ASI" and "ASI without FR", there is an obvious difference between the two coverage curves. And HESS with FR (i.e., "ASI without SI") achieves significantly faster coverage ratio growth compared to the baseline "HESS". These results demonstrate the FR module's effectiveness in accelerating exploration. Additionally, the exploration efficiency of ASI surpasses that of "ASI without SI". This indicates that the SI module can successfully facilitate FR-based exploration.

Figure 5(b) illustrates the exploitation performance comparison among HESS, ASI without FR, ASI without SI, and ASI in the S-shaped AntMaze environment. The results reveal that removing the SI module leads to a significant decrease in the average success rate and larger performance variance, demonstrating the effectiveness of SI. Moreover, removing the FR module also leads to a performance decrease, but not so significant as that caused by SI.

Notably, when we analyse Figure 5(b) together with 5(a), we can observe that the coverage ratio curves of "ASI" and "ASI without SI" do not differ much, but there exists obvious difference between the two success rate curves. This further proves that the employment of SI module leads to better exploitation of past experiences.

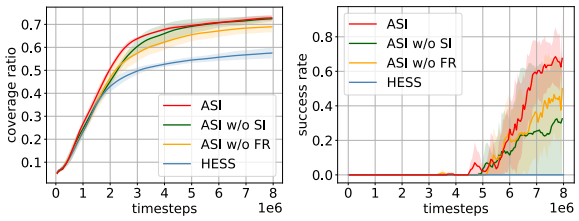
Ablation Study on Hyperparameters To investigate the influence of different hyperparameters on the experimental results, we conduct ablation studies on several hyperparameters, including frontier-reaching assignment probability and coefficients of high-level and low-level self-imitation losses.

Figure 6 shows coverage ratio curves under different frontier-reaching assignment probabilities. We can observe that the coverage ratio grows faster under non-zero frontier-reaching assignment probability, and neither too high nor too



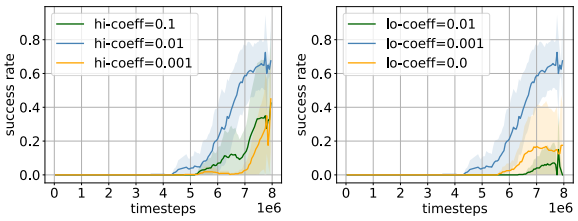
(a) AntMaze (U-shape) (b) AntMaze (S-shape) (c) AntMaze (W-shape) (d) AntFourRooms (e) HalfCheetahHurdle

Figure 4: Learning curves on a series of Ant and HalfCheetah locomotion tasks. The x axis denotes the number of timesteps and the y axis denotes the success rate in 10 test episodes. Each curve is the average of 5 runs. The shaded regions represent 95% confidence intervals. We smooth all the curves equally for visual clarity.



(a) Coverage ratio (b) Success rate

Figure 5: Ablation studies on FR ans SI modules.



(a) High-level coefficient (b) Low-level coefficient

Figure 7: Ablation studies on coefficients of high-level and low-level self-imitation losses.

low probabilities perform well enough. Too high probability overemphasizes exploration but lacks attention to exploitation, making the agent harder to accurately reach selected subgoals. This is also bad for reaching frontier regions and exploring unseen states. On the other hand, too low probability brings limited exploration ability gain, which is also not effective enough.

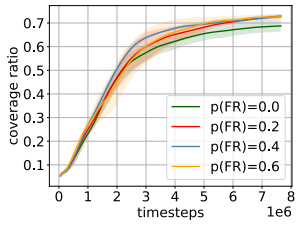


Figure 6: Ablation study on FR probability.

Figure 7 shows the success rate curves under different coefficients of high-level and low-level self-imitation losses. In the ablation study of high-level coefficients (Figure 7(a)), the low-level coefficient is kept the same (0.001); while in the ablation study of low-level coefficients (Figure 7(b)), the high-level coefficient is kept the same (0.01).

Results in Figure 7 indicate that both high- and low-level self-imitation coefficients cannot be too big or too small. Too big coefficients lead to stronger guidance, but as this guidance may not be optimal, the overall effect may be not good enough. And too small coefficients can only provide limited guidance, which is also not effective enough. So for all comparative experiments, we set high-level coefficient $\alpha = 0.01$ and low-level coefficient $\beta = 0.001$.

Discussion

In addition to HESS, ASI is compatible with other subgoal selection strategies. Besides, ASI does not depend on any specific subgoal representation method. This means that ASI can serve as a plug-in and can be integrated with many existing methods.

Conclusion

We propose ASI, an effective scheme for enhancing exploration and exploitation in GCHRL. We leverage the adjacency information between subgoal representations and maintain a latent graph to capture these information. Based on the latent graph, we perform active frontier-reaching to accelerate exploration and take advantage of hierarchical self-imitation learning to facilitate both frontier-reaching and policy training. These two modules significantly improve performances on challenging tasks.

One potential issue is the time consumption of constructing the graph and searching the routes. We point that although these procedures require some extra time, ASI achieves comparable overall training time consumption (from start to convergence) to baselines in complex environments (e.g., S-shaped AntMaze) due to its high data efficiency. Besides, ASI does not require the graph and the routes at deployment time.

ASI also has some limitations. In dynamic environments such as AntPush, the routes given by the latent graph may be invalid, potentially leading the agent to learn wrong information. Thus, an interesting future work is to adapt ASI to accommodate such environments.

References

Andrychowicz, M.; Crow, D.; Ray, A.; Schneider, J.; Fong, R.; Welinder, P.; McGrew, B.; Tobin, J.; Abbeel, P.; and Zaremba, W. 2017. Hindsight Experience Replay. In *Advances in Neural Information Processing Systems*, volume 30, 5055–5065.

Bacon, P.-L.; Harb, J.; and Precup, D. 2017. The Option-Critic Architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 1726–1734.

Chane-Sane, E.; Schmid, C.; and Laptev, I. 2021. Goal-Conditioned Reinforcement Learning with Imagined Subgoals. In *International Conference on Machine Learning*, volume 38, 1430–1440.

Dayan, P.; and Hinton, G. E. 1992. Feudal Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 5, 271–278.

Dijkstra, E. W. 1959. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1: 269–271.

Ding, Y.; Florensa, C.; Abbeel, P.; and Phielipp, M. 2019. Goal-conditioned Imitation Learning. In *Advances in Neural Information Processing Systems*, volume 32, 15324–15335.

Dwiel, Z.; Candadai, M.; Phielipp, M.; and Bansal, A. K. 2019. Hierarchical Policy Learning is Sensitive to Goal Space Design. *arXiv preprint arXiv:1905.01537*.

Eysenbach, B.; Salakhutdinov, R. R.; and Levine, S. 2019. Search on the Replay Buffer: Bridging Planning and Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 32, 15246–15257.

Ghosh, D.; Gupta, A.; and Levine, S. 2019. Learning Actionable Representations with Goal Conditioned Policies. In *International Conference on Learning Representations*.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *International Conference on Machine Learning*, 1861–1870.

Ho, J.; and Ermon, S. 2016. Generative Adversarial Imitation Learning. In *Advances in neural information processing systems*, volume 29, 4572–4580.

Huang, Z.; Liu, F.; and Su, H. 2019. Mapping State Space using Landmarks for Universal Goal Reaching. In *Advances in Neural Information Processing Systems*, volume 32, 1942–1952.

Kalashnikov, D.; Irpan, A.; Pastor, P.; Ibarz, J.; Herzog, A.; Jang, E.; Quillen, D.; Holly, E.; Kalakrishnan, M.; Vanhoucke, V.; et al. 2018. Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation. In *Conference on Robot Learning*, 651–673.

Kim, J.; Seo, Y.; Ahn, S.; Son, K.; and Shin, J. 2023. Imitating Graph-Based Planning with Goal-Conditioned Policies. In *International Conference on Learning Representations*, volume 11.

Kim, J.; Seo, Y.; and Shin, J. 2021. Landmark-Guided Subgoal Generation in Hierarchical Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 34, 28336–28349.

Kingma, D. P. 2014. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*.

Kulkarni, T. D.; Narasimhan, K.; Saeedi, A.; and Tenenbaum, J. 2016. Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation. In *Advances in Neural Information Processing Systems*, volume 30, 3682–3690.

Lample, G.; and Chaplot, D. S. 2017. Playing FPS Games with Deep Reinforcement Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2140–2146.

Lee, S.; Kim, J.; Jang, I.; and Kim, H. J. 2022. DHRL:AGraph-Based Approach for Long-Horizon and Sparse Hierarchical Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 35, 13668–13678.

Levine, S.; Finn, C.; Darrell, T.; and Abbeel, P. 2016. End-to-End Training of Deep Visuomotor Policies. *Journal of Machine Learning Research*, 17(39): 1–40.

Levy, A.; Konidaris, G. D.; Platt, R. W.; and Saenko, K. 2017. Learning Multi-Level Hierarchies with Hindsight. In *International Conference on Learning Representations*.

Li, S.; Zhang, J.; Wang, J.; Yu, Y.; and Zhang, C. 2022. Active Hierarchical Exploration with Stable Subgoal Representation Learning. In *International Conference on Learning Representations*.

Li, S.; Zheng, L.; Wang, J.; and Zhang, C. 2021. Learning Subgoal Representations with Slow Dynamics. In *International Conference on Learning Representations*.

Liu, E. Z.; Keramati, R.; Seshadri, S.; Guu, K.; Pasupat, P.; Brunskill, E.; and Liang, P. 2020. Learning Abstract Models for Strategic Exploration and Fast Reward Transfer. *arXiv preprint arXiv:2007.05896*.

Luo, Y.; Sun, F.; Ji, T.; and Zhan, X. 2024. Bidirectional-Reachable Hierarchical Reinforcement Learning with Mutually Responsive Policies. In *Reinforcement Learning Conference*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M. A.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level Control through Deep Reinforcement Learning. *Nature*, 518(7540): 529–533.

Nachum, O.; Gu, S.; Lee, H.; and Levine, S. 2018. Data-Efficient Hierarchical Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 31, 3307–3317.

Nachum, O.; Gu, S.; Lee, H.; and Levine, S. 2019. Near-Optimal Representation Learning for Hierarchical Reinforcement Learning. In *International Conference on Learning Representations*.

Nair, S.; and Finn, C. 2020. Hierarchical Foresight: Self-Supervised Learning of Long-Horizon Tasks via Visual Subgoal Generation. In *International Conference on Learning Representations*.

Nasiriany, S.; Pong, V.; Lin, S.; and Levine, S. 2019. Planning with Goal-Conditioned Policies. In *Advances in Neural Information Processing Systems*, volume 33, 14843–14854.

Pateria, S.; Subagdja, B.; Tan, A.-H.; and Quek, C. 2021. Hierarchical Reinforcement Learning: A Comprehensive Survey. *ACM Computing Surveys*, 54(5): 1–35.

Péré, A.; Forestier, S.; Sigaud, O.; and Oudeyer, P.-Y. 2018. Unsupervised Learning of Goal Spaces for Intrinsically Motivated Goal Exploration. In *International Conference on Learning Representations*.

Pitis, S.; Chan, H.; Zhao, S.; Stadie, B.; and Ba, J. 2020. MaximumEntropy Gain Exploration for Long Horizon Multi-goal Reinforcement Learning. In *International Conference on Machine Learning*, volume 37, 7750–7761.

Puterman, M. L. 1990. Markov Decision Processes. *Handbooks in Operations Research and Management Science*, 2: 331–434.

Rafati, J.; and Noelle, D. C. 2019. Learning Representations in Model-Free Hierarchical Reinforcement Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 10009–10010.

Röder, F.; Eppe, M.; Nguyen, P. D.; and Wermter, S. 2020. Curious Hierarchical Actor-Critic Reinforcement Learning. In *International Conference on Artificial Neural Networks*, volume 29, 408–419.

Schaul, T.; Horgan, D.; Gregor, K.; and Silver, D. 2015. Universal Value Function Approximators. In *International Conference on Machine Learning*, volume 37, 1312–1320.

Schmidhuber, J.; and Wahnsiedler, R. 1992. Planning Simple Trajectories using Neural Subgoal Generators. In *Proceedings of the International Conference on Simulation of Adaptive Behavior*, volume 2, 196–202.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M. I.; and Abbeel, P. 2015. High-Dimensional Continuous Control using Generalized Advantage Estimation. *arXiv preprint arXiv:1506.02438*.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; Lillicrap, T. P.; Simonyan, K.; and Hassabis, D. 2018. A General Reinforcement Learning Algorithm that Masters Chess, Shogi, and Go through Self-play. *Science*, 362(6419): 1140 – 1144.

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

Vezhnevets, A. S.; Osindero, S.; Schaul, T.; Heess, N.; Jaderberg, M.; Silver, D.; and Kavukcuoglu, K. 2017. FeUdal Networks for Hierarchical Reinforcement Learning. In *International Conference on Machine Learning*, volume 34, 3540–3549.

Wang, V. H.; Wang, T.; Yang, W.; Kamarainen, J.-K.; and Pajarinen, J. 2024. Probabilistic Subgoal Representations for Hierarchical Reinforcement Learning. In *International Conference on Machine Learning*, volume 41, 51755–51770.

Zhang, L.; Yang, G.; and Stadie, B. C. 2021. World Model as a Graph: Learning Latent Landmarks for Planning. In *International Conference on Machine Learning*, volume 38, 12611–12620.

Zhang, M.; Vikram, S.; Smith, L.; Abbeel, P.; Johnson, M.; and Levine, S. 2019. SOLAR: Deep Structured Representations for Model-Based Reinforcement Learning. In *International Conference on Machine Learning*, volume 36, 7444–7453.

Zhang, T.; Guo, S.; Tan, T.; Hu, X.; and Chen, F. 2020. Generating Adjacency-Constrained Subgoals in Hierarchical Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 33, 21579–21590.

Zhu, Y.; Mottaghi, R.; Kolve, E.; Lim, J. J.; Gupta, A.; Fei-Fei, L.; and Farhadi, A. 2017. Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning. In *IEEE International Conference on Robotics and Automation*, 3357–3364.