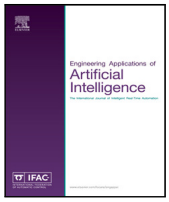
Engineering Applications of Artificial Intelligence 126 (2023) 107130

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



Hierarchical reinforcement learning with adaptive scheduling for robot control☆

Zhigang Huang, Quan Liu *, Fei Zhu

School of Computer Science and Technology, Soochow University, 215006, Suzhou, Jiangsu, PR China

ARTICLE INFO

Dataset link: HAS code (Original data)

Keywords:
Hierarchical reinforcement learning
Exploration and exploitation
Scheduling
Sparse reward

ABSTRACT

Conventional hierarchical reinforcement learning (HRL) relies on discrete options to represent explicitly distinguishable knowledge, which may lead to severe performance bottlenecks. It is possible to represent richer knowledge through continuous options, but reliable scheduling methods are lacking. To design an available scheduling method for continuous options, in this paper, the hierarchical reinforcement learning with adaptive scheduling (HAS) algorithm is proposed. Its low-level controller learns diverse options, while the high-level controller schedules options to learn solutions. It achieves an adaptive balance between exploration and exploitation during the frequent scheduling of continuous options, maximizing the representation potential of continuous options. It builds on multi-step static scheduling and makes switching decisions according to the relative advantages of the previous and the estimated continuous options, enabling the agent to focus on different behaviors at different phases of the task. The expected t-step distance is applied to demonstrate the superiority of adaptive scheduling in terms of exploration. Furthermore, an interruption incentive based on annealing is proposed to alleviate excessive exploration during the early training phase, accelerating the convergence rate. Finally, we apply HAS to robot control with sparse rewards in continuous spaces, and develop a comprehensive experimental analysis scheme. The experimental results not only demonstrate the high performance and robustness of HAS, but also provide evidence that the adaptive scheduling method has a positive effect both on the representation and option policies.

1. Introduction

Reinforcement learning (RL) (Cobbe et al., 2021) is a method for encouraging the agent to solve problems by collecting rewards. However, it is difficult for conventional RL algorithms to obtain positive rewards on sparse reward tasks due to the lack of a reliable exploration mechanism (Wagenmaker et al., 2022). To improve the agent's exploration, intrinsic motivation methods (Aubret et al., 2023) have been proposed. The count-based method (Tang et al., 2017) calculates the number of state visits, which encourages the agent to visit less visited states. The state entropy method (Li et al., 2021a,b) utilizes the state entropy as the intrinsic reward, which encourages the state visits to approximate the ideal probability distribution. The hindsight method (Andrychowicz et al., 2017) assumes that the visited state is a subgoal, guaranteeing the agent's reachability. Hierarchical reinforcement learning (HRL) (Dukkipati et al., 2022; Bagaria and Konidaris, 2020) employs temporal abstraction techniques to represent multi-level knowledge, achieving cross-temporal behavior. The first two belong to the guided exploration by the environment, while the third belongs to the agent's active exploration of the environment. In this paper, we focus on HRL for an adequate balance between exploration and exploitation.

HRL algorithms typically employ two-layer network structures (Cho et al., 2022). Through the value function and the interruption function, high-level and low-level networks can be connected, and the option policy is influenced by both environmental and intrinsic rewards (Klissarov and Precup, 2021; Jain et al., 2021). For exploring an environment without prior knowledge, mutual information-based intrinsic rewards are used to learn task-agnostic options (Dai et al., 2021; Baumli et al., 2021). The low-level controller represents diverse options based on intrinsic motivation (Abramowitz and Nitschke, 2022), while the high-level controller schedules options based on environmental rewards (Jain et al., 2021). However, most of them rely on **discrete**

☆ This work has been supported by the National Natural Science Foundation of China (61772355), Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD).

* Corresponding author.
E-mail addresses: 20204027016@stu.suda.edu.cn (Z. Huang), quanliu@suda.edu.cn (Q. Liu), zhufei@suda.edu.cn (F. Zhu).

https://doi.org/10.1016/j.engappai.2023.107130
Received 26 April 2023; Received in revised form 14 August 2023; Accepted 8 September 2023
Available online 22 September 2023

options (Ding and Zhu, 2022). Although the agent can behave cross-temporally, its exploration ability is still restricted due to the limited number of discrete options (Machado et al., 2017). It is unable to perform diverse trajectories to cover fine-grained state spaces and to acquire rich scheduling combinations, resulting in severe performance bottlenecks. For sparse reward problems with high exploration requirements, such as robot control, the situation is particularly dire.

To achieve sufficient exploration and extend the application region of HRL, it is realistic and natural to choose **continuous options** (Rana et al., 2022) to represent richer knowledge. Their intra-option policy is a continuous function, so the update of option values can be generalized to adjacent spaces. The agent executes similar actions in the same state by taking similar continuous options. They provide the agent with more possible scheduling combinations and more opportunities to learn near-optimal policies. However, conventional scheduling methods are not suitable for continuous options. Either they are sensitive to the scheduling parameters (Khetarpal et al., 2020), or the switching frequency cannot be reasonably controlled, even leading to the hierarchical degradation dilemma (Bacon et al., 2017). Therefore, our objective is to develop a practical scheduling method that maximizes the superiority of continuous options without prior knowledge.

Based on this principle, we propose the hierarchical reinforcement learning with adaptive scheduling (HAS) algorithm. Its core idea is to balance exploration and exploitation through adaptive scheduling. Its switching condition is determined by the relative advantages of the previous and estimated continuous options, and each switch is based on a multi-step static scheduling process. To demonstrate its positive effects, we construct a comprehensive experimental analysis scheme in terms of adaptive mechanism, representation, and option policies. Specifically, the main contributions of this paper are as follows:

- We propose an adaptive scheduling method to balance exploration and exploitation during the frequent scheduling of continuous options
- We design an interruption incentive to alleviate the excessive exploration dilemma, accelerating the convergence rate.
- Our algorithm effectively solves robot control problems with sparse rewards in continuous spaces, which is challenging for conventional HRL.

The remainder of the paper is organized as follows. Section 2 reports on the background of intrinsic motivation and HRL, including options' characteristics, representation, and scheduling. Section 3 presents our proposed approach in detail, which is divided into four parts: the hierarchical framework, the scheduling method, the measure of exploration, and the interruption incentive. The experimental results of our method are reported in Section 4. Finally, Section 5 is our conclusion.

2. Background

It is difficult for traditional RL to solve large-scale sparse reward problems, such as robot control (Chua et al., 2018). Although the ϵ-greedy policy (Hasselt, 2010) and noise (Fujimoto et al., 2018) can make the policy more flexible, they cannot motivate the agent to explore a wider area. To improve the agent's ability to explore, intrinsic motivation methods (Savinov et al., 2018) are proposed. Not only can we design additional rewards based on intrinsic motivation measures to encourage exploration (Brunner et al., 2018), but we can also construct a hierarchical architecture of policies to achieve acting with temporal abstraction (Cho et al., 2022).

In this section, we introduce several intrinsic motivation measures that are commonly used in deep reinforcement learning (DRL), and then discuss the differences between our method and these methods. Afterward, we introduce the three main components of HRL: option, representation, and scheduling. These form the basis of our algorithm. Some of the most important HRL algorithms serve as our baselines.

2.1. Intrinsic motivation measure

The frequency of state visits (Ermolov and Sebe, 2020) is one of the most important measures of intrinsic motivation. Count-Based methods (Bellemare et al., 2016; Tang et al., 2017) guide the agent towards unvisited or infrequently visited regions by calculating the (pseudo) number of state visits. However, they are difficult to extend to large-scale state spaces. Based on the forward and inverse networks, ICM (Pathak et al., 2017) and ADM (Choi et al., 2018) build a curiosity model that employs the state prediction error as the intrinsic reward. The state visits determine the prediction error of the forward network. A state with a large prediction error is more likely to be visited. RND (Burda et al., 2018) develops a deterministic, stochastically initialized network. It can also alleviate problems associated with the stochasticity of the objective fitting function and the deficiency of expressiveness. In addition, there are methods that consider the learning progress and reachability of goal states. For instance, NGU (Badia et al., 2020) uses a self-supervised inverse dynamics model to train the embeddings of the nearest neighbor lookup, biasing the novelty signal towards what the agent can control. Novelty (Tao et al., 2020) generates intrinsic rewards that are based on the distance of nearest neighbors in the low-dimensional representation space.

Alternatively, learning the ideal state distribution (Lee et al., 2021) is also a popular method of intrinsic motivation. SMM (Lee et al., 2021) employs Kullback–Leibler (KL) divergence to measure the difference between a parametric policy and an ideal distribution. Skew-fit (Pong et al., 2020) weights infrequent states and trains the generative model with weighted samples. The goals from the generative model are then used to obtain a state distribution with higher entropy. OMEGA (Pitis et al., 2020) explores the boundaries of visited goals by learning a density model. LESSON (Li et al., 2020b) learns state distributions with slow features to approximate a uniform distribution. Some of these algorithms are based on a hierarchical structure (Li et al., 2021b). They differ from ours in that they focus primarily on reachability, i.e. exploring the entire state space. While our primary goal is to achieve near-optimal solutions by balancing exploration and exploitation.

2.2. Hierarchical reinforcement learning

HRL is an intrinsic motivation method based on temporal abstraction (Li et al., 2020a). We illustrate a generic HRL structure as shown in Fig. 1. Intuitively, we can only select a finite number of discrete options from the discrete distribution (left) (Hou et al., 2020), while sampling an infinite number of continuous options from the continuous distribution (right). From the perspective of representation and scheduling, the discrete option set can only provide a limited representation result (Mayr et al., 2022), resulting in a tortuous path. Even if their number could be increased, it would be impractical for the option policy to traverse each option value. In contrast, with a continuous option set, the agent can acquire a smooth path from a diverse representation result. The trajectories can cover a wide range of fine-grained spaces. In the following, we introduce important work about representation and scheduling.

Representation. Representation refers to the motivation process by which options are endowed with varying degrees of knowledge (Bacon and Precup, 2018). Currently, the dominant representation method applies mutual information (Salge et al., 2014; Dai et al., 2021) to establish the guiding relationship between options (including discrete options and continuous options) and trajectories (including various combinations of states and actions). VIC (Gregor et al., 2016) first introduced mutual information (Klyubin et al., 2004) to generate the inferred relationship between options and states. DIAYN (Eysenbach et al., 2018) created the inferred relationship between options, states, and actions. DADS (Sharma et al., 2019) discovers predictable options while learning their dynamics. AdInfo (Osa et al., 2019) provided

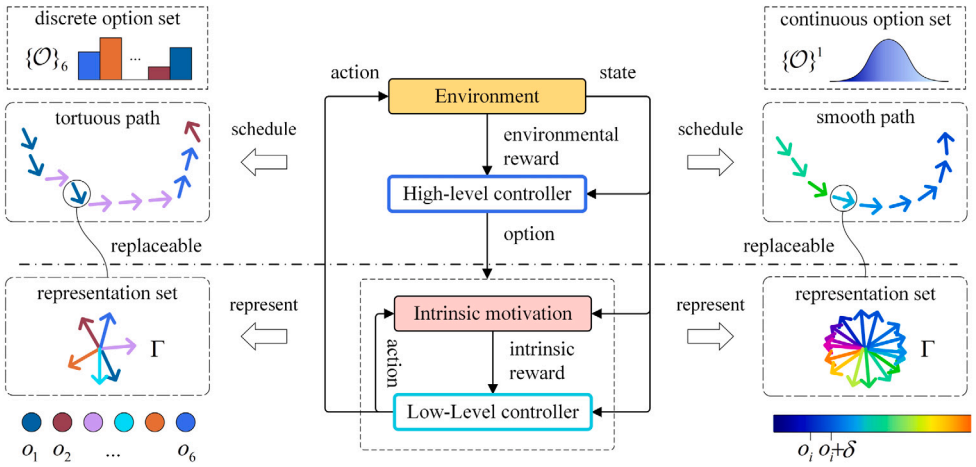


Fig. 1. A generic HRL structure. The high-level controller generates options according to environmental rewards, and schedules the options to generate paths. The low-level controller generates actions according to options and intrinsic motivation to establish the guiding relationship between options and trajectories. On the left are the representation and scheduling results of discrete options, and on the right, are continuous options. The trajectories marked with black circles in the "path" can be replaced with any option in the "set".

options with the semantics of state–action spaces through mutual information. HIDIO (Zhang et al., 2021) employed continuous options for the first time and compared the effects of different mutual information forms. HAS applies a typical form of mutual information consisting of options and states (Karl et al., 2019). It can be written:

$$I(O;S) = -H(O|S) + H(O) \tag{1}$$
$$= \mathbb{E}_{o,s'\sim p(o,s')}[\log p(o|s')] - \mathbb{E}_{o\sim p(o)}[\log p(o)]$$

Scheduling. Scheduling is a control mechanism acting on the option policy. It can either implicitly affect the output of the option policy (Riemer et al., 2020) or explicitly control the execution cycle of the option policy (Florensa et al., 2017). The most representative methods for these two control mechanisms are dynamic scheduling (Harutyunyan et al., 2017) and static scheduling (Li et al., 2019b). Dynamic scheduling is based on the option policy gradient theorem about the termination function (Bacon et al., 2017; Harb et al., 2018). The termination function $\beta \in [0, 1]$ is trained using a neural network. Since a single option can be executed for a long period of time, the intra-option learning method (Sutton et al., 1999) is proposed. The value functions affected by the termination function are represented by the arrival value function $U^{\pi_\mathcal{O}}(s', o)$:

$$U^{\pi_\mathcal{O}}(s', o) = (1 - \beta(s'))Q^{\pi_\mathcal{O}}(s', o) + \beta(s')V^{\pi_\mathcal{O}}(s') \tag{2}$$

Static scheduling is based on the MSA concept (Schoknecht and Riedmiller, 2003), which switches options at each intra-option step k ($k \geq 1$). It emphasizes the stability of scheduling. SNN4HRL (Florensa et al., 2017) and AdInfo (Osa et al., 2019) switch the option at each time step. DIAYN (Eysenbach et al., 2018) and DADS (Sharma et al., 2019) select different values of k for different tasks. HAAR (Li et al., 2019b) uses a gradually decreasing intra-option step k. Our method applies both dynamic and static scheduling. We improve the traditional dynamic scheduling to accommodate the characteristics of continuous options.

Meta reinforcement learning (MRL) (Igl et al., 2020; Song et al., 2019) also involves the scheduling of sub-policies. For instance, Frans et al. (2017) learns shareable low-level policies from task distribution. Then, based on static scheduling, the agent executes the low-level policy on a new task, while fine-tuning the high-level policy. However, MRL focuses on learning consistent behavior from task distributions to quickly adapt to new tasks (Li et al., 2019a; Fakoor et al., 2019). Option learning is typically accompanied by task switching (Chua et al., 2023). While HRL focuses on task decomposition (Infante et al., 2022) or the temporal abstraction of agent behavior (Mayr et al., 2022) to improve exploration in large-scale problem and sparse reward tasks.

3. Adaptive scheduling algorithm

Frequent switching is essential to take full advantage of the representation superiority of continuous options. It can also reduce the detrimental effects of redundant continuous options on policy learning. However, too frequent switching is not conducive to stable performance and may even result in a hierarchical degradation dilemma. Thus, we propose the hierarchical reinforcement learning with adaptive scheduling (HAS) algorithm to address these issues. In this section, the framework of HAS, an adaptive scheduling method, and an interruption incentive are discussed in detail. Meanwhile, the exploration capability of different scheduling methods is measured using the expected t-step distance.

3.1. Framework of HAS

Fig. 2 illustrates the structure of HAS. It shows how the high-level controller of the robotic arm generates an option, followed by the low-level controller executing a sequence of actions guided by this option to push the box. The option policy $\pi_\mathcal{O}$ generates an **estimated option** $o^+ \leftarrow \pi_\mathcal{O}(\cdot|s)$ in a state s. The intra-option policy π_o generates an action $a \leftarrow \pi_o(\cdot|s, o)$ in an extended state (s, o). The agent performs interrupt determination every **intra-option step** k time steps. The termination function β determines an **activated option** $o \leftarrow \beta(\cdot|s, o^+, o^-)$ in a state s according to a **previous option** o^- and the estimated option o^+. The output of the termination function represents the probability of selecting the previous option. A bool value is obtained by sampling from this output probability. When the value is True, the switch takes place, and the estimated option o^+ will be regarded as an activated option o, otherwise, the previous option o^- will continue to be used. η denotes the interruption incentive term acting on the termination function β. In this framework, intra-option step k, termination function β, and interruption incentive η together play the role of scheduling.

Fig. 3 illustrates the training process of HAS. It applies SAC as the underlying algorithm, so its training process follows the off-policy method. Since the high-level and low-level controllers have separate network structures, the rollout information of the agent is stored in two experience buffers. The high-level experience unit contains the activated option o, the transition state $(s, \tilde{s}')$ of the uninterrupted time step, and the corresponding accumulated environmental rewards R. The low-level experience unit contains the activated option o and the transition state s' of a single time step. After batch sampling, high-level experience $(s, o, \tilde{s}', R)$ and low-level experience (o, s') are used to train

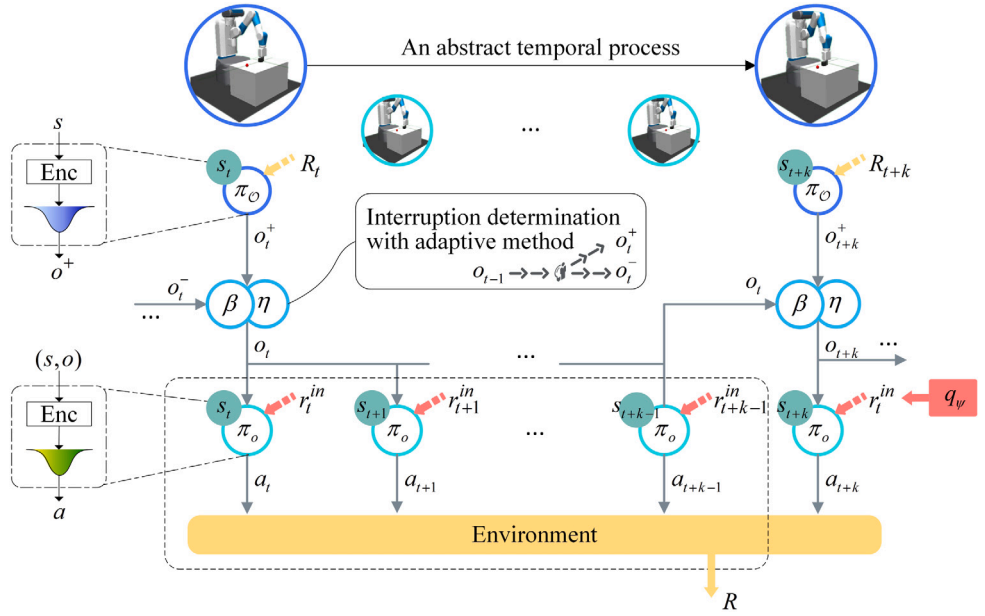


Fig. 2. The Framework of HAS. At time t, $\pi_{\mathcal{O}}$ selects o_t^+ in s_t. Next, β and η jointly determine o_t according to o_t^+ and o_t^-. Then, π_o selects a_t in (s_t, o_t). An agent interacts with the environment and reaches s_{t+1}. Afterward, π_o selects actions in new extended states with fixed o_t until $\pi_{\mathcal{O}}$ is executed again at time $t+k$. Repeat the process until the task has been completed or the maximum time step has been reached. Note that o_t and o_{t+1}^- are equivalent.

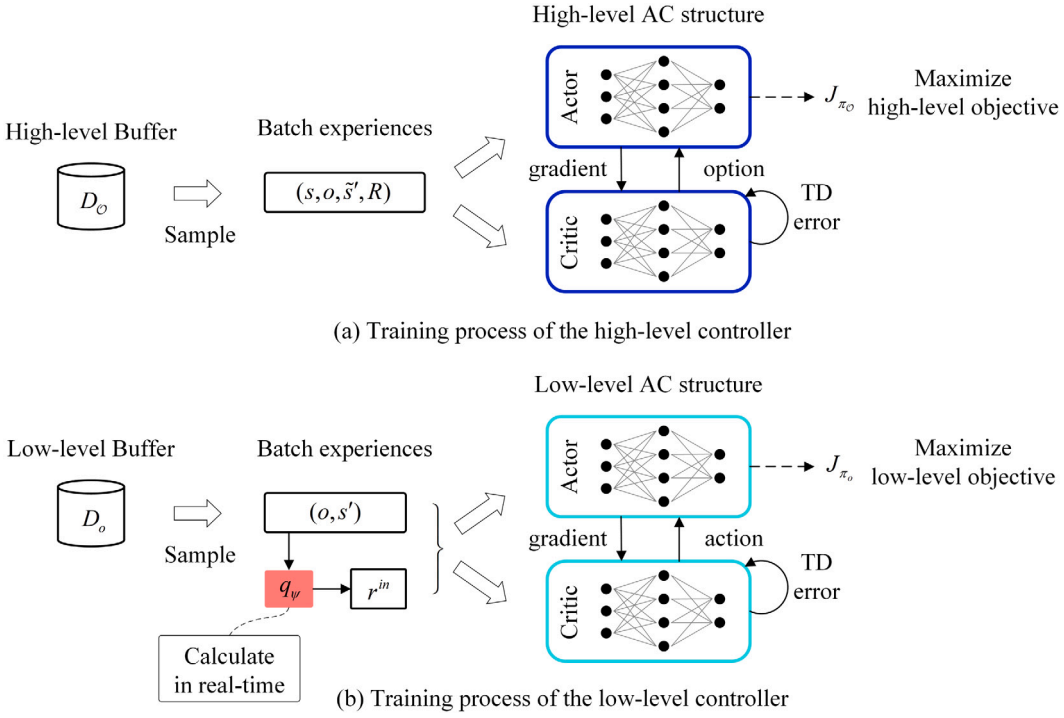


Fig. 3. The training process of HAS. There are independent experience buffers and AC (actor–critic) networks for the high-level and low-level controllers. Our experimental environments return numerical states, so AC networks apply fully connected network structures. The number of AC hidden layers and the number of neurons are given in Appendix B.

high-layer policy $\pi_{\mathcal{O}}$ and low-level policy π_o, respectively. The training details can be seen in Algorithm 1.

The goal of the high-level controller is to maximize the cumulative environmental reward. The high-level objective function $J_{\pi_{\mathcal{O}}}$ can be written:

$$J_{\pi_{\mathcal{O}}} = \mathbb{E}_{\pi_{\mathcal{O}}}\left[\sum_{t=0}^{T} \gamma^{t/\tilde{k}} R_t + \alpha H_{\mathcal{O},\beta}\right], \ where \ R_t = \sum_{l=t}^{t+\tilde{k}} r_l^{env} \tag{3}$$

where R_t denotes the cumulative environmental rewards from time t to $t+\tilde{k}$. $\tilde{k}$ is a multiple of k, denoting the time step during which the continuous option is executed uninterrupted. t takes values in the set of intra-option steps $t \sim \{0, \tilde{k}_1, \tilde{k}_2, \cdots\}$, and T is the maximum episode step. The discount of r^{env} is disregarded in consideration of sparse rewards. α is the temperature coefficient. $H_{\mathcal{O},\beta}$ denotes the combined entropy of $\pi_{\mathcal{O}}$ and β:

$$H_{\mathcal{O},\beta} = H_{\mathcal{O}} + H_{\beta} = \log \pi_{\mathcal{O}}(o^+|s) + \log \beta \tag{4}$$

The goal of the low-level controller is to maximize the intrinsic reward based on mutual information. The low-level objective function J_{π_o} can be written:

$$J_{\pi_o} = \mathbb{E}_{\pi_o}\left[r_t^{in} + \alpha H_o\right], where \quad r_t^{in} = \log q_\psi(o_t|s_{t+1}) - \log p(o_t) \tag{5}$$

where H_o denotes the entropy of the intra-option policy. $q_\psi(o_t|s_{t+1})$ denotes the variational inference of $p(o_t|s_{t+1})$ in Eq. (1). It is the negative value of the discriminator loss.

3.2. Learning adaptive scheduling

The main function of adaptive scheduling is to balance exploration and exploitation during the frequent scheduling of continuous options. Contrary to discrete options, the number of continuous options is infinite, and there is a small difference between trajectories guided by similar continuous options. Frequent scheduling can provide an agent with more opportunities to eliminate redundant behaviors and acquire near-optimal policies. Moreover, the option policy is unlikely to generate the same continuous option twice. After being interrupted, they cannot be selected again. Thus, scheduling plays a critical role in the HRL algorithm based on continuous options.

Specifically, the adaptive scheduling method builds on a multi-step static scheduling process and develops a new switching judgment mechanism for dynamic scheduling. Its design basis consists of two components. First, the multi-step static scheduling of $k > 1$ provides a fundamental exploration capability, enabling the agent to leave its local area. In the case of similar regions that are coherent, such as empty areas without obstacles, the agent's behavior can be made more directed. Second, the continuous option policy involves a certain degree of randomness. In the presence of noise perturbations, it is impossible to guarantee that the estimated option o^+ will be superior to the previous option o^-. If the switching judgment mechanism is still based on the advantage function $A(s,o') = Q(s,o') - V(s)$ as conventional methods (Harb et al., 2018), it will generate a suboptimal continuous option, thereby diverting actions away from the more optimal solution. To guarantee that a superior option is always generated, our interruption probability is calculated from the relative advantage between the previous option o^- and the estimated option o^+:

$$\beta(o^+|s,o^+,o^-) = \frac{\exp Q^{\pi_\mathcal{O}}(s,o^+)}{\exp Q^{\pi_\mathcal{O}}(s,o^-) + \exp Q^{\pi_\mathcal{O}}(s,o^+)} \tag{6}$$

where $\beta(o^+|s,o^+,o^-)$ denotes the probability of the estimated option o^+ being selected, which is abbreviated as $\beta(o^+)$. $Q^{\pi_\mathcal{O}}(s,o^+)$ and $Q^{\pi_\mathcal{O}}(s,o^-)$ denote the estimated option value and the previous option value. The softmax form provides randomness. Here, the extended-state transition function of HAS is:

$$P(\cdot|s,o^-) = (1-\beta(o^+))P(o^-|s,o^-) + \beta(o^+)P(o^+|s,o^-) \tag{7}$$

Correspondingly, the iterative process of the state value function $V^{\pi_\mathcal{O}}(s_t)$ will be influenced by the termination function $\beta(o^+)$:

$$\begin{aligned} V^{\pi_\mathcal{O}}(s_t) &= \mathbb{E}_{o_t\sim\pi_\mathcal{O}(\cdot|s_t)}\left[Q^{\pi_\mathcal{O}}(s_t,o_t)\right] \\ &= \mathbb{E}_{o_t^+\sim\pi_\mathcal{O}(\cdot|s_t)}\left[(1-\beta(o_t^+))Q^{\pi_\mathcal{O}}(s_t,o_t^-) + \beta(o_t^+)Q^{\pi_\mathcal{O}}(s_t,o_t^+)\right] \end{aligned} \tag{8}$$

The interruption judgment mechanism in Eq. (8) reflects the adaptive balance between exploration and exploitation. As the interruption probability is determined by the relative values of the options, the agent's behaviors will exhibit different emphases during the training process. During the early training phase (or random walk phase), the option value network is initialized at random. It causes continuous options to switch with equal probability. The agent prioritizes exploration and tends to leave its local area. This procedure also provides the low-level controller with a wide range of experience, enabling the intra-option policy to discover a stable set of trajectories. When the agent is consistently rewarded, the option values are revised and the interruption probability gradually decreases. It prioritizes exploitation and obtains a superior estimated option with a high probability. This process can also reduce redundant behaviors.

Algorithm 1 Hierarchical Reinforcement Learning with Adaptive Scheduling

Initialize: Random parameters for option policy φ, intra-option policy ϕ, and discriminator ψ. Empty high-level replay buffer $D_\mathcal{O}$ and low-level replay buffer D_o.
while *max experiment step not met* **do**
——— **Experience gathering module** ———
  Sample an estimated option $o^+ \sim \pi_\varphi(\cdot|s)$, $o \leftarrow o^+$ \\ follow the option policy at the first time of episodes.
  for *episode rollout step t* **do**
    for *intra-option step k* **do**
      Sample an action $a \sim \pi_\phi(\cdot|s,o)$
      Get next state s' and reward r^{env}
      $s \leftarrow s'$, $D_o \leftarrow (o,s')$ \\ store the low-level experience
    end for
    $R \leftarrow \sum_{l=t}^{t+k} r^{env}(o,s_l) + R$ \\ cumulative environmental rewards over a non-interrupted period
    $o^- \leftarrow o$, $t \leftarrow t+k$
    Sample an estimated option $o^+ \sim \pi_\varphi(\cdot|s)$
    Generate an activated option $o \sim \beta_\eta$ according to Equation (6) and Equation (10)
    if $o \neq o^-$: $\tilde{s}' \leftarrow s'$, $D_\mathcal{O} \leftarrow (s,o,\tilde{s}'_k,R)$, $R = 0$ \\ store the high-level experience, and then reset the accumulated environmental rewards
  end for
——— **High-level training** ———
  for *option policy training batches* **do**
    Sample batches $(s,o,\tilde{s}',R) \in D_\mathcal{O}$ uniformly \\ $\tilde{s}'$ denotes the state after a non-interrupted period
    Update the option policy network by maximizing Equation (3)
  end for
——— **Low-level training** ———
  for *intra-option policy training batches* **do**
    Sample batches $(o,s') \in D_o$ uniformly \\ s' denotes the state after one time step
    Calculate the negative predicted error of the discriminator network $q_\psi(o|s')$ and get intrinsic rewards r^{in} \\ compute intrinsic rewards in real-time via the discriminator network
    Update the intra-option policy network and the discriminator by maximizing Equation (5)
  end for
end while

3.3. Expected t-step distance

With adaptive scheduling, an agent can explore a larger degree of space while maintaining fine-grained behavioral control. Especially during the initial training and random walk phases, it can speed up the learning process, enabling the agent to reach target areas quickly. To demonstrate that adaptive scheduling provides superior exploration potential over other scheduling methods, we first make the following definition. Our objective is to solve sparse reward problems, such as robot control, and analyze these problems with optimal intra-option policies and no environmental rewards obtained. At this moment, the actions guided by the set of continuous options uniformly partition the state spaces surrounding the agent (see Appendix A for the proof) (Eysenbach et al., 2018). We further assume that the agent moves a unit distance d independently and identically in any direction.

We apply the expected t-step distance $\delta_{Sc(k)\to t}(s)$ (Schoknecht and Riedmiller, 2003) to measure how far an agent moves on average. It is

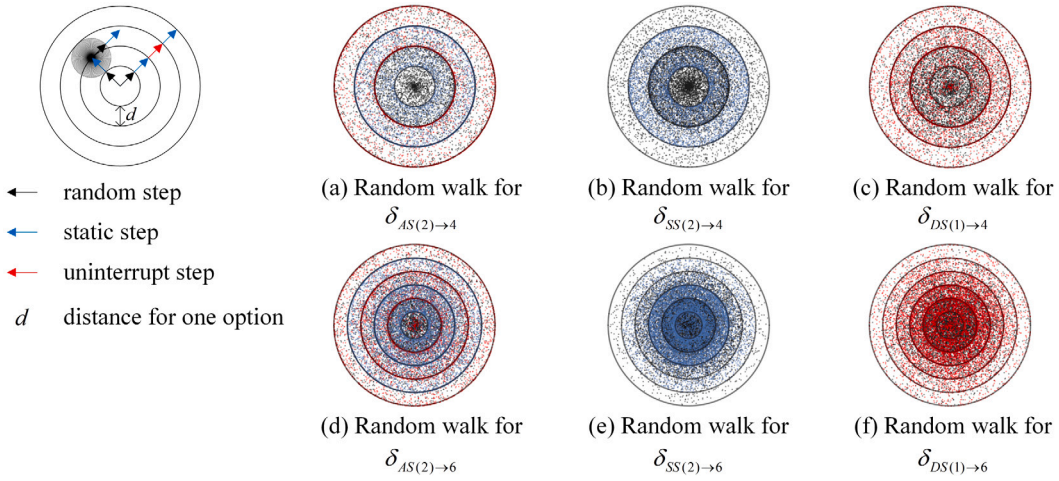


Fig. 4. The arrival states of different scheduling methods in time steps $t = 4$ (a–c) and $t = 6$ (d–f). Black dots indicate the states for random walks, blue dots indicate the states for static walks, and red dots indicate the states for non-interruption scheduling. Each task runs 10,000 episodes.

Table 1
The expected t-step distance. Distance ratios are calculated relative to the adaptive scheduling method.

	$t = 4$	Ratio	$t = 6$	Ratio
Adaptive scheduling $AS(2)$	**2.77**	1.00	**3.45**	1.00
Static scheduling $SS(2)$	2.52	0.91	2.88	0.83
Dynamic scheduling $DS(1)$	2.49	0.90	2.98	0.86

defined as the expected distance that the agent will move after t time steps from the state s using the scheduling method $Sc(k)$:

$$\delta_{Sc(k)\to t}(s) = \sum_{\bar{s}\in S} P_{Sc(k)\to t}(s,\bar{s}) D(s,\bar{s}) \tag{9}$$

where $D(s,\bar{s})$ is the arrival distance, which denotes the minimal number of options that are necessary to reach $\bar{s}$ from s. Distances less than one unit are calculated as one unit. They are weighted by the arrival probability $P_{Sc(k)\to t}(s,\bar{s})$, which denotes the probability that the agent reaches $\bar{s}$ from s in t time steps using $Sc(k)$.

In the following, we will compare adaptive scheduling $AS(2)$, static scheduling $SS(2)$, and dynamic scheduling $DS(1)$ according to the expected t-step distance. Meanwhile, we will analyze the effect of the intra-option step k on exploration and exploitation.

Impact of scheduling on exploration. Fig. 4 illustrates the arrival states of different scheduling methods in time steps $t = 4$ and $t = 6$ under the random walk. Their expected t-step distances are listed in Table 1. Note that the arrival results at time step $t = 1$ are not recorded because they are identical across all methods.

As can be seen in Fig. 4, the arrival states of dynamic and static scheduling methods are mainly distributed within circles, while those of the adaptive scheduling method are mainly distributed on circles. When the time step is increased to 6, the arrival states of dynamic and static scheduling methods remain concentrated in areas $D \le 4d$, while those of the adaptive scheduling method can be widely distributed in areas $D > 4d$. These properties are also reflected in Table 1. The adaptive scheduling method has the greatest expected t-step distance under the same intra-option step k condition. As the time step increases, its superiority increases as well.

It can be seen that the adaptive scheduling method enables the agent to break through its local area, resulting in a high degree of exploration. The property can be applied to three-dimensional spaces as well.

Impact of intra-option step on behavior. The intra-option step k determines the basic behavior mode. The larger the value of k, the larger the movement distance, but the less flexible the agent becomes. In the real world, the actions of humans and intelligent systems are not completed instantly. Pushing a box or picking up an object, for example, requires multiple primitive actions to achieve desired outcomes. This characteristic is reflected in the HAS idea that stable scheduling is essential for solving complex problems. This is also the basis for our proposal to execute the switching judgment based on multi-step static scheduling.

The intra-option step k has a significant effect on scheduling. A reasonable increase in k to improve exploration is more in line with HRL requirements for solving sparse reward problems. Nevertheless, k should not be too large, as this may hinder the agent to navigate tight spaces or reach target areas. For example, the agent may sway to the left and right outside the target areas, but cannot reach them. This results in a pessimistic local optimal solution. Using adaptive scheduling, it is possible to achieve a balance between exploration and exploitation with a small value of k, without manually adjusting k as the task progresses.

3.4. Interruption incentive

As discussed above, adaptive scheduling emphasizes exploration during the early training phase. It should be noted, however, that when certain options are rewarded in advance, their value rises first. They may have a greater chance of being selected and maintained than others. It is possible for the agent to be trapped in an excessive exploration dilemma, where the agent uses one or very few options to complete episodes. There is a restriction on the agent's behavior. This is also a facet of the hierarchical degradation problem, which results in a slower convergence rate.

To alleviate this problem, we propose an interruption incentive η based on the annealing mechanism for Eq. (6). During the early training phase, it appropriately increases the interruption probability of the option. The purpose of annealing is to prevent the destruction of the adaptive balance. The termination function $\beta_\eta(o^+)$ with the interruption incentive is as follows:

$$\beta_\eta(o^+) = min\left(\beta(o^+) - \mathbb{I}_{t<T*\ell}\left[(1-\frac{t}{T*\ell})\eta\right], 0\right) \tag{10}$$

where η is a smaller interruption incentive value. t denotes the current time step. $T * \ell$ denotes the duration of the interruption incentive. In a heuristic way, ℓ is set to 2/5. Under the indicator function $\mathbb{I}$, the interruption incentive is restricted to the early training phase. Grid

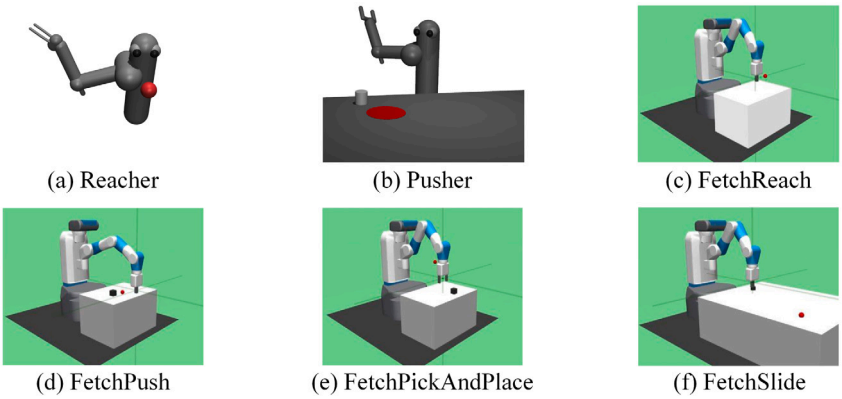


Fig. 5. Experimental environments. (a–b) are two tasks on Mujoco. (c–f) are four tasks on Robotics. Red spheres indicate targets, and black squares indicate movable objects.

Table 2
Basic characteristics of Tracks and Robotics. The max experiment step represents the total time step in the training process.

Environment	Task	State space dimension	Action space dimension	Max episode step	Max experiment step	Exclusive property
Mujoco	Reacher	17	7	100	3e6	–
	Pusher	20				Seq-decision
Robotics	FetchReach	10	4	50	3e6	–
	FetchPush	25				Seq-decision
	FetchPickAndPlace	25				Seq-decision
	FetchSlide	25				Seq-decision

search with ℓ on different tasks may produce superior results. Now, the non-interruption probability is $1 - \beta_\eta(o^+)$.

Unlike HAAR (Li et al., 2019b), the interruption incentive does not explicitly change the option length. Scheduling still depends mainly on the option value. Although the incentive method is very simple, its positive effects are readily apparent. In the experimental section, we will demonstrate this with ablation.

Based on the information provided above, the pseudocode of HAS is in Algorithm 1, which consists of three modules: experience gathering, high-level training, and low-level training.

4. Experimental results

We develop an exhaustive experimental analysis scheme, including a comparison, two types of ablation, and three effects experiments. First, it demonstrates the superiority of HAS for solving robot control with sparse rewards in continuous spaces. Second, it verifies the effectiveness of our proposed techniques through ablation studies, as well as the robustness. Third, it identifies the operation mechanism of the adaptive scheduling method and analyzes the root of HAS advantages from multiple perspectives. As a whole, this scheme comprehensively supports the core idea that the continuous option leads to a qualitative shift in HRL, while adaptive scheduling can fully utilize continuous scheduling. It is reasonable to consider exploration and exploitation in a balanced manner during the frequent scheduling of continuous options.

4.1. Environments

Our experiment applies Mujoco (multi-joint dynamics and contact) (Chua et al., 2018) environment and Robotics (Brockman et al., 2016) environment. They are general-purpose physics engines that simulate the interaction of cross-linked structures with their environments in fields such as robotics, biomechanics, and machine learning. We choose representative robot control problems associated with sparse reward tasks in continuous spaces, as depicted in Fig. 5. Mujoco includes Reacher and Pusher, while Robotics includes FetchReach, FetchPush, FetchPickAndPlace, and FetchSlide. They consist of three components: the agent, the movable object, and the target. Policies control the rotation angle of each joint of the robotic arm to complete tasks. The state space consists of the coordinates, angles, and velocities of joints, the coordinates of the object (if it exists), and the coordinates of the target. The action space consists of the rotation angles of joints.

The agent is penalized -0.1 for each step, except when it touches the target or pushes/grabs/hits the object to the target area. The task is deemed successful if it is not punished at the end of the maximum episode step. Otherwise, it is deemed unsuccessful. The episode length of Mujoco is 100, while that of Robotics is 50. Pusher, FetchPush, FetchPickAndPlace, and FetchSlide with a sequential decision process, in which the agent must learn how to the control arm to interact with a target, making learning more challenging. The basic characteristics of Mujoco and Robotics are summarized in Table 2.

4.2. Settings

The following guidelines are followed when comparing performance. Each task provides 10 random seeds for all algorithms, and each seed evaluates every 50 training iterations. Each evaluation consists of 20 test episodes, whose results are recorded as the criteria standard. All algorithms use consistent parameters as soon as possible, and they are trained from scratch. We use both success rate and terminal success rate metrics to describe algorithm performance. The success rate is calculated by averaging the success rates at different evaluations from all seeds. The terminal success rate refers to the average success rate of the last 10% time steps in the evaluation condition, and the terminal variance is calculated from this data.

4.3. Baselines

A total of six representative HRL algorithms that are highly correlated with HAS, as well as two predominant underlying algorithms are applied as baselines. Both the high-level and low-level controllers of HAS and most of baselines are based on SAC (Haarnoja et al.,

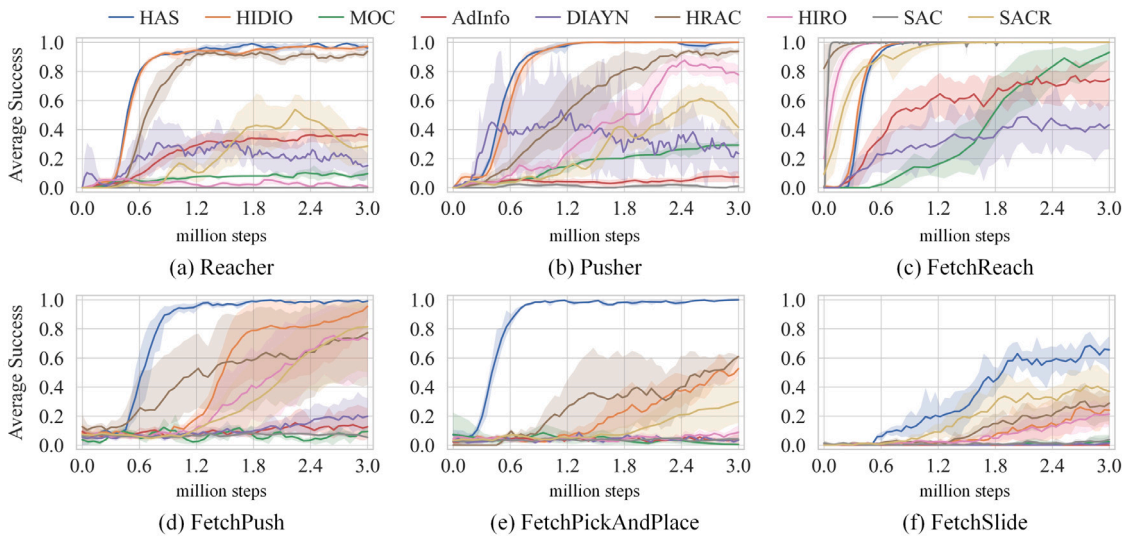


Fig. 6. Comparison of the average success rates of HAS against baselines. The shaded areas indicate standard deviations.

Table 3
Comparison of the terminal success rate (%) of HAS and baselines. The best one is highlighted in bold. The terminal variance of data with a success rate greater than 30% is annotated.

Algorithm	Reacher	Pusher	FetchReach	FetchPush	FetchPickAndPlace	FetchSlide	Average
HAS	**0.97** ± 0.00	0.99 ± 0.00	**1.00** ± 0.00	**0.99** ± 0.00	**1.00** ± 0.00	**0.64** ± 0.02	**0.93**
HIDIO	0.96 ± 0.00	**1.00** ± 0.00	**1.00** ± 0.00	0.90 ± 0.04	0.48 ± 0.01	0.22	0.76
MOC	0.09	0.29	0.99 ± 0.00	0.08	0.05	0.05	0.26
AdInfo	0.36 ± 0.00	0.07	0.74 ± 0.04	0.13	0.04	0.00	0.22
DIAYN	0.19	0.28	0.42 ± 0.05	0.20	0.04	0.01	0.19
HRAC	0.92 ± 0.00	0.92 ± 0.00	**1.00** ± 0.00	0.74 ± 0.16	0.54 ± 0.01	0.28	0.73
HRIO	0.02	0.81 ± 0.01	**1.00** ± 0.00	0.73 ± 0.13	0.08	0.20	0.47
SAC	0.00	0.01	**1.00** ± 0.00	0.07	0.04	0.02	0.19
SACR	0.32 ± 0.04	0.52 ±	**1.00** ± 0.00	0.83 ± 0.12	0.30 ± 0.05	0.37 ± 0.02	0.56

2018). The core parameters of HAS are shown in Appendix B. These baselines meet the requirements of up-to-date technology, high quality, and identical fields, ensuring the credibility of comparisons. Their characteristics are listed below:

- **HIDIO** (Zhang et al., 2021). It is the most recent HRL algorithm to employ continuous options. Multiple discriminator instances are provided to generate distinct intrinsic rewards. However, the continuous option is only treated as a meaningless code, which prevents the algorithm from discovering the diversity characteristic of continuous options and further developing an appropriate scheduling method. Its limitations serve as an inspiration for us.
- **MOC** (Klissarov and Precup, 2021). This is an up-to-date HRL algorithm based on discrete options. It proposes a method for updating all relevant options for a visited state, ensuring that all options are updated and employed as much as possible.
- **AdInfo** (Osa et al., 2019). It is the typical HRL algorithm based on discrete options. It clusters options by establishing a correlation between options and advantage functions.
- **DIAYN** (Eysenbach et al., 2018). It is the most classic HRL algorithm based on discrete options. It creates the inferred relationship between options, states, and actions.
- **HRAC** (Zhang et al., 2020). This is an up-to-date HRL algorithm based on goals. The goal space is constrained to the region adjacent to the current state. It is capable of solving sparse reward problems in continuous spaces.
- **HIRO** (Nachum et al., 2018). This is one of the most classic HRL algorithms based on goals. The direction residual between the desired arrival state and the current state is used as a goal. The L2 distance between the current state and the goal is used as an intrinsic reward.
- **SAC** (Haarnoja et al., 2018). As the predominant underlying algorithm of HRL, it is an off-policy method based on the maximum entropy RL framework. It encourages the agent to act randomly.
- **SACR**. This is a multi-step extension of SAC. Similar to static scheduling, each action is executed several times. We utilize this method to demonstrate the superiority of the hierarchical framework over the flat RL method.

4.4. Comparison

Fig. 6 illustrates the average success rates of HAS against baselines. Table 3 shows their terminal success rates. It can be seen that the majority of algorithms can complete FetchReach, but only HAS, HIDIO, and HRAC can complete Reacher, Pusher, FetchPush, and FetchPickAndPlace with a success rate close to 100%. Among them, HAS converges the fastest and has a very small shadow area, which reflects its high stability. HAS is also capable of solving FetchSlide with a success rate of more than 60%, while other algorithms are incapable of doing so. Observing SAC and its non-hierarchical extension algorithm SACR, we can see that the hierarchical framework truly plays an important role. All these results indicate that HAS is superior to baselines, especially for challenging tasks. From the perspective of task difficulty, it is generally possible for most algorithms to solve Reacher and FetchReach because they do not involve sequential decisions. Pusher generates objects close to the target, so it is also not too challenging. As the difficulty of FetchPush, FetchPickAndPlace, and FetchSlide increases, the agent is required to perform more complex operations. For instance, if FetchSlide's agent fails to hit the object in the correct direction, or does not have enough strength to hit the object, the task will fail. This is also one reason why baseline performance lags far behind HAS on the last three tasks.

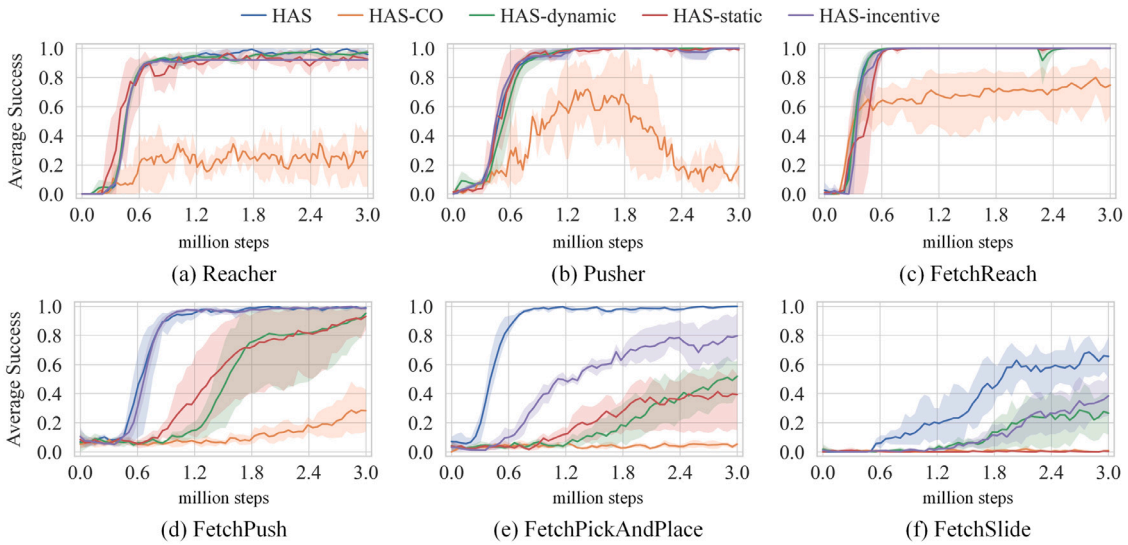


Fig. 7. Functional ablation for HAS. The method following "-" indicates the component to be removed.

The success rates of MOC, AdInfo, and DIAYN on FetchPush, FetchPickAndPlace, and FetchSlide are almost always below 5%. The poor performance is due to the limitations of discrete options, which cannot provide a variety of representation and scheduling possibilities. When faced with sparse reward tasks in continuous spaces, the agent cannot fully explore the space to reach the possible locations of the target, nor can it generate effective solutions to incrementally improve the success rate. We consider this to be the performance bottleneck.

HRAC and HIRO choose arbitrary visited space coordinates as their subgoals. While HRAC and HIRO outperform the HRL algorithms based on discrete options, they clearly lag behind HAS. In particular, they are significantly less stable on tasks involving sequential decisions, including Pusher, FetchPush, FetchPickAndPlace, and FetchSlide. It is possible that constructing a set of subgoals from an initial position to a target position generates a significant amount of randomness.

In summary, continuous options unlock the potential of temporal abstraction with rich representation. It is the foundation upon which HAS and HIDIO are able to successfully solve complex problems. While adaptive scheduling takes full advantage of the diversity of continuous options resulting in more efficient and stable performance.

4.5. Ablation study

We construct functional and parametric ablation experiments. The purpose of this study is to determine whether the components of HAS play a positive role, and to examine the sensitivity of HAS to the fundamental parameters. Moreover, we conduct ablation experiments on the data samples of the high-level and low-level controllers to determine the computation complexity of HAS.

4.5.1. Functional ablation

The continuous option method, the dynamic scheduling method, the static scheduling method, and the interruption incentive are removed from HAS, respectively. Their experimental results are illustrated in Fig. 7. When the continuous option method is removed, the switching judgment mechanism reverts to the loss update method based on the advantage function (Harb et al., 2018). When the dynamic scheduling method is removed, the interruption incentive is also removed.

It can be seen that the performance of the four ablation algorithms decreases on most tasks. Among them, HAS-CO suffers the greatest performance loss. Even though we make sure that its structure, data processing techniques, and parameters are as similar as possible to HAS, it is almost incapable of accomplishing tasks other than FetchReach, which is similar to the results of AdInfo and DIAYN. The performance and stability of both HAS-dynamic and HAS-static are impaired. It is shown that dynamic scheduling and static scheduling are both important and compatible. In addition, the convergence rate of HAS-incentive becomes slower. On FetchPickAndPlace and FetchSlide, its performance drops by almost half. Although it only intervenes in the early training phases, its effects can be observed throughout the entire training process. The results demonstrate that the interruption incentive helps the agent to escape the dilemma of excessive exploration, resulting in a significant increase in performance.

It can be concluded that continuous options are responsible for the qualitative shift in HRL. It serves as the foundation for overcoming the performance bottleneck caused by discrete options. Moreover, our analysis of the balance between exploration and exploitation is reasonable. Adaptive scheduling and the interruption incentive should be considered as a whole. Each component is critical to its success.

4.5.2. Parametric ablation

Option dimension n and intra-option step k are the fundamental parameters of adaptive scheduling. We ablate each parameter separately. The ablation results of HIDIO are used for comparison. Their terminal average success rates under different option dimensions and intra-option steps are listed in Tables 4 and 5.

As shown in Table 4, HAS + dim = 6 has the highest terminal success rate with 93%, while the optimal HIDIO is HIDIO + dim = 6 with 78%. Their optimal option dimensions are all six. This reflects the fact that the option is the carrier of knowledge representation. An appropriate option dimension n is conducive to representing rich knowledge. In addition, the success rates of HAS are significantly higher than those of HIDIO under the same conditions. In HAS, the lowest terminal success rate is still greater than 60%, while in HIDIO, it is less than 30%. Among them, HIDIO + dim = 2 cannot complete the majority of tasks on Robotics, and its success rate is close to 0%. Compared with HIDIO, HAS is much less sensitive to the option dimension. This indicates that HAS achieves stable performance with varying option dimensions through adaptive scheduling.

As shown in Table 5, HAS + step = 3 has the highest terminal success rate with 93%, while the optimal HIDIO is HIDIO + step = 5 with 77%. In addition to the optimal data, the results about "+ step = 2" are of great concern to us. Compared with HIDIO + step = 2, HAS + step = 2 has a 31% higher success rate and can excellently complete the majority of tasks. This is consistent with our evaluation regarding the expected t-step distance: Adaptive scheduling can facilitate a higher

Table 4
Comparison of the terminal success rate (%) of HAS and HIDIO under different option dimensions. "+ dim = n" denotes the algorithm using n-dimensional options. The best one is highlighted in bold.

Algorithm	Reacher	Pusher	FetchReach	FetchPush	FetchPickAndPlace	FetchSlide	Average
HAS + dim = 2	0.42	0.35	0.95	0.95	0.96	0.12	0.63
HAS + dim = 4	0.97	0.99	**1.00**	**0.99**	**1.00**	0.18	0.85
HAS + dim = 6	**0.98**	0.98	**1.00**	0.98	0.98	**0.64**	**0.93**
HIDIO + dim = 2	0.36	0.25	0.92	0.08	0.03	0.02	0.28
HIDIO + dim = 4	0.96	**1.00**	**1.00**	0.98	0.45	0.21	0.77
HIDIO + dim = 6	0.96	0.95	**1.00**	0.90	0.54	0.30	0.78

Table 5
Comparison of the terminal success rate (%) of HAS and HIDIO under different intra-option steps. "+ step = k" denotes the algorithm using intra-option step k.

Algorithm	Reacher	Pusher	FetchReach	FetchPush	FetchPickAndPlace	FetchSlide	Average
HAS + step = 2	0.95	0.99	**1.00**	0.98	0.78	0.42	0.85
HAS + step = 3	0.97	0.99	**1.00**	**0.99**	**1.00**	**0.64**	**0.93**
HAS + step = 5	**0.98**	0.99	**1.00**	0.97	0.71	0.21	0.81
HIDIO + step = 2	0.86	0.82	**1.00**	0.24	0.24	0.08	0.54
HIDIO + step = 3	0.96	**1.00**	**1.00**	0.90	0.48	0.21	0.76
HIDIO + step = 5	0.96	0.97	**1.00**	0.98	0.54	0.19	0.77

Table 6
Comparison of the terminal success rate (%) of HAS under different sample data sizes. "+ Hb = n" denotes the algorithm using high-level sample data sizes n, and "+ Lb = n" denotes the algorithm using low-level sample data sizes n.

Algorithm	Reacher	Pusher	FetchReach	FetchPush	FetchPickAndPlace	FetchSlide	Average
HAS + Hb = 1024	0.95	**1.00**	**1.00**	0.98	0.98	0.41	0.89
HAS + Hb = 2048	**0.97**	0.97	**1.00**	**0.99**	**1.00**	**0.64**	**0.93**
HAS + Hb = 4096	**0.97**	0.99	**1.00**	**0.99**	0.98	0.42	0.89
HAS + Lb = 1024	**0.97**	**1.00**	**1.00**	0.98	0.94	0.45	0.89
HAS + Lb = 2048	0.91	0.99	**1.00**	**0.99**	**1.00**	**0.64**	0.92
HAS + Lb = 4096	**0.97**	0.99	**1.00**	0.98	0.98	0.46	0.90

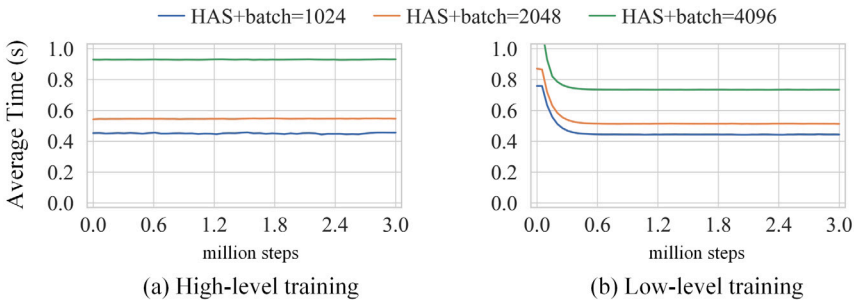


Fig. 8. Training time for HAS on all tasks under different sample data sizes. "+ batch = n" denotes the algorithm using sample data sizes n. The time unit is seconds.

level of action capacity. However, we should avoid too large an intra-option step. Particularly for HAS, its performance decreases when the intra-option step is increased to five. If the action sequences are too long, the agent may miss target areas or push objects out of these areas. As long as the agent cannot consistently obtain rewards, it will be difficult to reduce the interruption probability, resulting in a pessimistic local optimal solution.

In summary, the HAS is sufficiently robust to changes in fundamental parameters within a certain range. It is less sensitive to the option dimension n and well suited to smaller intra-option step k. These experimental results show that HAS benefits from adaptive scheduling. Of course, it is natural to select different parameters for different environments or tasks. It is just that HAS is able to minimize the workload associated with parameter adjustment.

4.5.3. Computation complexity

Our high-level and low-level controllers have separate experience buffers and are trained independently. We perform ablation experiments on data samples to validate the effect of data on computational complexity. Table 6 shows their terminal success rates. It can be seen that different sample data sizes do not have a serious impact on their performance. All of their terminal success rates remain above 85%. The results also demonstrate the high robustness of HAS to the number of data samples.

Fig. 8 illustrates the average training time on all tasks under different sampled data sizes. The configuration is provided in Appendix C. It can be seen that the training time increases significantly as the sampled data size grows. In combination with the results in Table 3, we believe that 1024 or 2048 provides sufficient results and reduces training time.

4.6. Effects of adaptive scheduling

Three functional experiments were developed to on Robotics further investigate the source of adaptive scheduling advantages, including the adaptive mechanism, the effect on representation, and the effect on option policies.

4.6.1. Adaptive mechanism

Fig. 9 illustrates the option length of HAS and HAS-incentive with standard parameters. This is calculated by averaging the time steps in which each option is continuously executed at different evaluations from all seeds.

It can be seen that the option length of HAS is greater than the intra-option step ($k = 3$) in all tasks, indicating that non-interruption

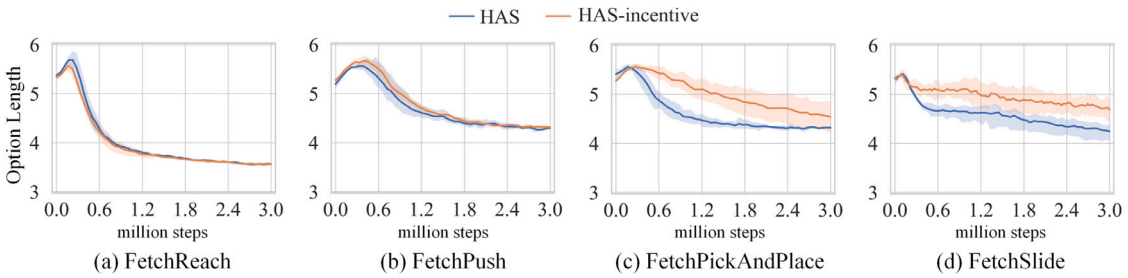


Fig. 9. Variation in the option length of HAS and HAS-incentive.

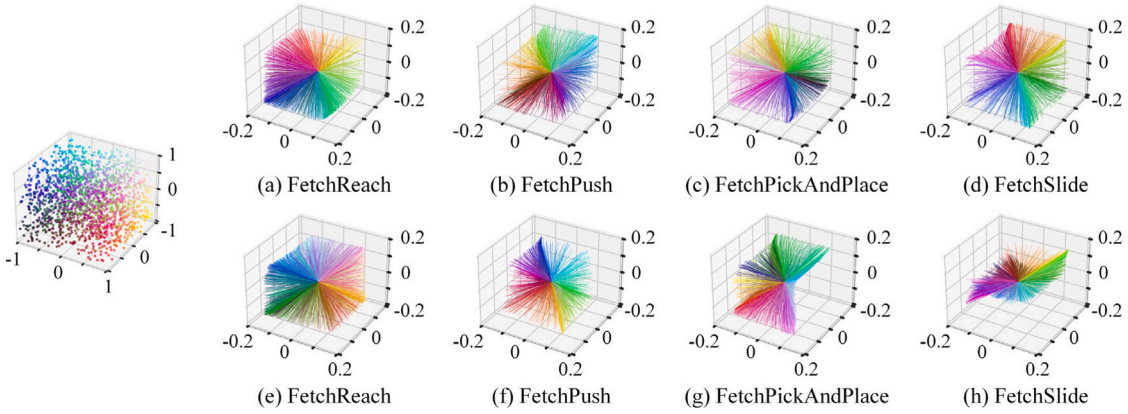


Fig. 10. Trajectories of $\{O\}^3$ HAS (a–d) and $\{O\}^3$ HIDIO (e–h). For each task, 10,000 continuous options are randomly sampled, with an intra-option step $k = 6$. Line colors correspond to the sampling distribution.

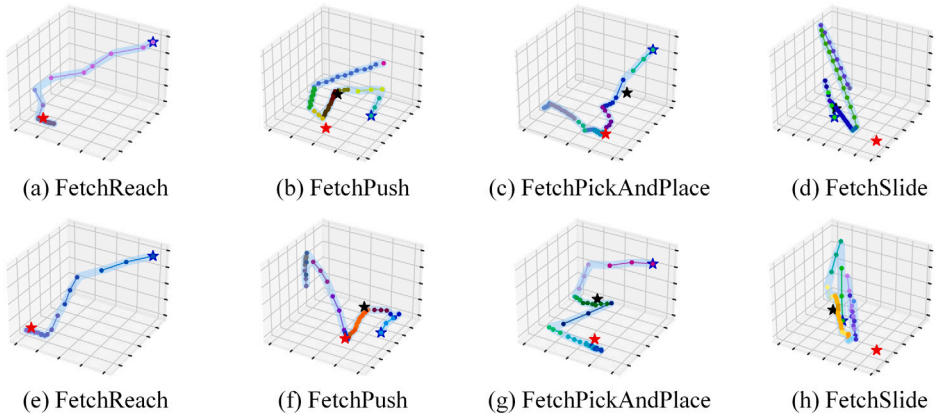


Fig. 11. Task path of HAS (a–d) and HIDIO (e–l). Dots indicate the states visited by the agent, and their colors correspond to those shown in Fig. 10. The light blue line indicates the route of the movement, and line segments indicate paths that are guided by the same option. The blue pentagram indicates the initial state, the red pentagram indicates the target state, and the black pentagram indicates the box that can be moved.

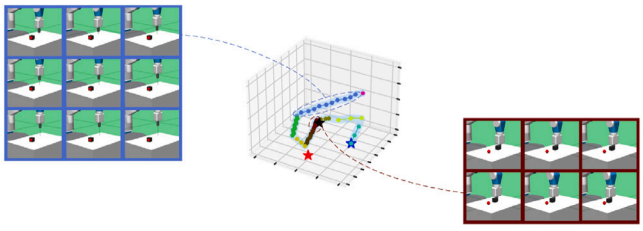


Fig. 12. Render case of the HAS's task path on FetchPush. The pictures are arranged from left to right, from top to bottom.

cases are actively used. Due to the random initialization of the value network, the option length is longer during the early training phase. At this moment, the adaptive scheduling method focuses primarily on exploration. Additionally, there is a slight increase in the option length. This is due to Robotics' random target generation around the agent. Value updates will be applied to a small percentage of options that are prioritized for rewards. Thus, the option policy will intend to select and maintain them, resulting in the dilemma of excessive exploration. The interruption incentive is exactly proposed to alleviate this problem. It can be observed that the decrease rate of the option length of HAS is apparently greater than HAS-incentive. This is also the primary reason why its performance is substantially superior to HAS-incentive.

During the mid-late phase of training, as the agent is exposed to rewards in a broader range of areas, the option value is widely updated. The agent can select more superior options to reach any target area. The

Table 7
Final task step of HAS and HIDIO. As Robotics is configured to take 50 time steps, we calculate the first time step required to complete tasks.

Algorithm	FetchReach	FetchPush	FetchpickAndPlace	FetchSlide	Average
HAS	**1.65**	**15.12**	**11.75**	**38.70**	**16.80**
HIDIO	1.91	18.87	34.26	44.22	24.81

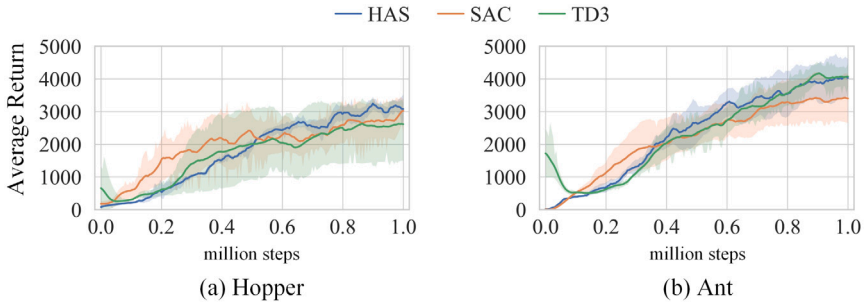


Fig. 13. Comparison of the average success rates of HAS against SAC and TD3 on Hopper and Ant.

option length also continuously decreases. At this moment, adaptive scheduling focuses primarily on exploitation.

The law of option length reflects the adaptability of HAS. Comparing Fig. 9 with Fig. 6, we can see that the success rate is negatively correlated with the option length. When the variation in option length tends to be flat, performance is also close to optimal. In addition, On FetchPickAndPlace and FetchSlide, although the option length of HAS-incentive continues to decrease, it remains greater than HAS at the final moment. Excessive exploration makes it difficult to learn a near-optimal strategy. Improving the exploitation level requires a considerable amount of effort. While the interruption incentive greatly alleviates this problem. Noted, even though adaptive scheduling may introduce this new problem, the performance of HAS-incentive is still significantly superior to that of HIDIO.

4.6.2. Effect on representation

In HAS, the high-level controller provides rollout and training information to the low-level controller. Therefore, the scheduling process influences the diversity degree of continuous options to some extent. We select several HAS and HIDIO models with superior performance and compare their representation capabilities by drawing the trajectories of a set of options, as depicted in Fig. 10. It can be seen that HAS and HIDIO represent similarly on FetchReach and FetchPush. On FetchPickAndPlace and FetchSlide, however, HIDIO seems to lose the ability to act in certain directions, which may prevent it from reaching some tricky angles. We believe this is due to the limited exploration capabilities of HIDIO. The high-level controller cannot provide enough rich information to the low-level controller through extensive exploration.

4.6.3. Effect on option policy

The most significant effect of adaptive scheduling on HAS lies in the process of optimizing the option policy. Besides the performance improvement shown in Fig. 6, this effect can also be observed in the timing of the option switching, as well as the relationship between the gradually decaying option length and the final task step. Here, we list the final task steps of HAS and HIDIO in Table 7. Meanwhile, we select several HAS and HIDIO models with superior performance, and then illustrate their paths in Fig. 11. We manually position the target away from the robotic arm's starting position to intuitively illustrate their disparity.

As shown in Fig. 11, the agent prefers to maintain options in areas where the movement direction should be maintained, such as approaching and leaving the target area. When it is necessary to adjust the movement direction, such as adjusting the direction of pushing and hitting the box, the agent prefers to switch options. These characteristics suggest that adaptive scheduling places a different emphasis on exploration and exploitation in different areas. Compared with HIDIO, the path of HAS is more directional. Fig. 12 shows a render case of the HAS's task path on FetchPush. On the right, the arm is pushing the box to the target area. On the left, the arm is leaving the target area to ensure that the box cannot be moved. In conjunction with Table 7, it is evident that HAS has a shorter task step than HIDIO, with an average decrease of 68%. HAS can learn better policies over time.

In summary, the results of the three functional experiments demonstrate that adaptive scheduling plays an important role in the learning process of HAS. The attribute of adaptability is manifested in a variety of learning phases and areas in all tasks. It is possible to achieve rich representation and stable scheduling by the adaptive capacity. Consequently, HAS is more efficient and easier to learn near-optimal policies.

4.7. More comparison results

Besides sparse reward tasks, HAS also maintains the processing power of traditional RL for dense reward tasks. We select two dense reward tasks, Hopper and Ant from Mujoco and evaluate HAS, SAC, and TD3 (Fujimoto et al., 2018) based on the average return index, as shown in Fig. 13. The results demonstrate that HAS is able to solve dense reward tasks. However, the learning rate of HAS is slower than SAC and TD3 during the early training phase, as it requires more time to learn the low-level controller. This limitation, on Robotics, does not cause our focus because SAC is not able to complete these sparse reward tasks. In addition, compared with the experimental results of Robotics, the performance of HAS does not have a significant gap with SAC and TD3. This is because dense reward environments do not require much exploration. The agent should first consider its sensitivity to instantaneous rewards, rather than its ability to explore.

5. Conclusion

We propose the hierarchical reinforcement learning algorithm with adaptive scheduling (HAS) algorithm. This algorithm exploits the representation and scheduling potential of continuous options to solve

Table B.8
Parameters of HAS. "▲" denotes that the marked parameter is shared by the high-level and low-level controllers.

Description	Unit	Value	
		Mujoco	Robotics
Rollout step	*step*	25	50
Policy training batches per iteration ▲	*num*	100	40
Mini batch size ▲	*num*	4096	2048
Learning rate ▲		3e-5	1e-3
Option dim		4, 6(FetchSlide)	
Interruption incentive		0.05	
Intra-option step	*step*	3	
Initial collect step	*step*	10000	
Replay buffer length ▲	*num*	200000	
Number of units in high AC (actor–critic) hidden layers		(256, 256, 256)	
Number of units in low AC hidden layers		(128, 128, 128)	
Number of units in discriminator hidden layers		(64, 64)	
Hidden layers activation ▲		Relu	
Optimizer ▲		Adam	
High level reward discount factor		0.99	
Low level reward discount factor		1	
Coefficient for soft updating ▲		1e-3	
Policy target entropy min prob ▲		0.05	
Interval episodes per evaluation	*episode*	50	
Episodes per evaluation	*episode*	10	

Table C.9
The amount of computing and the type of resources.

Configuration	Value
Architecture	x86_64
CPU op-mode(s)	32-bit, 64-bit
CPU(s)	48
On-line CPU(s) list	0–47
Thread(s) per core	1
Core(s) per socket	24
CPU family	6
Model	85
Model name	Intel(R) Xeon(R) Gold 5220R CPU @ 2.20 GHz
Stepping	7
CPU MHz	999.963
CPU max MHz	4000
CPU min MHz	1000
3D controller	NVIDIA Corporation GP102GL [Tesla P40] (rev a1)

robot control problems with sparse rewards in continuous spaces. It emphasizes the balance between exploration and exploitation during frequent scheduling. Through a multi-step static scheduling process and a value judgment switching mechanism, the agent's behavior can be adaptively adjusted. The interruption incentive is introduced to the switching mechanism to improve adaptability and alleviate excessive exploration. In comparison experiments, we select recent baselines that are highly relevant to HAS. The results demonstrate that HAS has a significant advantage in terms of performance and convergence rate. Our ablation study verifies that All technical components contribute positively, and HAS is enough robust to fundamental parameters. In functional experiments, we record the option length of HAS at different phases of tasks, which intuitively reflects the operating mechanism of adaptive scheduling. This mechanism provides HAS with a more stable representation result and a superior option policy.

Noted, our agent's exploration ability depends almost entirely on the active exploration mechanism of temporal abstraction, as opposed to building a comprehensive understanding of the environment. That is, there is no reliable mechanism to guide the agent away from well-explored regions, which may slow down the convergence rate. Incorporating an intrinsic motivation measure into HAS may further improve the agent's exploration ability without compromising its temporal abstraction. They are compatible with each other. In the future, we will attempt to put this idea into practice.

CRediT authorship contribution statement

Zhigang Huang: Conceptualization, Methodology, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Project administration. **Quan Liu:** Supervision, Resources, Writing – review & editing, Funding acquisition. **Fei Zhu:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I have shared the link to my data/coda at the Attach File step

HAS code (Original data) (github).

Acknowledgments

This work has been supported by the National Natural Science Foundation of China (61772355), Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD).

Appendix A. Optimum for continuous options

DIAYN proved that a set of discrete options that evenly partition the discrete state space is the optimal solution for the low-level objective. This optimality problem can also be applied to continuous options in continuous state spaces. When continuous options evenly partition the continuous st ate space, we can always infer the continuous option from the continuous state, so $H(O|S) = 0$. Assuming a continuous distribution with constant mean and variance, the probability distribution of maximum entropy would be Gaussian. A standard Gaussian distribution can be constructed to achieve the maximum entropy: $H(O) = \log\sqrt{2\pi}$. Thus, a set of continuous options that evenly partition the continuous state space can maximize mutual information.

Appendix B. Parameters

The core parameters of HRL in this paper are mostly derived from SAC. We employ a grid search method to identify useful parameters, since the papers of SAC did not involve Robotics. As much as possible, we ensure that all algorithms have consistent parameters. Table B.8 describes the parameters of HAS.

Appendix C. Resource

Table C.9 lists the amount of computing and the type of resources.

References

Abramowitz, S., Nitschke, G., 2022. Towards run-time efficient hierarchical reinforcement learning. In: IEEE Congress on Evolutionary Computation. IEEE, pp. 1–8.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., Zaremba, W., 2017. Hindsight experience replay. In: Advances in Neural Information Processing Systems, Vol. 30. MIT Press, Cambridge, pp. 5048–5058.

Aubret, A., Matignon, L., Hassas, S., 2023. An information-theoretic perspective on intrinsic motivation in reinforcement learning. Entropy 25 (2), 327.

Bacon, P.L., Harb, J., Precup, D., 2017. The option-critic architecture. In: AAAI Conference on Artificial Intelligence. AAAI, Menlo Park, pp. 1726–1734. http://dx.doi.org/10.1609/aaai.v31i1.10916.

Bacon, P.-L., Precup, D., 2018. Constructing temporal abstractions autonomously in reinforcement learning. AI Mag. 39 (1), 39–50.

Badia, A.P., Sprechmann, P., Vitvitskyi, A., Guo, D., Piot, B., Kapturowski, S., Tieleman, O., Arjovsky, M., Pritzel, A., Bolt, A., 2020. Never give up: Learning directed exploration strategies. In: International Conference on Learning Representations.

Bagaria, A., Konidaris, G., 2020. Option discovery using deep skill chaining. In: International Conference on Learning Representations.

Baumli, K., Warde-Farley, D., Hansen, S., Mnih, V., 2021. Relative variational intrinsic control. In: AAAI Conference on Artificial Intelligence, Vol. 35. AAAI, CA, pp. 6732–6740. http://dx.doi.org/10.1609/aaai.v35i8.16832.

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., Munos, R., 2016. Unifying count-based exploration and intrinsic motivation. In: Advances in Neural Information Processing Systems, Vol. 29. pp. 1471–1479.

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

Brunner, G., Fritsche, M., Richter, O., Wattenhofer, R., 2018. Using state predictions for value regularization in curiosity driven deep reinforcement learning. In: International Conference on Tools with Artificial Intelligence. IEEE, pp. 25–29.

Burda, Y., Edwards, H., Storkey, A., Klimov, O., 2018. Exploration by random network distillation. In: International Conference on Learning Representations.

Cho, D., Kim, J., Kim, H.J., 2022. Unsupervised reinforcement learning for transferable manipulation skill discovery. IEEE Robot. Autom. Lett. 7 (3), 7455–7462.

Choi, J., Guo, Y., Moczulski, M., Oh, J., Wu, N., Norouzi, M., Lee, H., 2018. Contingency-aware exploration in reinforcement learning. In: International Conference on Learning Representations.

Chua, K., Calandra, R., McAllister, R., Levine, S., 2018. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In: Advances in Neural Information Processing Systems, Vol. 31.

Chua, K., Lei, Q., Lee, J., 2023. Provable hierarchy-based meta-reinforcement learning. In: International Conference on Artificial Intelligence and Statistics. PMLR, pp. 10918–10967.

Cobbe, K.W., Hilton, J., Klimov, O., Schulman, J., 2021. Phasic policy gradient. In: International Conference on Machine Learning. PMLR, pp. 2020–2027.

Dai, S., Xu, W., Hofmann, A., Williams, B., 2021. An empowerment-based solution to robotic manipulation tasks with sparse rewards. In: Robotics: Science and Systems. http://dx.doi.org/10.15607/rss.2021.xvii.001.

Ding, F., Zhu, F., 2022. HLifeRL: A hierarchical lifelong reinforcement learning framework. J. King Saud Univ.-Comput. Inf. Sci. 34 (7), 4312–4321.

Dukkipati, A., Banerjee, R., Ayyagari, R.S., Udaybhai, D.P., 2022. Learning skills to navigate without a master: A sequential multi-policy reinforcement learning algorithm. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, pp. 2483–2489.

Ermolov, A., Sebe, N., 2020. Latent world models for intrinsically motivated exploration. In: Advances in Neural Information Processing Systems, Vol. 33. pp. 5565–5575.

Eysenbach, B., Gupta, A., Ibarz, J., Levine, S., 2018. Diversity is all you need: Learning skills without a reward function. In: International Conference on Learning Representations.

Fakoor, R., Chaudhari, P., Soatto, S., Smola, A.J., 2019. Meta-Q-learning. In: International Conference on Learning Representations.

Florensa, C., Duan, Y., Abbeel, P., 2017. Stochastic neural networks for hierarchical reinforcement learning. In: International Conference on Learning Representations.

Frans, K., Ho, J., Chen, X., Abbeel, P., Schulman, J., 2017. Meta learning shared hierarchies. In: International Conference on Learning Representations.

Fujimoto, S., van Hoof, H., Meger, D., 2018. Addressing function approximation error in actor-critic methods. In: International Conference on Machine Learning, Vol. 80. ACM, New York, pp. 1582–1591.

Gregor, K., Rezende, D.J., Wierstra, D., 2016. Variational intrinsic control. arXiv preprint arXiv:1611.07507.

Haarnoja, T., Zhou, A., Abbeel, P., Levine, S., 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International Conference on Machine Learning. ACM, New York, pp. 1861–1870.

Harb, J., Bacon, P.L., Klissarov, M., Precup, D., 2018. When waiting is not an option: Learning options with a deliberation cost. In: AAAI Conference on Artificial Intelligence, Vol. 32. AAAI, Menlo Park, pp. 3165–3172. http://dx.doi.org/10.1609/aaai.v32i1.11831.

Harutyunyan, A., Vrancx, P., Bacon, P.-L., Precup, D., Nowe, A., 2017. Learning with options that terminate off-policy. In: AAAI Conference on Artificial Intelligence. AAAI, Menlo Park, pp. 3173–3182. http://dx.doi.org/10.1609/aaai.v32i1.11740.

Hasselt, H., 2010. Double Q-learning. In: Advances in Neural Information Processing Systems, Vol. 23. MIT Press, Cambridge, pp. 2613–2621.

Hou, Z., Zhang, K., Wan, Y., Li, D., Fu, C., Yu, H., 2020. Off-policy maximum entropy reinforcement learning: Soft actor-critic with advantage weighted mixture policy. arXiv preprint arXiv:2002.02829.

Igl, M., Gambardella, A., He, J., Nardelli, N., Siddharth, N., Böhmer, W., Whiteson, S., 2020. Multitask soft option learning. In: Conference on Uncertainty in Artificial Intelligence. AUAI, pp. 969–978.

Infante, G., Jonsson, A., Gómez, V., 2022. Globally optimal hierarchical reinforcement learning for linearly-solvable Markov decision processes. In: AAAI Conference on Artificial Intelligence, Vol. 36. pp. 6970–6977.

Jain, A., Khetarpal, K., Precup, D., 2021. Safe option-critic: Learning safety in the option-critic architecture. Knowl. Eng. Rev. http://dx.doi.org/10.1017/s0269888921000035.

Karl, M., Becker-Ehmck, P., Soelch, M., Benbouzid, D., Smagt, P.v.d., Bayer, J., 2019. Unsupervised real-time control through variational empowerment. In: International Symposium of Robotics Research. Springer, pp. 158–173. http://dx.doi.org/10.1007/978-3-030-95459-8_10.

Khetarpal, K., Klissarov, M., Chevalier-Boisvert, M., Bacon, P.L., Precup, D., 2020. Options of interest: Temporal abstraction with interest functions. In: AAAI Conference on Artificial Intelligence, Vol. 34. AAAI, Menlo Park, pp. 4444–4451. http://dx.doi.org/10.1609/aaai.v34i04.5871.

Klissarov, M., Precup, D., 2021. Flexible option learning. In: Advances in Neural Information Processing Systems, Vol. 34. MIT Press, Cambridge.

Klyubin, A.S., Polani, D., Nehaniv, C.L., 2004. Empowerment: A universal agent-centric measure of control. In: IEEE Congress on Evolutionary Computation, Vol. 1. IEEE, Piscataway, pp. 128–135. http://dx.doi.org/10.1109/cec.2005.1554676.

Lee, L., Eysenbach, B., Parisotto, E., Xing, E., Levine, S., Salakhutdinov, R., 2021. Efficient exploration via state marginal matching. In: AAAI Conference on Artificial Intelligence, Vol. 35, no. 12. pp. 10859–10867.

Li, A.C., Florensa, C., Clavera, I., Abbeel, P., 2019a. Sub-policy adaptation for hierarchical reinforcement learning. In: International Conference on Learning Representations.

Li, C., Ma, X., Zhang, C., Yang, J., Xia, L., Zhao, Q., 2020a. SOAC: The soft option actor-critic architecture. arXiv preprint arXiv:2006.14363.

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

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

Li, S., Zhang, J., Wang, J., Zhang, C., 2021b. Efficient hierarchical exploration with stable subgoal representation learning. In: International Conference on Machine Learning.

Li, S., Zheng, L., Wang, J., Zhang, C., 2020b. Learning subgoal representations with slow dynamics. In: International Conference on Learning Representations.

Machado, M.C., Bellemare, M.G., Bowling, M., 2017. A Laplacian framework for option discovery in reinforcement learning. In: International Conference on Machine Learning. ACM, New York, pp. 2295–2304.

Mayr, M., Ahmad, F., Chatzilygeroudis, K., Nardi, L., Krueger, V., 2022. Skill-based multi-objective reinforcement learning of industrial robot tasks with planning and knowledge integration. In: IEEE International Conference on Robotics and Biomimetics. IEEE, pp. 1995–2002.

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

Osa, T., Tangkaratt, V., Sugiyama, M., 2019. Hierarchical reinforcement learning via advantage-weighted information maximization. In: International Conference on Learning Representations.

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

Pitis, S., Chan, H., Zhao, S., Stadie, B., Ba, J., 2020. Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In: International Conference on Machine Learning. PMLR, pp. 7750–7761.

Pong, V.H., Dalal, M., Lin, S., Nair, A., Bahl, S., Levine, S., 2020. Skew-fit: State-covering self-supervised reinforcement learning. In: International Conference on Machine Learning. pp. 7783–7792.

Rana, K., Xu, M., Tidd, B., Milford, M., Sünderhauf, N., 2022. Residual skill policies: Learning an adaptable skill-based action space for reinforcement learning for robotics. arXiv preprint arXiv:2211.02231.

Riemer, M., Cases, I., Rosenbaum, C., Liu, M., Tesauro, G., 2020. On the role of weight sharing during deep option learning. In: AAAI Conference on Artificial Intelligence, Vol. 34. AAAI, Menlo Park, pp. 5519–5526. http://dx.doi.org/10.1609/aaai.v34i04.6003.

Salge, C., Glackin, C., Polani, D., 2014. Empowerment–An Introduction. In: Guided Self-Organization: Inception, Springer, Epping, Australia, pp. 67–114. http://dx.doi.org/10.1007/978-3-642-53734-9_4.

Savinov, N., Raichuk, A., Marinier, R., Vincent, D., Pollefeys, M., Lillicrap, T., Gelly, S., 2018. Episodic curiosity through reachability. In: International Conference on Learning Representations.

Schoknecht, R., Riedmiller, M., 2003. Reinforcement learning on explicitly specified time scales. Neural Comput. Appl. 12, 61–80.

Sharma, A., Gu, S., Levine, S., Kumar, V., Hausman, K., 2019. Dynamics-aware unsupervised discovery of skills. In: International Conference on Learning Representations.

Song, Y., Wang, J., Lukasiewicz, T., Xu, Z., Xu, M., 2019. Diversity-driven extensible hierarchical reinforcement learning. In: AAAI Conference on Artificial Intelligence, Vol. 33. AAAI, Menlo Park, pp. 4992–4999. http://dx.doi.org/10.1609/aaai.v33i01.33014992.

Sutton, R.S., Precup, D., Singh, S., 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. Artificial Intelligence 112 (1–2), 181–211. http://dx.doi.org/10.1016/s0004-3702(99)00052-1.

Tang, H., Houthooft, R., Foote, D., Stooke, A., Chen, O.X., Duan, Y., Schulman, J., DeTurck, F., Abbeel, P., 2017. exploration: A study of count-based exploration for deep reinforcement learning. In: Advances in Neural Information Processing Systems. pp. 2753–2762.

Tao, R.Y., François-Lavet, V., Pineau, J., 2020. Novelty search in representational space for sample efficient exploration. In: Advances in Neural Information Processing Systems, Vol. 33. pp. 8114–8126.

Wagenmaker, A.J., Chen, Y., Simchowitz, M., Du, S., Jamieson, K., 2022. Reward-free RL is no harder than reward-aware RL in linear Markov decision processes. In: International Conference on Machine Learning. PMLR, pp. 22430–22456.

Zhang, T., Guo, S., Tan, T., Hu, X., Chen, F., 2020. Generating adjacency-constrained subgoals in hierarchical reinforcement learning. In: Advances in Neural Information Processing Systems, Vol. 33. MIT Press, Cambridge, pp. 85–114.

Zhang, J., Yu, H., Xu, W., 2021. Hierarchical reinforcement learning by discovering intrinsic options. In: International Conference on Learning Representations.