ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230386

Exploration and Exploitation in Hierarchical Reinforcement Learning with Adaptive Scheduling

Zhigang Huang[a] **and Quan Liu**[a;*]

[a]School of Computer Science and Technology, Soochow University, 215006, Suzhou, Jiangsu, P.R.China

Abstract. In hierarchical reinforcement learning (HRL), continuous options provide a knowledge carrier that is more aligned with human behavior, but reliable scheduling methods are not yet available. To design an available scheduling method for continuous options, in this paper, the hierarchical reinforcement learning with adaptive scheduling (HAS) algorithm is proposed. It focuses on achieving an adaptive balance between exploration and exploitation during the frequent scheduling 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 options, enabling the agent to focus on different behaviors at different phases. 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, accelerating the convergence rate. We develop a comprehensive experimental analysis scheme. The experimental results demonstrate the high performance and robustness of HAS. Moreover, it provides evidence that adaptive scheduling has a positive effect both on the representation and option policies.

1 Introduction

Hierarchical reinforcement learning (HRL) [14] represents various levels of knowledge with options and combines them through scheduling to acquire temporal abstract solutions. In terms of representation, an option can be either a trajectory that is potentially useful [15], or a high-level landmark from the symbolic model [10]. In terms of scheduling, an option can be either a sequence of primitive actions [4], or a long-term event that traverses the entire environment [6]. From a more fundamental perspective on distributions, we categorize HRL algorithms into **discrete options** and **continuous options**. They are the knowledge carriers of HRL and have different characteristics. Through them, we will identify the limitations and room for improvement of conventional HRL.

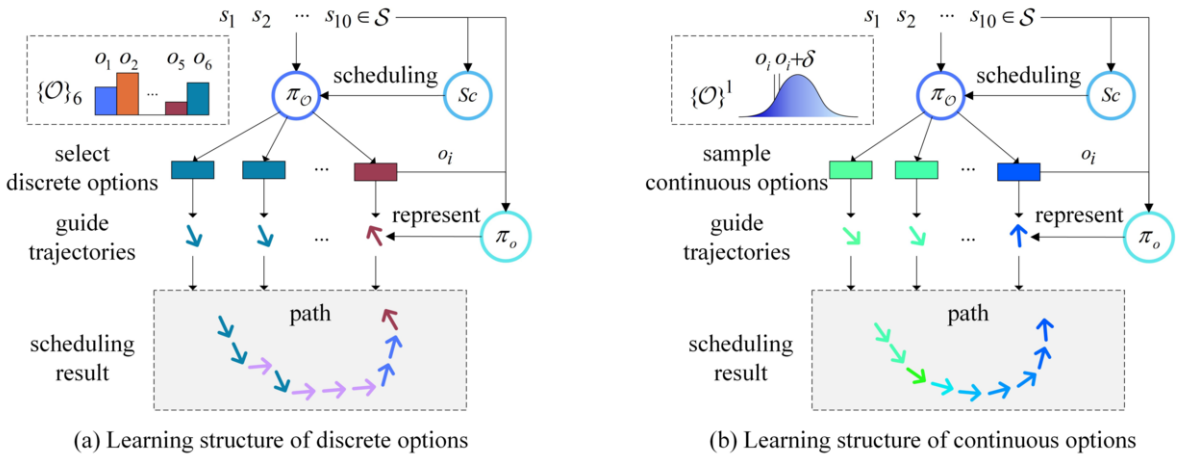
Consider the scenario where a robotic arm pushes a box into a target area. An infinite number of action combinations can be taken to achieve this goal. Any restrictions on its behavior could hinder its success. If rewards can only be earned by completing tasks [32], policies should be more exploratory. By representing sequence actions as options, it is possible to achieve the goal of compressing the state space and simplifying the solution process. However, conventional HRL algorithms [5] rely heavily on discrete options. The limited number requires them to be empirically explainable [28] and non-redundant [30]. Even as the number of discrete options increases, it is still unrealistic for the option policy to traverse each option value. As a result, they are unable to guide diverse trajectories to cover fine-grained state spaces and to acquire rich scheduling combinations. This leads to a robotic arm performing tasks only by explicitly executing meaningful but limited knowledge, resulting in severe performance bottlenecks.

For sparse reward problems with high exploration requirements, such as robot control, it is more realistic and natural to choose continuous options [24] to represent richer knowledge. Their intra-option policy is a continuous function, so option-related value updates 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, they need to be scheduled frequently to discover the best combination solution for tasks. Conventional scheduling methods are not suitable for continuous options. Either they are sensitive to the scheduling parameters [16], or the switching frequency cannot be reasonably controlled [2]. Therefore, our objective is to develop an innovative and practical scheduling method that maximizes the superiority of continuous options, without prior experience, using a framework that is as simple and stable as possible. This method will be of great benefit to HRL applications.

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. Specifically, the main contributions of this paper are as follows:

- We propose an innovative scheduling method to implement adaptive scheduling of continuous options.
- We design an interruption incentive to alleviate excessive exploration, accelerating the convergence rate.
- We validate the action capability of adaptive scheduling using expected t-step distance.
- Our algorithm can effectively solve sparse reward problems in continuous spaces.
- We construct a comprehensive experimental analysis scheme to demonstrate the positive effects of adaptive scheduling.

* Corresponding Author. Email:quanliu@suda.edu.cn



(a) Learning structure of discrete options

(b) Learning structure of continuous options

Figure 1. The HRL learning structures. The option policy $\pi_\mathcal{O}$ generates options under the control of scheduling. The intra-option policy π_o represents options and establishes the guiding relationship between trajectories and options.

2 Background

HRL [6] employs temporal abstraction techniques to represent multi-level knowledge, achieving cross-temporal behavior. A Markov decision process (MDP) [31] and a semi-Markov decision process (SMDP) [1] are its foundations. In this section, we describe the options' characteristics, representation, and scheduling. Our method is inspired by them. We construct a generic HRL structure, as depicted in Figure 1, to reflect their relationship.

2.1 Option

Discrete options [29] are generally obtained by sampling from a discrete distribution. As shown in Figure 1 (a), the high-level controller selects an option o in the state s from the option set $\{\mathcal{O}\}_N$ with the number n. Continuous options are generally obtained by sampling from a continuous distribution. As shown in Figure 1 (b), the high-level controller samples an option o in the state s from the distribution $\{\mathcal{O}\}^n$ with the dimension n.

In the majority of algorithms, the number of discrete options is limited to a single digit [19]. The reason is that massively increasing the number of discrete options is prohibitively expensive and intractable. It becomes impractical for the option policy to traverse each option value, and it tends to generate meaningless options due to the lack of correlation between adjacent discrete options. In contrast, the intra-option policy of continuous options allows value updates to be generalized to neighboring areas. Their trajectories can cover a wide range of fine-grained spaces and assist the agent in learning near-optimal policies. However, the infinite number also makes them less stable. Taking full advantage of the representation superiority of continuous options while avoiding scheduling imbalances is challenging for conventional HRL algorithms.

2.2 Representation

Representation refers to the process by which options are endowed with varying degrees of knowledge [21]. Currently, the dominant representation method applies mutual information [25] to establish the guiding relationship between options (including discrete options and continuous options) and trajectories (including various combinations of states and actions). VIC [9] first introduced mutual information [18] to generate the inferred relationship between options and states. DIAYN [7] created the inferred relationship between options, states, and actions. IST [30] encourages the agent to discover states that are challenging to the policy so that more complex actions can be obtained through options. HIDIO [33] employed continuous options for the first time and compared the effects of different mutual information forms. A typical form of mutual information consists of options and states. It can be written as:

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

2.3 Scheduling

As shown in Figure 1, scheduling is a control mechanism acting on the option policy. The most representative control mechanisms are dynamic scheduling [13] and static scheduling [20]. Dynamic scheduling is based on the option policy gradient theorem about the termination function proposed by OC [2]. The termination function $\beta \in [0, 1]$ is trained using a neural network. The termination probability and the random events determine whether the current option is switched, emphasizing the flexibility of scheduling. A2OC [12] introduced a correction cost to reduce the frequency of option switching.

Static scheduling is based on the MSA concept [26], which switches options at each intra-option step k ($k \geq 1$). It emphasizes the stability of scheduling. SNN4HRL [8] switches the option at each time step, DADS [27] selects different values of k for different tasks, while HAAR [20] uses a gradually decreasing intra-option step k.

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 sec-

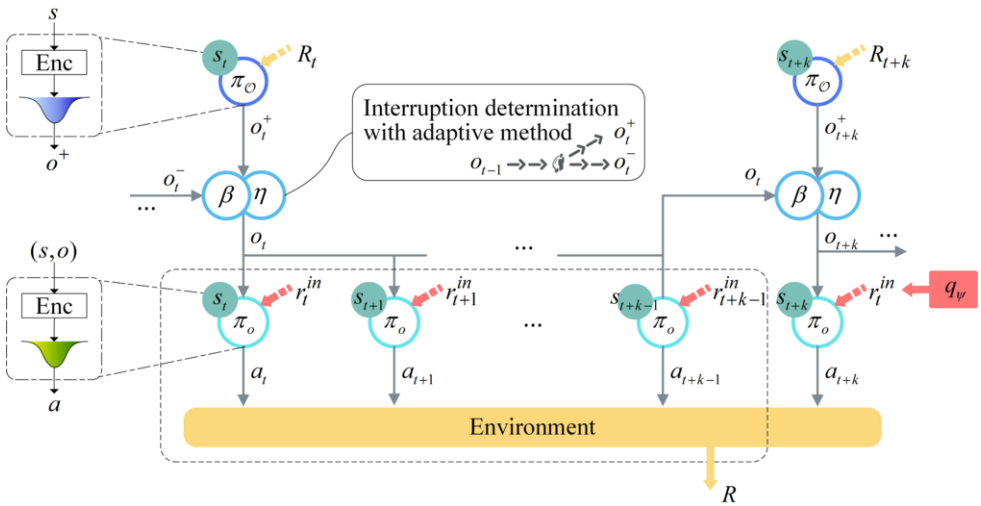


Figure 2. The Framework of HAS. At time t, $\pi_{\mathcal{O}}$ selects o_t^+ in s_t. Next, β and η jointly determine o_t according to o_t^+ and o_t^-. Then, π_o selects a_t in (s_t, o_t). An agent interacts with the environment and reaches s_{t+1}. Afterward, π_o selects actions in new extended states 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.

tion, 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

Figure 2 illustrates the framework of HAS. 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. β denotes a termination function. It 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^+. When the termination function returns 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.

The high-level objective function $J_{\pi_{\mathcal{O}}}$ of HAS can be written:

$$J_{\pi_{\mathcal{O}}} = \mathbb{E}_{\pi_{\mathcal{O}}}\left[\sum_{t=0}^{T}\gamma^{t/\tilde{k}}R_t + \alpha H_{\mathcal{O},\beta}\right],\ where\ R_t = \sum_{l=t}^{t+\tilde{k}} r_l^{env} \tag{2}$$

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. $H_{\mathcal{O},\beta}$ denotes the combined entropy of $\pi_{\mathcal{O}}$ and β. α is the temperature coefficient.

HAS acquires intrinsic rewards r^{in} using mutual information. Its low-level objective function J_{π_o} can be written:

$$J_{\pi_o} = \mathbb{E}_{\pi_o}\left[r_t^{in} + \alpha H_o\right], \tag{3}$$
$$where\ r_t^{in} = \log q_\psi(o_t|s_{t+1}) - \log p(o_t)$$

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 Equation (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. The number of continuous options is infinite, and there is a small difference between trajectories guided by similar continuous options. Moreover, the option policy is unlikely to generate the same continuous option twice. Thus, frequent scheduling can provide an agent with more opportunities to eliminate redundant behaviors and acquire near-optimal policies.

Specifically, the adaptive scheduling method builds on a multi-step static scheduling process and develops a new switching judgment mechanism for dynamic scheduling. First, the multi-step static scheduling of $k > 1$ provides a fundamental exploration capability, enabling the agent to leave its local area. Second, the continuous option policy involves a certain degree of randomness. It is impossible to guarantee that the estimated option o^+ will be superior to the previous option o^-. If the switching judgment mechanism is based on the advantage function $A(s,o')$ as conventional dynamic scheduling methods[12], it is easy to generate a suboptimal continuous option. 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^+:

$$\beta(o^+|s,o^+,o^-) = \frac{\exp Q^{\pi_{\mathcal{O}}}(s,o^+)}{\exp Q^{\pi_{\mathcal{O}}}(s,o^-) + \exp Q^{\pi_{\mathcal{O}}}(s,o^+)} \tag{4}$$

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:

$$P(\cdot|s,o^-) = (1-\beta(o^+))P(o^-|s,o^-) + \beta(o^+)P(o^+|s,o^-) \tag{5}$$

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

$$\begin{aligned} V^{\pi_{\mathcal{O}}}(s_t) &= \mathbb{E}_{o_t \sim \pi_{\mathcal{O}}(\cdot|s_t)}\left[Q^{\pi_{\mathcal{O}}}(s_t, o_t)\right] \\ &= \mathbb{E}_{o_t^+ \sim \pi_{\mathcal{O}}(\cdot|s_t)}\big[(1-\beta(o_t^+))Q^{\pi_{\mathcal{O}}}(s_t, o_t^-) + \\ &\quad \beta(o_t^+)Q^{\pi_{\mathcal{O}}}(s_t, o_t^+)\big] \end{aligned} \tag{6}$$

As a result of the interruption judgment mechanism in Equation (6), the agent's behavior exhibits different emphases during training. 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. 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.

3.3 Exploration capability

With adaptive scheduling, an agent can explore a larger degree of space while maintaining fine-grained behavioral control. To demonstrate that adaptive scheduling possesses superior exploration potential, we first make the following definition. Our objective is to solve sparse reward problems, such as robot control, 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 [7]. We further assume that the agent moves a unit distance d independently and identically in any direction.

We apply the expected t-step distance $\delta_{Sc(k)\to t}(s)$ [26] to measure how far an agent moves on average. It is defined as the expected distance that the agent will move after t time steps from the state s using the scheduling method $Sc(k)$:

$$\delta_{Sc(k)\to t}(s) = \sum_{\bar{s}\in\mathcal{S}} P_{Sc(k)\to t}(s, \bar{s}) D(s, \bar{s}) \tag{7}$$

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 $Sc(k)$.

Figure 3 illustrates the arrival states of adaptive scheduling $AS(2)$ and static scheduling $SS(2)$ in time steps $t = 4$ and $t = 6$ under the random walk. Their expected t-step distances are listed in Table 1. The arrival results at time step $t = 1$ are not recorded because they are identical across all methods.

Table 1. The expected t-step distance. It represents the exploration capabilities of different scheduling methods.

	$t=4$	$t=6$
Adaptive scheduling $AS(2)$	**2.77**	**3.45**
Static scheduling $SS(2)$	2.52	2.88

As can be seen in Figure 3, the arrival states of static scheduling method 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 the static scheduling method remain concentrated in areas $D \leq 4d$, 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. Dynamic scheduling $DS(1)$ can be seen as a weakened adaptive scheduling, so its expected t-step distance is lower as well.

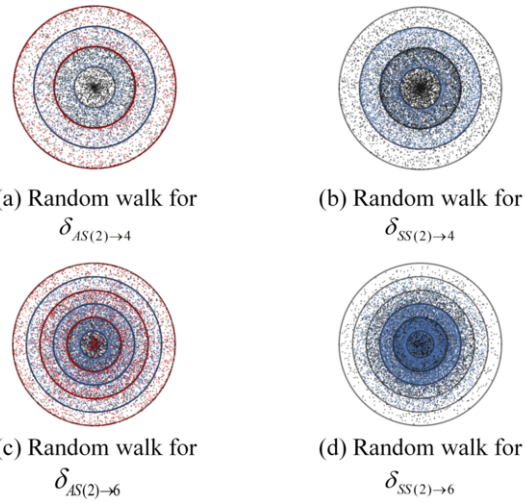


Figure 3. The arrival states of different scheduling methods in time steps $t = 4$ (a-b) and $t = 6$ (c-d). The data distributions are applied to analyze the exploration capabilities of different scheduling methods. Black dots indicate the states for random walks, blue dots indicate the states for static walks, and red dots indicate the states for non-interruption scheduling. Each task runs $10,000$ episodes.

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.

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 become trapped in an excessive exploration dilemma, resulting in a slower convergence rate. To alleviate this problem, we propose an interruption incentive η based on the annealing mechanism. The purpose of annealing is to prevent the destruction of the adaptive balance. The revised interruption probability should not affect the choice of the optimal option during the mid-late training phase. Thus, we design the termination function $\beta_\eta(o^+)$ with the interruption incentive as follows:

$$\beta_\eta(o^+) = min\left(\beta(o^+) - \mathbb{I}_{t<T*\ell}\left[(1-\frac{t}{T*\ell})\eta\right], 0\right) \tag{8}$$

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 search with ℓ on different tasks may produce superior results. However, our experiments indicate that 2/5 is sufficient. Unlike HAAR [20], the interruption incentive does not explicitly change the option length. Scheduling still depends mainly on the option value.

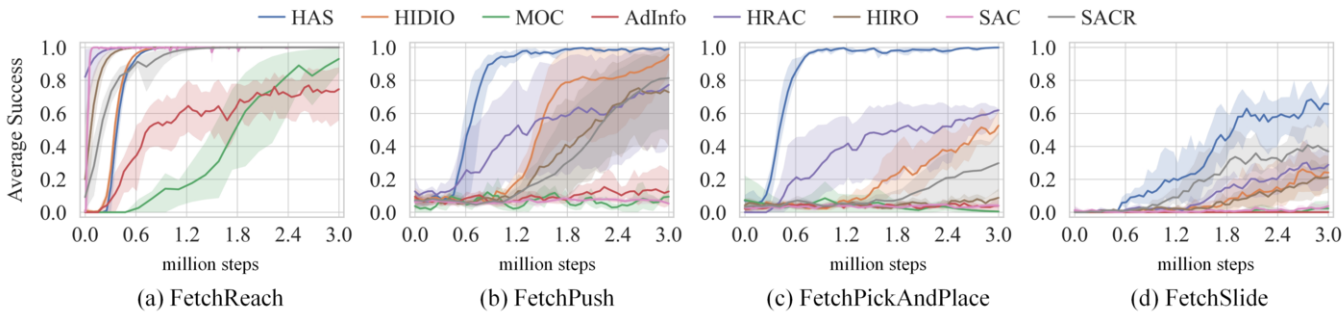


Figure 4. Comparison of the average success rates of HAS against baselines. The shaded areas indicate standard deviations.

4 Experimental results

We develop an exhaustive experimental analysis scheme. First, it demonstrates the superiority of HAS for solving robot control problems with sparse rewards in continuous spaces. Second, it verifies the effectiveness of our proposed techniques through ablation studies. Third, it identifies the operation mechanism of the adaptive scheduling method and analyzes the root of HAS advantages from multiple perspectives.

4.1 Environments

Our experiment applies Robotics environment [3]. It is a sparse reward environment in a continuous three-dimensional state space. The agent is penalized -0.1 for each step, except when it touches the target or pushes/grabs/hits the object. The task is deemed successful if it is not punished at the end of the maximum episode step. Otherwise, it is deemed unsuccessful.

4.2 Experimental Setup

We select a total of five representative HRL baselines that are highly correlated with HAS, including **HIDIO** [33], **MOC** [17], **AdInfo** [23], **HRAC** [34], and **HIRO** [22]. Among them, HIDIO is the first HRL algorithm to employ continuous options. MOC and AdInfo are typical HRL algorithms based on discrete options. HRAC and HIRO are up-to-date HRL algorithms based on subgoals. Besides, two predominant underlying algorithms are also applied as baselines, including **SAC** [11] and **SACR**. SACR 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. They meet the requirements of up-to-date technology, high quality, and identical fields, ensuring the credibility of comparisons. Additionally, FetchPush, FetchPickAndPlace, and FetchSlide have sequential decision processes, in which the agent must learn how to the control arm to interact with a target, making learning more challenging for HAS and baselines.

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.

4.3 Comparison

Figure 4 illustrates the average success rates of HAS against baselines. It can be seen that the majority of algorithms can complete FetchReach, but only HAS, HIDIO, and HRAC can complete FetchPush and FetchPickAndPlace with a success rate close to 100%. Among them, HAS converges the fastest. HAS 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. The success rates of MOC and AdInfo are almost always below 5% except for FetchReach. The poor performance is due to the limitations of discrete options, which cannot provide a variety of representation and scheduling possibilities. We consider this to be the performance bottleneck. While HRAC and HIRO outperform the HRL algorithms based on discrete options, they clearly lag behind HAS.

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

4.4 Ablation study

We construct modular and parametric ablation experiments. They are used to determine whether the components of HAS play a positive role and to study the sensitivity of HAS to fundamental parameters.

Modular 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 Figure 5. When the continuous option method is removed, the switching judgment mechanism reverts to the loss update method based on the advantage function [12].

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, processing techniques, and parameters are as similar as possible to HAS, it is almost incapable of accomplishing most tasks. It is demonstrated that discrete options suffer from a severe performance bottleneck and cannot benefit from our methods. Performance and stability of both HAS-dynamic and HAS-static are impaired. It is shown that dynamic

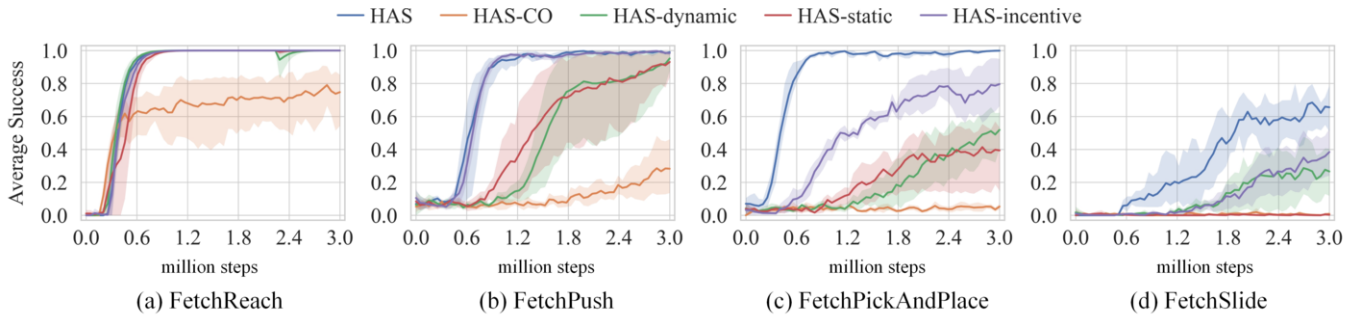


Figure 5. Modular ablation for HAS. The method following "-" indicates the component to be removed.

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. It demonstrates that the interruption incentive helps the agent to escape the dilemma of excessive exploration.

Parametric ablation. Option dimension n and intra-option step k are the fundamental parameters of adaptive scheduling. We ablate each parameter separately. Their terminal average success rates under different option dimensions and intra-option steps are listed in Table 2.

Table 2. Comparison of the terminal success rate (%) of HAS under different option dimensions and intra-option steps. " + dim=n" denotes the algorithm using n-dimensional option, and " + step=k" denotes the algorithm using intra-option step k. The best one is highlighted in bold.

Algorithm	Fetch-Reach	Fetch-Push	Fetch-PickAndPlace	Fetch-Slide	Average
HAS+dim=2	0.95	0.95	0.96	0.12	0.75
HAS+dim=4	**1.00**	**0.99**	**1.00**	0.18	0.79
HAS+dim=6	**1.00**	0.98	0.98	**0.64**	**0.90**
HAS+step=2	**1.00**	0.98	0.78	0.42	0.79
HAS+step=3	**1.00**	**0.99**	**1.00**	**0.64**	**0.90**
HAS+step=5	**1.00**	0.97	0.71	0.21	0.72

As shown in Table 2, HAS+dim=6 and HAS+step=3, the HAS algorithm with standard parameters, have the highest terminal success rate with 90%. HAS+dim=2 and HAS+dim=4 also have reasonable success rates. The results reflect that the option is the carrier of knowledge representation. An appropriate option dimension n is conducive to representing rich knowledge. In addition, as the step increases, the performance first increases and then decreases. HAS+step=5 is less than HAS+step=2 with 7%. It reminds us that it is important to avoid taking too many intra-option steps. If the action sequences are too long, the agent may miss target areas or push objects out of these areas.

In summary, the HAS is sufficiently robust to changes in fundamental parameters within a certain range. It is well suited to higher option dimension n and less sensitive to intra-option step k. This study demonstrates that HAS benefits from adaptive scheduling.

4.5 *Effects of adaptive scheduling*

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

4.5.1 *Adaptive mechanism*

Figure 6 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.

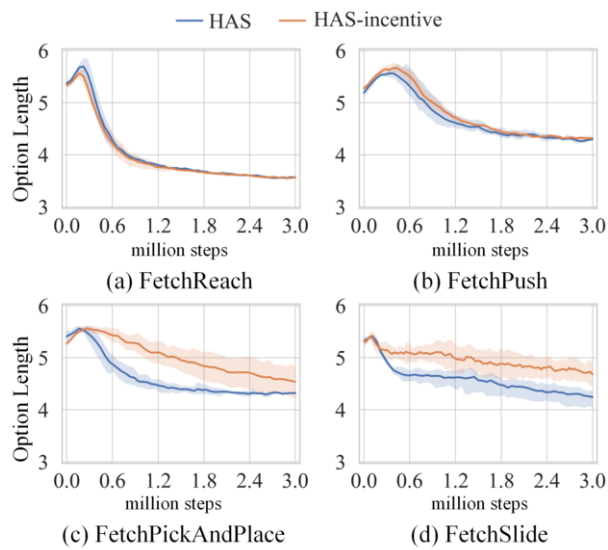


Figure 6. Variation in the option length of HAS and HAS-incentive.

It can be seen that the option length of HAS is larger than the intra-option step ($k = 3$) in all tasks, indicating that non-interruption 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. Moreover, the decrease rate of the option length of HAS is apparently larger than HAS-incentive. This indicates that the interruption incentive can alleviate the dilemma of excessive exploration. 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, and the option length continuously decreases. At this moment, adaptive scheduling focuses primarily on exploitation.

4.5.2 *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 on FetchPickAndPlace and FetchSlide, and then compare their representation capabilities by drawing the trajectories of a set of options, as depicted in Figure 7. It can be seen that HIDIO loses 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.

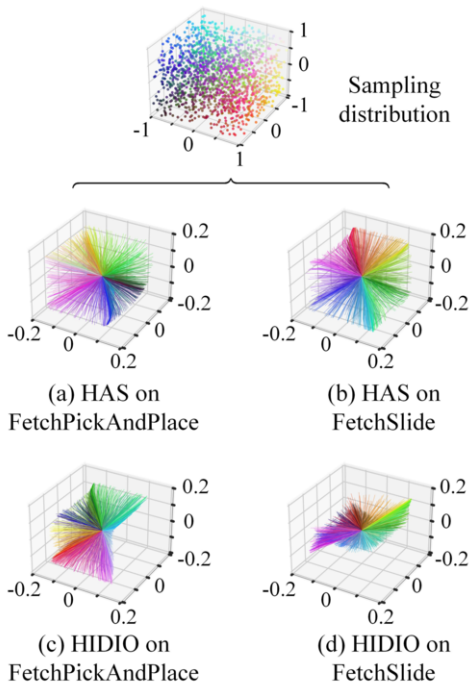


Figure 7. Trajectories of $\{O\}^3$ HAS (a-b) and $\{O\}^3$ HIDIO (c-d). For each task, $10,000$ continuous options are randomly sampled, with an intra-option step $k = 6$. Line colors correspond to the sampling distribution as the scatter plot on the top.

4.5.3 *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 Figure 4, 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 performance improvement process. Here, we select several HAS and HIDIO models with superior performance on FetchPickAndPlace and FetchSlide, and then illustrate their paths in Figure 8. We manually position the target away from the robotic arm's initial position to intuitively illustrate their disparity.

As shown in Figure 8, 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

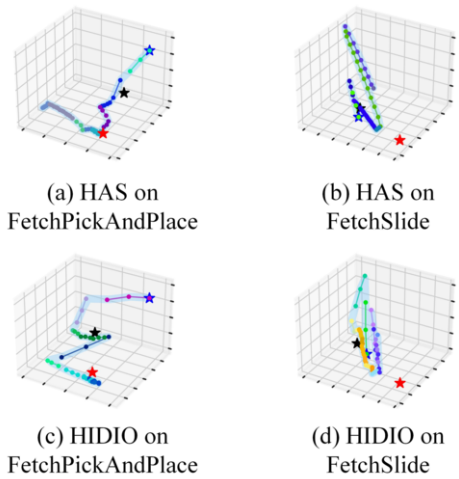


Figure 8. Task path of HAS (a-b) and HIDIO (c-d). Dots indicate the states visited by the agent, whose colors correspond to those shown in Figure 7. The light blue line indicates the route of the movement, and line segments indicate paths that are guided by the same option. The blue pentagram indicates the initial state, the red indicates the target state, and the black indicates the box that can be moved.

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. 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.

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 sparse reward problems in continuous spaces. It emphasizes the balance between exploration and exploitation during frequent scheduling. Through multi-step static scheduling and a value judgment switching mechanism, the agent's behavior can be adaptively adjusted. As a general principle, finding an appropriate option policy and switching frequency provides the agent with more opportunities to learn a near-optimal policy, i.e., obtain more rewards with fewer steps. It is also the fundamental goal of RL.

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. All components contribute positively, and HAS is less sensitive to fundamental parameters. Moreover, 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.

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).

References

[1] Sasha Abramowitz and Geoff Nitschke, 'Towards run-time efficient hierarchical reinforcement learning', in *IEEE Congress on Evolutionary Computation*, pp. 1–8. IEEE, (2022).

[2] Pierre Luc Bacon, Jean Harb, and Doina Precup, 'The option-critic architecture', in *AAAI Conference on Artificial Intelligence*, pp. 1726–1734, Menlo Park, (2017). AAAI.

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba, 'Openai gym', *arXiv preprint arXiv:1606.01540*, (2016).

[4] Daesol Cho, Jigang Kim, and H Jin Kim, 'Unsupervised reinforcement learning for transferable manipulation skill discovery', *IEEE Robotics and Automation Letters*, **7**(3), 7455–7462, (2022).

[5] Fan Ding and Fei Zhu, 'Hliferl: A hierarchical lifelong reinforcement learning framework', *Journal of King Saud University-Computer and Information Sciences*, **34**(7), 4312–4321, (2022).

[6] Ambedkar Dukkipati, Rajarshi Banerjee, Ranga Shaarad Ayyagari, and Dhaval Parmar Udaybhai, 'Learning skills to navigate without a master: A sequential multi-policy reinforcement learning algorithm', in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2483–2489. IEEE, (2022).

[7] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine, 'Diversity is all you need: Learning skills without a reward function', in *International Conference on Learning Representations*, (2018).

[8] Carlos Florensa, Yan Duan, and Pieter Abbeel, 'Stochastic neural networks for hierarchical reinforcement learning', in *International Conference on Learning Representations*, (2017).

[9] Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra, 'Variational intrinsic control', *arXiv preprint arXiv:1611.07507*, (2016).

[10] Lin Guan, Sarath Sreedharan, and Subbarao Kambhampati, 'Leveraging approximate symbolic models for reinforcement learning via skill diversity', in *International Conference on Machine Learning*, pp. 7949–7967. PMLR, (2022).

[11] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine, 'Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor', in *International Conference on Machine Learning*, pp. 1861–1870, New York, (2018). ACM.

[12] Jean Harb, Pierre Luc Bacon, Martin Klissarov, and Doina Precup, 'When waiting is not an option: Learning options with a deliberation cost', in *AAAI Conference on Artificial Intelligence*, volume 32, pp. 3165–3172, Menlo Park, (2018). AAAI.

[13] Anna Harutyunyan, Peter Vrancx, Pierre-Luc Bacon, Doina Precup, and Ann Nowe, 'Learning with options that terminate off-policy', in *AAAI Conference on Artificial Intelligence*, pp. 3173–3182, Menlo Park, (2017). AAAI.

[14] Shuncheng He, Yuhang Jiang, Hongchang Zhang, Jianzhun Shao, and Xiangyang Ji, 'Wasserstein unsupervised reinforcement learning', in *AAAI Conference on Artificial Intelligence*, volume 36, pp. 6884–6892, Menlo Park, (2022). AAAI.

[15] Liangyu Huo, Zulin Wang, Mai Xu, and Yuhang Song, 'A task-agnostic regularizer for diverse subpolicy discovery in hierarchical reinforcement learning', *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 1932–1944, (2023).

[16] Khimya Khetarpal, Martin Klissarov, Maxime Chevalier-Boisvert, Pierre Luc Bacon, and Doina Precup, 'Options of interest: Temporal abstraction with interest functions', in *AAAI Conference on Artificial Intelligence*, volume 34, pp. 4444–4451, Menlo Park, (2020). AAAI.

[17] Martin Klissarov and Doina Precup, 'Flexible option learning', in *Advances in Neural Information Processing Systems*, volume 34, Cambridge, (2021). MIT Press.

[18] Alexander S Klyubin, Daniel Polani, and Chrystopher L Nehaniv, 'Empowerment: A universal agent-centric measure of control', in *IEEE Congress on Evolutionary Computation*, volume 1, pp. 128–135, Piscataway, (2004). IEEE.

[19] David Kuric and Herke van Hoof, 'Meta reinforcement learning for fast adaptation of hierarchical policies', in *International Conference on Learning Representations*, (2022).

[20] Siyuan Li, Rui Wang, Minxue Tang, and Chongjie Zhang, 'Hierarchical reinforcement learning with advantage-based auxiliary rewards', in *Advances in Neural Information Processing Systems*, pp. 1409–1419, Cambridge, (2019). MIT Press.

[21] Jinxin Liu, Hao Shen, Donglin Wang, Yachen Kang, and Qiangxing Tian, 'Unsupervised domain adaptation with dynamics-aware rewards in reinforcement learning', in *Advances in Neural Information Processing Systems*, volume 34, pp. 28784–28797, Cambridge, (2021). MIT Press.

[22] Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine, 'Data-efficient hierarchical reinforcement learning', in *Advances in Neural Information Processing Systems*, pp. 3303–3313, Cambridge, (2018). MIT Press.

[23] Takayuki Osa, Voot Tangkaratt, and Masashi Sugiyama, 'Hierarchical reinforcement learning via advantage-weighted information maximization', in *International Conference on Learning Representations*, (2019).

[24] Krishan Rana, Ming Xu, Brendan Tidd, Michael Milford, and Niko Sünderhauf, 'Residual skill policies: Learning an adaptable skill-based action space for reinforcement learning for robotics', *arXiv preprint arXiv:2211.02231*, (2022).

[25] Christoph Salge, Cornelius Glackin, and Daniel Polani, *Empowerment–an introduction*, Guided Self-Organization: Inception, Springer, Epping, Australia, 2014.

[26] Ralf Schoknecht and Martin Riedmiller, 'Reinforcement learning on explicitly specified time scales', *Neural Computing and Applications*, **12**, 61–80, (2003).

[27] Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman, 'Dynamics-aware unsupervised discovery of skills', in *International Conference on Learning Representations*, (2019).

[28] Wonil Song, Sangryul Jeon, Hyesong Choi, Kwanghoon Sohn, and Dongbo Min, 'Weakly-supervised learning of disentangled and interpretable skills for hierarchical reinforcement learning', in *International Conference on Learning Representations*, (2022).

[29] Qiangxing Tian, Jinxin Liu, Guanchu Wang, and Donglin Wang, 'Unsupervised discovery of transitional skills for deep reinforcement learning', in *IEEE International Joint Conference on Neural Networks*, pp. 1–8. IEEE, (2021).

[30] Qiangxing Tian, Guanchu Wang, Jinxin Liu, Donglin Wang, and Yachen Kang, 'Independent skill transfer for deep reinforcement learning', in *International Joint Conference on Artificial Intelligence*, pp. 2901–2907. Morgan Kaufmann, (2019).

[31] Andrew J Wagenmaker, Yifang Chen, Max Simchowitz, Simon Du, and Kevin Jamieson, 'Reward-free rl is no harder than reward-aware rl in linear markov decision processes', in *International Conference on Machine Learning*, pp. 22430–22456. PMLR, (2022).

[32] Yuguang Yang, Michael A Bevan, and Bo Li, 'Hierarchical planning with deep reinforcement learning for 3d navigation of microrobots in blood vessels', *Advanced Intelligent Systems*, 2200168, (2022).

[33] Jesse Zhang, Haonan Yu, and Wei Xu, 'Hierarchical reinforcement learning by discovering intrinsic options', in *International Conference on Learning Representations*, (2021).

[34] Tianren Zhang, Shangqi Guo, Tian Tan, Xiaolin Hu, and Feng Chen, 'Generating adjacency-constrained subgoals in hierarchical reinforcement learning', in *Advances in Neural Information Processing Systems*, volume 33, pp. 85–114, Cambridge, (2020). MIT Press.