# Multi-agent Deep Covering Option Discovery

Jiayu Chen, Marina Haliem, Tian Lan, and Vaneet Aggarwal

***Abstract*—The use of options can greatly accelerate exploration in RL, especially when only sparse reward signals are available. While option discovery methods have been proposed for individual agents, in MARL settings, discovering collaborative options that can coordinate the behavior of multiple agents and encourage them to jointly visit under-explored regions of the state space has not been considered. In this paper, we propose a novel framework for multi-agent deep covering option discovery. Specifically, it first leverages an attention mechanism to find collaborative agent sub-groups that would benefit most from coordination. Then, a hierarchical algorithm based on soft actor-critic, namely H-MSAC, is developed to learn the multi-agent options for each sub-group and then to integrate them through a high-level policy. This hierarchical option construction allows our framework to strike a balance between scalability and effective collaboration among the agents. The evaluation based on multi-agent collaborative tasks shows that the proposed algorithm can effectively capture agent interaction during learning and significantly outperforms prior works using single-agent options or no options, in terms of both faster exploration and higher task rewards.**

***Index Terms*—Multi-agent Reinforcement Learning, Option Discovery, Deep Covering Options**

## I. Introduction

Options discovery [1] enables temporally-abstract actions to be constructed in the reinforcement learning process. It can greatly improve the performance of reinforcement learning agents by representing actions at different time scales. Among recent developments on the topic, *Covering Option Discovery* [2] has been shown to be a promising approach. It leverages Laplacian matrix extracted from the state-transition graph induced by the dynamics of the environment. To be specific, the second smallest eigenvalue of the Laplacian matrix, known as the algebraic connectivity of the graph, is considered as a measure of how well-connected the graph is [3]. In this case, it uses the algebraic connectivity as an intrinsic reward to train the option policy, with the goal of connecting the states that are not well-connected, encouraging the agent to explore infrequently-visited regions, and thus minimizing the agent's expected cover time of the state space. Recently, deep learning techniques have been developed to extend the use of covering options to large/infinite state space [4], e.g., *Deep Covering Option Discovery*. However, these efforts focus on discovering options for individual agents. Discovering collaborative options that encourage multiple agents to jointly visit under-explored regions of the state space has not been considered.

In this paper, we propose a novel framework for multi-agent deep covering option discovery. Recent works [5], [6], [7] compute options with exploratory behaviors for each individual agent by considering only its own state transitions, and then learn to collaboratively leverage these individual options. However, our proposed framework directly recognize joint options composed of multiple agents' temporally-abstract action sequences to encourage joint exploration. We note that in practical scenarios, multi-agent tasks can often be divided into a series of sub-tasks and each sub-task can be completed by a sub-group of the agents. Our proposed algorithm leverages an attention mechanism [8] in the option discovery process to quantify agents interactions and to find collaborative agent sub-groups. Thus, we can train a set of multi-agent options for each sub-group to complete their sub-tasks and then integrate them through a high-level policy. This sub-group partitioning and hierarchical learning structure can effectively construct collaborative options that jointly coordinate the exploration behavior of multiple agents, while keeping the algorithm scalable in practice.

The main contributions of our work are as follows: (1) We propose multi-agent deep covering option discovery and demonstrate that the use of multi-agent options can further improve the performance of MARL agents compared with single-agent options. (2) We propose to leverage an attention mechanism in the discovery process to enable agents to find peer agents that it should interact with closely and form sub-groups. (3) We propose a hierarchical MARL algorithm based on soft actor-critic [9], which integrates the training of intra-option policies (for constructing options) and the high-level policy (for integrating the options). The proposed algorithm, which is evaluated on MARL collaborative tasks, significantly outperforms prior works in terms of faster exploration and higher task rewards.

## II. Related Work

**Option Discovery.** The option framework was proposed in [1], which extends the usual notion of actions to include options — closed-loop policies for taking actions over a period of time. Formally, a set of options defined over an MDP constitutes a semi-MDP (SMDP), where the SMDP actions (options) are no longer black boxes, but policies in the base MDP which can be learned in their own right. In literature, lots of option discovery algorithms utilize the task-dependent reward signals generated by the environment, such as [10], [11], [12], [13]. However, these algorithms require dense reward signals which are usually hard to obtain. Therefore, [14] proposed an approach to generate options through maximizing an information theoretic objective so that each option can generate diverse behaviors. It learns useful skills/options without reward signals and thus can be applied in environments where only sparse rewards are available.

On the other hand, [15], [2] focused on *Covering Option Discovery*, a method which is also not based on the task-

J. Chen, M. Haliem, and V. Aggarwal are with Purdue University, West Lafayette IN 47907, USA, email: {chen3686,mwadea,vaneet}@purdue.edu. T. Lan is with the George Washington University, Washinton DC 20052, USA, email:tlan@gwu.edu.

dependent signals but on the Laplacian matrix of the environment's state-transition graph. This method aims at minimizing the expected cover time of the state space with a uniformly random policy. To realize this, it augments the agent's action set with options obtained from the eigenvector associated with the second smallest eigenvalue (algebraic connectivity) of the Laplacian matrix. However, this Laplacian-based framework can only be applied to tabular settings. To mitigate this issue, [4] proposed *Deep Covering Option Discovery* to combine covering options with modern representation learning techniques for the eigen-function estimation, which can be applied in domains with infinite state space. In [4], the authors compared their approach with the one proposed by [14]: both approaches are sample-based and scalable to large-scale state space, but RL agents with deep covering options have better performance on the same benchmarks.

Note that all the approaches mentioned above are for single-agent scenarios and we will extend the adoption of deep covering options to multi-agent reinforcement learning.

**Options in multi-agent scenarios.** As mentioned in Section I, most of the researches about adopting options in MARL tried to define or learn the options for each individual agent first, and then learn the collaborative behaviors among the agents based on their extended action sets – {primitive actions, individual options}. Therefore, the options they use are still single-agent options, and the coordination in the multi-agent system can only be shown/utilized in the option-choosing process while not the option discovery process. We can classify these works by the option discovery methods they used: [16], [17] directly defined the options based on their task without the learning process; [18], [5], [6] learned the options based on the task-related reward signals generated by the environment; [7] trained the options based on a reward function that is a weighted summation of the environment reward and the information theoretic reward term proposed in [14].

In this paper, we propose to construct multi-agent deep covering options using the Laplacian-based framework mentioned above. Also, in an N-agent system, there may be not only N-agent options, but also one-agent options, two-agent options, etc. In this case, we divide the agents into some sub-groups based on their interaction relationship first, which is realized through the attention mechanism [8], and then construct the interaction patterns (multi-agent options) for each sub-group accordingly. Through these improvements, the coordination among agents is considered in the option discovery process, which has the potential to further improve the performance of MARL agents.

## III. Background

Before extending deep covering options to the multi-agent setting, we will introduce the formal definition of the option framework and some key issues of deep covering options.

### A. Formal Definition of Option:

In this paper, we use the term *options* for the generalization of primitive actions to include temporally-extended courses of actions. As defined in [1], an option $\omega$ consists of three components: an intra-option policy $\pi_\omega : \mathcal{S}$ x $\mathcal{A} \rightarrow [0, 1]$, a termination condition $\beta_\omega : \mathcal{S} \rightarrow [0, 1]$, and an initiation set $I_\omega \subseteq \mathcal{S}$. An option $< I_\omega, \pi_\omega, \beta_\omega >$ is available in state $s$ if and only if $s \in I_\omega$. If the option $\omega$ is taken, actions are selected according to $\pi_\omega$ until $\omega$ terminates stochastically according to $\beta_\omega$. Therefore, in order to get an option, we need to train/define the intra-option policy, and give out the definition of the termination condition and initiation set.

### B. Deep Covering Option Discovery

As descried in [4], they get the deep covering options by greedily maximizing the state-space graph's algebraic connectivity – the second smallest eigenvalue, so as to minimize the expected cover time of the state space. To realize this, they first compute the eigenfunction $f$ associated with the algebraic connectivity by minimizing $G(f)$:

$$G(f) = \frac{1}{2}\mathbb{E}_{(s,s')\sim\mathcal{H}}[(f(s) - f(s'))^2] + \eta\mathbb{E}_{s\sim\rho,s'\sim\rho} \\ [(f(s) - 1)(f(s') - 1) + f(s)^2 f(s')^2] \tag{1}$$

where $\mathcal{H}$ is the set of sampled state-transitions and $\rho$ is the distribution of the states in $\mathcal{H}$. Note that this is a sample-based approach and thus can scale to infinite state-space domains. Then, based on the computed $f$, they define the termination set as a set of states where the $f$ value is smaller than the $k$-th percentile of the $f$ values on $\mathcal{H}$. Accordingly, the initiation set is defined as the complement of the termination set. As for the intra-option policy, they train it through maximizing the reward, $r(s, a, s') = f(s) - f(s')$, to encourage the agent to explore the states with lower $f$ values, i.e., the less-explored states in the termination set.

In this paper, we will compute $f$ of the joint observation space of certain multiple agents, and then learn multi-agent options based on it to encourage the joint exploration and thus increase the connectivity of the joint observation space.

## IV. Proposed Approach

In this section, we will introduce *Multi-agent Deep Covering Option Discovery* and how to adopt it in a MARL setting. First, we will give out the hierarchical framework of the algorithm and the key objective functions to optimize. Then, we will show how to integrate the attention mechanism in the network design. Finally, we will provide **H-MSAC**, a hierarchical MARL algorithm based on soft actor-critic.

### A. Main Framework

In order to take advantage of options in the learning process, we adopt a hierarchical RL framework shown as Algorithm 1. Typically, we train a RL agent to select among the primitive actions, aiming to maximize the accumulated reward. In our algorithm framework, we view this agent as a one-step option – primitive option. When getting a new observation, the hierarchical RL agent first decides on which option $\omega$ to use according to the high-level policy, and then decides on the action (primitive action) to take based on the corresponding intra-option policy $\pi_\omega$. As mentioned in Section III, once an

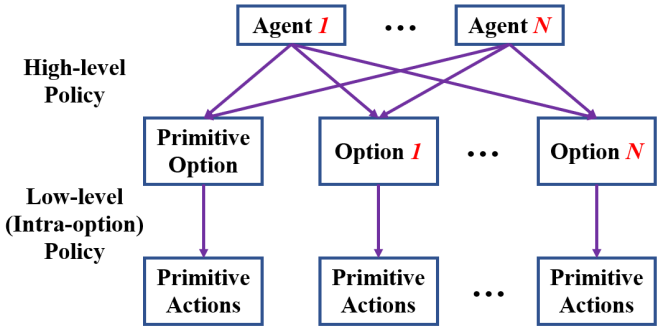



Fig. 1: The agent first decides on which option $\omega$ to use according to the high-level policy, and then decides on the action (primitive action) to take based on the corresponding intra-option policy $\pi_\omega$. **Primitive option**: Typically, we train a RL agent to select among the primitive actions; we view this agent as a special option, whose intra-option policy lasts for only one step. **Option 1** $\sim$ **N**: Based on the attention mechanism, each agent can figure out which agents to collaborate closely and form a sub-group with, so there are at most $N$ sub-groups (duplicate ones need to be eliminated), and we need to train a multi-agent option for each sub-group.

**Algorithm 1** Main Framework

**Input**: primitive option $A$, high-level policies for each agent $\pi_{1:N}$ and corresponding Q-functions $Q_{1:N}$, generation times of options $N_\omega$, generation frequency $N_{int}$;
**Initialize** the set of options $\Omega' \leftarrow \{A\}$
**Create** an empty replay buffer B
**Set** $n_\omega \leftarrow 0$
**for** episode $i = 1$ to $N_{epi}$ **do**
  **Collect** a trajectory $\tau_i$ by repeating this process until done: choose an available option from $\Omega'$ according to $\pi_{1:N}$, and then execute the corresponding intra-option policy until it terminates
  **Update** $B$ with $\tau_i$
  **if** $i \ mod \ N_{int} == 0$ and $n_\omega < N_\omega$ **then**
    **Generate** a set of multi-agent options $\Omega$ using Algorithm 2 based on trajectories in $B$
    **Update** $\Omega'$: $\Omega' \leftarrow \{A\} \cup \Omega$
    **Update** $n_\omega$: $n_\omega \leftarrow n_\omega + 1$
  **end if**
  **Sample** trajectories $\tau_{1:batchsize}$ from $B$
  **Update** $\pi_{1:N}$, $Q_{1:N}$ and options within $\Omega'$ using **H-MSAC** (defined in Section IV-C) based on $\tau_{1:batchsize}$
**end for**

option is taken, the actions will be selected according to $\pi_\omega$ until the option terminates, so we do not decide on a new option until last option terminates.

Except for the primitive option, other options are extracted from transitions collected in the training process. To realize this, we extend Algorithm 1 of [4] and Equation (1) from a single-agent scenario to a multi-agent scenario to obtain our Algorithm 2 and Equation (2). Through this extension, collaboration among the agents is considered not only at a high level, but also in the option discovery process, which can further improve the performance in scenarios where close collaboration is required.

**Algorithm 2** Multi-agent Option Discovery

**Input**: percentile $0 \leq k \leq 100$, joint observation transitions $T$: $\{((o_1, ..., o_N), (o'_1, ..., o'_N))_{1:sizeof(T)}\}$;
**Output**: a set of multi-agent options $\Omega$;
**Create** an empty set of options $\Omega$
**Divide** the $N$ agents into sub-groups using Algorithm 3
**for** every sub-group $G$ **do**
  **Define** the agents in $G$ as $\{g_1, ..., g_{sizeof(G)}\}$
  **Extract** the set of transitions for $G$ from $T$ as $T_G \leftarrow \{(\underbrace{(o_{g_1}, ..., o_{g_{sizeof(G)}})}_{\vec{o_G}}, \underbrace{(o'_{g_1}, ..., o'_{g_{sizeof(G)}})}_{\vec{o_G}'})_{1:sizeof(T)}\}$
  **Learn** the connectivity function of $G$'s joint observation space $f_G$ by minimizing $L(f_G)$ (Equation (2))
  **Define** $\beta'$ as the $k$-th percentile value of $f_G$ in $T_G$
  **Acquire** the termination condition of option $\omega_G$:

$$\beta_G(\vec{o_G}) \leftarrow \begin{cases} 1 & if \ f_G(\vec{o_G}) < \beta' \\ 0 & otherwise \end{cases}$$

  **Acquire** the initiation set of option $\omega_G$: $I_G \leftarrow \{\vec{o_G} | \beta_G(\vec{o_G}) == 0\}$
  **Learn** the policy $\pi_G$ of option $\omega_G$ by maximizing the accumulated reward defined as Equation (3) using **H-MSAC** based on $T_G$
  **Add** option $\omega_G$: $< I_G, \pi_G, \beta_G >$ to $\Omega$
**end for**

$$L(f_G) = \frac{1}{2}\mathbb{E}_{(\vec{o},\vec{o'})\sim T_G}[(f_G(\vec{o}) - f_G(\vec{o'}))^2] + \eta\mathbb{E}_{\vec{o}\sim\rho,\vec{o'}\sim\rho} [(f_G(\vec{o}) - 1)(f_G(\vec{o'}) - 1) + f_G(\vec{o})^2 f_G(\vec{o'})^2] \quad (2)$$

As mentioned in Algorithm 2, the reward function for learning the intra-option policy is based on $f_G$ which represents the connectivity of the joint observation space and is task-independent. Through this intrinsic reward term, agents are encouraged to explore their joint observation space. Further, inspired by [7], we can take advantage of the extrinsic reward that is specific to the environment to guarantee that the skills/options learned are also useful for team performance, so the reward at step $t$ for learning the intra-option policy can be defined as:

$$(r_t^i)_{option} = (r_t^i)_{env} + \eta[f_{G_i}(o_t^{G_i}) - f_{G_i}(o_{t+1}^{G_i})] \quad (3)$$

where $(r_t^i)_{env}$ is task-related, $\eta$ is the weight for the intrinsic term, and $G_i$ is the sub-group that agent $i$ belongs to.

### *B. Network Structure*

In real-life multi-agent systems, the agents can usually be divided into some sub-groups and each sub-group is responsible for a sub-task (option). In this paper, we adopt the widely-used soft attention mechanism [8] to get the interaction relationship among the agents from which we can abstract the sub-groups. Through this division, we can also avoid the joint observation

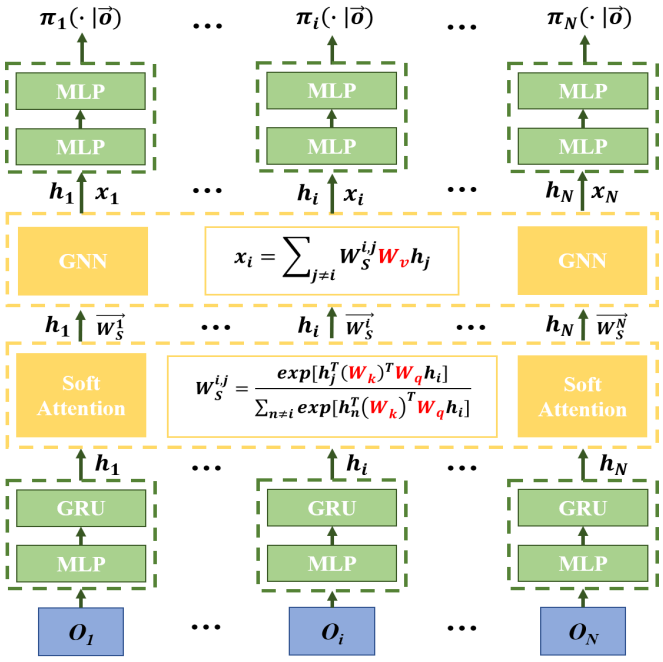



Fig. 2: Policy Network

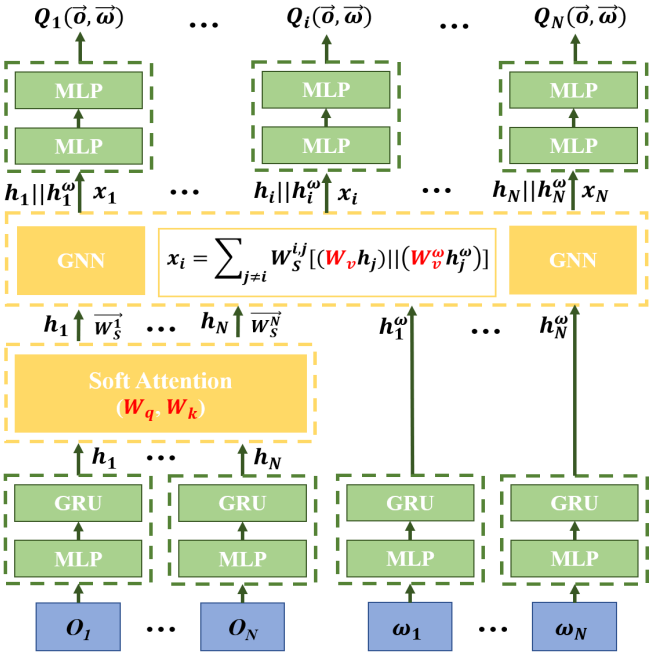



Fig. 3: Q-function Network

space growing too large, so as to learn the multi-agent options more efficiently.

The network structure of the high-level policy is shown in Figure 2. When deciding on which multi-agent option is available or to execute, agent $i$ needs to know other agents' observation to see whether they have reached the initiation set, so the communication among the agents is required. Further, we view this multi-agent system as a fully-connected directional graph, where each node represents an agent and the weight of edge $i \to j$ represents the importance of agent $j$ to agent $i$. These weights can be learnt through a soft attention mechanism (a query-key system). After that, we adopt GNN (e.g., weighted summation) to extract $x_i$, the contribution from other agents to agent $i$, and then concatenate it with its own observation embedding $h_i$ as the input of its own policy head $\pi_i$. In this way, $\pi_i$ is based on observations of all the agents, and the attention mechanism filters out observations from the agents that have no/low interaction with agent $i$, which can reduce the communication cost and ensure the scalability of the system.

**Algorithm 3** Sub-group Division

**Input**: threshold $z \in (0, 1)$, a set of joint observations $T$: $\{(o_1, ..., o_N)_{1:sizeof(T)}\}$, policy network $\pi$;
**Output**: a set of sub-groups;
**for** $t = 1$ to $sizeof(T)$ **do**
  **Feed** $(o_1, ..., o_N)_t$ into $\pi$
  **Save** the soft attention weights $(W_S^{i,j})_t$, where $i \in [1, N]$ and $j \neq i$
**end for**
**for** each agent $i$ **do**
  **Insert** $i$ to its sub-group $G_i$
  **for** each agent $j \neq i$ **do**
    **if** $\sum_t (W_S^{i,j})_t \geq z * \sum_{n \neq i} \sum_t (W_S^{i,n})_t$ **then**
      **Insert** $j$ to $G_i$
    **end if**
  **end for**
**end for**
**Collect** $G \leftarrow \{G_1, ..., G_N\}$ and eliminate duplicate elements
**Return** $G$

The Q-value function network is shown as Figure 3. We only use the observations $o_{1:N}$ to extract the interaction relationship rather than the observation-option pairs $(o, \omega)_{1:N}$ (like in [19], [20]), because: (1) the interaction relationship extracted by the policy and Q-function network should be the same; (2) parameters of the attention mechanism can be shared between the two networks, which can improve the training efficiency. Also, note that the parameters of the soft attention and GNN part: $W_k, W_q, W_v, W_v^\omega$ are shared by all the agents.

The structure of the networks for learning the intra-option policy $\pi_\omega$ is the same as the ones for learning the high-level policy. As mentioned above, the primitive actions can be viewed as one-step options and the policy network for selecting among the primitive actions can be used for the sub-group division. As shown in Algorithm 3, we use the transitions collected in the training process as input and calculate the accumulated soft attention weight matrix based on the output of the attention layer. Then we compare the normalized results with a predefined threshold $z$ to finalize the sub-group for each agent. ($z$ is set as 0.6.)

### *C. Hierarchical Multi-agent Reinforcement Learning based on Soft Actor-Critic*

In this section, we extend soft actor-critic (SAC) [9], a widely-used reinforcement learning algorithm, to a hierarchical

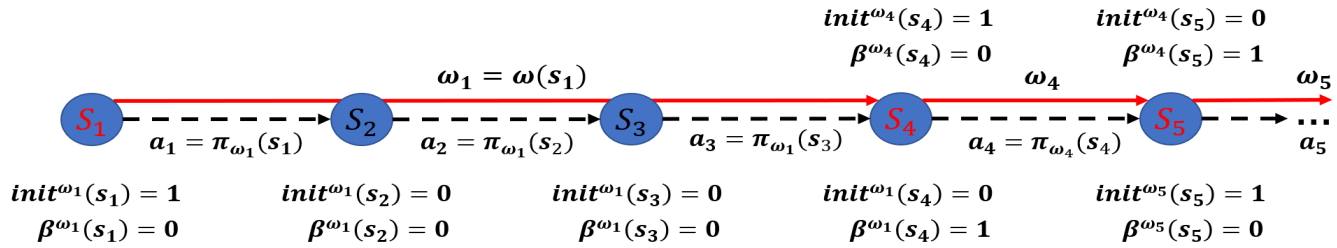



Fig. 4: Single-agent Scenario: The agent chooses the option $\omega_1$ at $s_1$ ($init^{\omega_1}(s_1) == 1$), then the corresponding intra-option policy $\pi_{\omega_1}$ is executed until $s_4$ ($\beta^{\omega_1}(s_4) == 1$). Meanwhile, a new option $\omega_4$ is chosen at $s_4$ ($init^{\omega_4}(s_4) == 1$). Note that the high-level policy is executed at state $s$ only when the last option $\omega$ ends at this step ($\beta^{\omega}(s) == 1$).

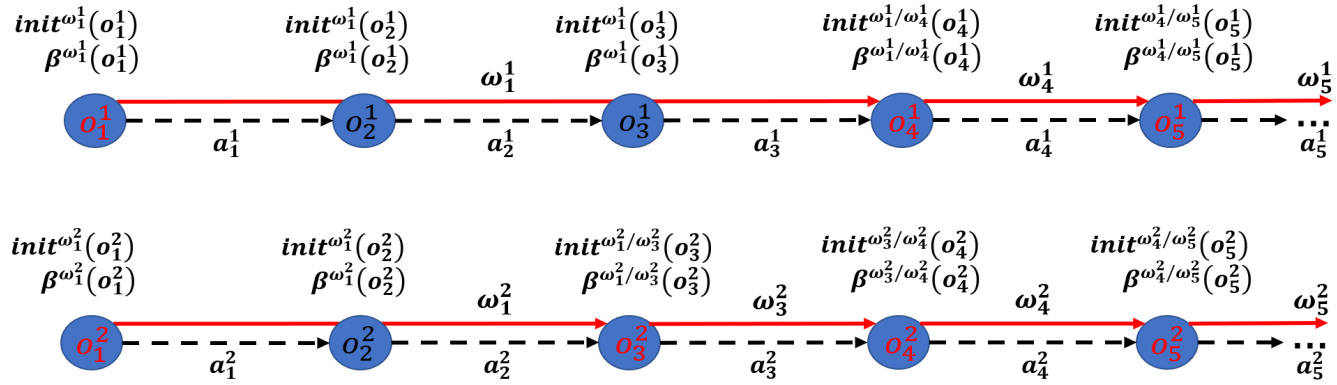



Fig. 5: Multi-agent Scenario: Similar with the single-agent scenario, $init^{\omega_t^i}(o_t^i) == 1$, if agent $i$ starts to execute the option $\omega_t^i$ when getting the observation $o_t^i$ at step $t$; $\beta^{\omega_t^i}(o_{t'}^i) == 1$, if the option $\omega_t^i$ ends at step $t'$. However, the option choice among the agents maybe asynchronous, which brings new difficulty as compared with the single-agent scenario. For example, agent 2 starts a new option at step 3, while agent 1 is still within $\omega_1^1$, so the two agents' high-level policy are not executed at the same time. Note that when updating the high-level policy of agent $i$, the gradient calculated based on the transition at step $t$ can be applied only if $init^{\omega_t^i}(o_t^i) == 1$, which means the high-level policy of agent $i$ is executed at this step.

multi-agent reinforcement learning scenario with attentive actors and critics, which we name as **H-MSAC**. This algorithm can be used to update the high/low-level policy and corresponding Q-function network in an off-policy manner.

The authors of [20] extended SAC to a multi-agent scenario with attentive critics and define the objective functions as follows:

$$L_Q(\phi) = \sum_{i=1}^{N} \mathbb{E}_{(o,a,r,o')\sim B}[(Q_i^{\phi}(o,a) - y_i)^2] \tag{4}$$

$$y_i = r_i + \gamma \mathbb{E}_{a'\sim\pi_{\overline{\theta}}(o')}[Q_i^{\overline{\phi}}(o',a') - \alpha log(\pi_{\overline{\theta}_i}(a_i'|o_i'))] \tag{5}$$

$$\begin{aligned}\nabla_{\theta_i} J(\pi_{\theta_i}) = \mathbb{E}_{o\sim B, a\sim\pi_\theta(o)}[\nabla_{\theta_i} log(\pi_{\theta_i}(a_i|o_i)) \\ (-\alpha log(\pi_{\theta_i}(a_i|o_i)) + Q_i^{\phi}(o,a) - b(o,a_{\setminus i}))]\end{aligned} \tag{6}$$

$$\begin{aligned}b(o,a_{\setminus i}) &= \mathbb{E}_{a_i\sim\pi_{\theta_i}(o_i)}[Q_i^{\phi}(o,(a_i,a_{\setminus i}))] \\ &= \sum_{a_i\in A_i} \pi_{\theta_i}(a_i|o_i) Q_i^{\phi}(o,(a_i,a_{\setminus i}))\end{aligned} \tag{7}$$

where $B$ represents the replay buffer, $\overline{\theta}$ and $\overline{\phi}$ are the parameters of the target policy and Q-function network, $\alpha log(\pi_\theta)$ is the entropy term. $b$ is the baseline term of the advantage function, and in the case of discrete policies, it can be calculated explicitly (Equation (7)). However, we have to estimate the expectation in Equation (5) and (6) through sampling. For example, in Equation (5), we can't enumerate all the possible joint actions $a'$, so we calculate $a_i' \sim \pi_{\overline{\theta}_i}(o_i'),\ for\ i = 1\ to\ N$, and then obtain the sampled joint action $a'$.

In our setting, we use attentive actors, which means that each agent's policy takes observations of all the agents as input, so we modify the calculation of $y_i$, $\nabla_\theta J(\pi_\theta)$, and $b(o,a_{\setminus i})$ as follows:

$$y_i = r_i + \gamma \mathbb{E}_{a'\sim\pi_{\overline{\theta}}(o')}[Q_i^{\overline{\phi}}(o',a') - \alpha log(\pi_{\overline{\theta}_i}(a_i'|o'))] \tag{8}$$

$$\begin{aligned}\nabla_\theta J(\pi_\theta) = \sum_{i=1}^{N} \mathbb{E}_{o\sim B, a\sim\pi_\theta(o)}[\nabla_{\theta_i} log(\pi_{\theta_i}(a_i|o)) \\ (-\alpha log(\pi_{\theta_i}(a_i|o)) + Q_i^{\phi}(o,a) - b(o,a_{\setminus i}))]\end{aligned} \tag{9}$$

$$\begin{aligned}b(o,a_{\setminus i}) &= \mathbb{E}_{a_i\sim\pi_{\theta_i}(o)}[Q_i^{\phi}(o,(a_i,a_{\setminus i}))] \\ &= \sum_{a_i\in A_i} \pi_{\theta_i}(a_i|o) Q_i^{\phi}(o,(a_i,a_{\setminus i}))\end{aligned} \tag{10}$$

Note that when estimating the expectation in Equation (8) and (9), we don't need any extra sampling process as compared with Equation (5) and (6).

As mentioned above, the learning process is based on the extended action set which includes primitive actions and options. As shown in Figure 4, the primitive actions are viewed as one-step options (e.g., $\omega_4$), and once an option is chosen by the high-level policy, the corresponding intra-option policy will be executed until its termination condition is satisfied. For example, in Figure 4, the agent chooses the option $\omega_1$ at $s_1$ ($init^{\omega_1}(s_1) == 1$), then the intra-option policy $\pi^{\omega_1}$ is executed until $s_4$ ($\beta^{\omega_1}(s_4) == 1$). Next, we describe the objective functions for learning with options. We transfer the option-critic framework [5] from a Q-learning setting to SAC, and extend it from the single-agent scenario to the multi-agent scenario.

As shown in Figure 4, during the sampling process,we can collect transitions like $(s_t,\ \omega_t,\ init_t^{\omega_t},\ a_t,\ r_t,\ s_{t+1},\ \beta_{t+1}^{\omega_t})$, where $\omega_t$ is the option choice at step $t$, $init^{\omega_t}$ and $\beta^{\omega_t}$ are the corresponding initiation and termination signal. Based on them, we can define the objective functions for the single agent scenario as follows:

$$L_Q(\phi) = \mathbb{E}_{(s_t,\omega_t,r_t,s_{t+1},\beta_{t+1}^{\omega_t})\sim B}[(Q^{\phi}(s_t,\omega_t) - y_t)^2] \quad (11)$$

$$\begin{aligned} y_t = r_t + \gamma[&(1-\beta_{t+1}^{\omega_t})Q^{\overline{\phi}}(s_{t+1},\omega_t)+ \\ &\beta_{t+1}^{\omega_t}\mathbb{E}_{\omega_{t+1}\sim\pi_{\overline{\theta}}(s_{t+1})}(Q^{\overline{\phi}}(s_{t+1},\omega_{t+1}) - \alpha log(\pi_{\overline{\theta}}(\omega_{t+1}|s_{t+1})))] \end{aligned} \quad (12)$$

In this case, if $\omega_t$ terminates at $s_{t+1}$ ($\beta_{t+1}^{\omega_t} == 1$), the agent will choose a new option based on the target high-level policy $\pi_{\overline{\theta}}$; otherwise, it will still execute $\omega_t$ (i.e., $\omega_{t+1} == \omega_t$). The objective function for updating the policy network is given as:

$$\begin{aligned} &\nabla_\theta J(\pi_\theta) = \\ &\mathbb{E}_{(s_t,init_t)\sim B,\omega_t\sim\pi_\theta(s_t)}[init_t * \nabla_\theta log(\pi_\theta(\omega_t|s_t)) \\ &(-\alpha log(\pi_\theta(\omega_t|s_t)) + Q^{\phi}(s_t,\omega_t) - b(s_t))] \end{aligned} \quad (13)$$

The intuition behind this equation is that $init_t$ marks the steps where the high-level policy $\pi_\theta$ is executed ($init_t == 1$) and thus the computed gradient is effective for update.

As shown in Figure 5, when extended to the multi-agent scenario ($N$ agents), we can collect transitions like: $(o_t,\ \omega_t,\ init_t,\ a_t,\ r_t,\ o_{t+1},\ \beta_{t+1}) = [(o_t^1,...,o_t^N),\ (\omega_t^1,...,\omega_t^N),\ (init_t^{\omega_t^1},...,init_t^{\omega_t^N}),\ (a_t^1,...,a_t^N),\ (r_t^1,...,r_t^N),\ (o_{t+1}^1,...,o_{t+1}^N),\ (\beta_{t+1}^{\omega_t^1},...,\beta_{t+1}^{\omega_t^N})]$. Note that the option choice among the agents maybe asynchronous and this brings new difficulty. For example, in Figure 5, agent 2 starts a new option at step 3, while agent 1 is still within $\omega_1^1$, so we can't update the two agents' high-level policy at the same time. Next, we define its objective functions as follows:

$$L_Q(\phi) = \sum_{i=1}^{N}\mathbb{E}_{(o_t,\omega_t,r_t,o_{t+1},\beta_{t+1})\sim B}[(Q_i^{\phi}(o_t,\omega_t) - y_t^i)^2] \quad (14)$$

$$\begin{aligned} y_t^i = r_t^i + \gamma\mathbb{E}_{\omega_{t+1}\sim\pi_{\overline{\theta}}(o_{t+1})}[&Q_i^{\overline{\phi}}(o_{t+1},\omega_{t+1}) \\ &-\alpha\beta_{t+1}^{\omega_t^i} log(\pi_{\overline{\theta}_i}(\omega_{t+1}^i|o_{t+1}))] \end{aligned} \quad (15)$$

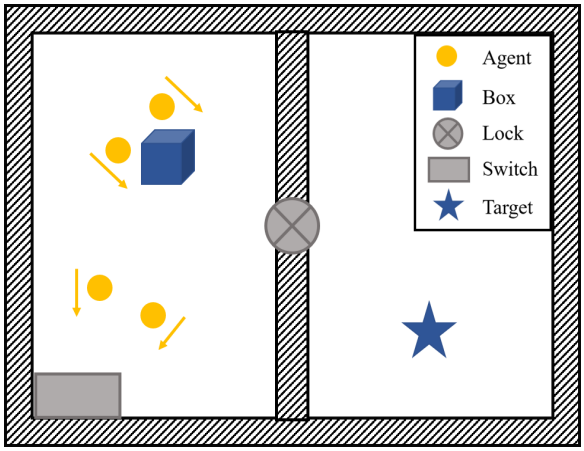



Fig. 6: Task Scenario for Evaluation

When estimating the expectation in Equation (15), we need to sample the joint option $\omega_{t+1}$, which is generated in this way: for $i = 1$ to $N$: if $\beta_{t+1}^{\omega_t^i} == 1$, $\omega_{t+1}^i \sim \pi_{\overline{\theta}_i}(o_{t+1})$; otherwise, $\omega_{t+1}^i = \omega_t^i$. The intuition is that new option isn't required until last option terminates.

$$\begin{aligned} \nabla_\theta J(\pi_\theta) = \sum_{i=1}^{N}\mathbb{E}_{(o_t,\omega_t,init_t)\sim B,\hat{\omega}_t\sim\pi_\theta(o_t)}[&init_t^i * \\ &\nabla_{\theta_i} log(\pi_{\theta_i}(\hat{\omega}_t^i|o_t))(-\alpha log(\pi_{\theta_i}(\hat{\omega}_t^i|o_t))+ \\ &Q_i^{\phi}(o_t,\hat{\omega}_t) - b(o_t,\hat{\omega}_t^{\setminus i}))] \end{aligned} \quad (16)$$

$$b(o_t,\hat{\omega}_t^{\setminus i}) = \mathbb{E}_{\hat{\omega}_t^i\sim\pi_{\theta_i}(o_t)}[Q_i^{\phi}(o_t,(\hat{\omega}_t^i,\hat{\omega}_t^{\setminus i}))] \quad (17)$$

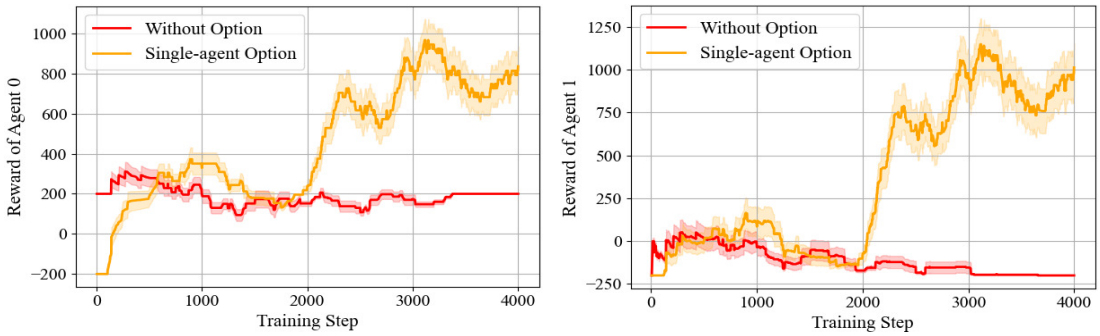
where the expectation in Equation (17) is calculated in the same way as Equation (10), while the expectation in Equation (16) is estimated by sampling $\hat{\omega}_t$: for $i = 1$ to $N$: if $init_t^i == 1$, $\hat{\omega}_t^i \sim \pi_{\theta_i}(o_t)$; otherwise, $\hat{\omega}_t^i = \omega_t^i$. The intuition behind this is similar with that of Equation (13), that is we need to use $init_t^i$ to mask out the steps where $\pi_{\theta_i}$ is not executed. In conclusion, the objective functions of **H-MSAC** are listed as Equation (14)-(17).

## V. Evaluation

Generally, a multi-agent task scenario can be divided into some sub-tasks and each sub-task can be completed by a sub-group of the agents, and meanwhile, the sub-groups need to coordinate with each other very well to get the whole task done. In our algorithm, under a hierarchical MARL framework, we learn the optimal grouping through the attention mechanism, learn the sub-skill of each sub-group through the multi-agent option discovery and learn the coordination among the sub-groups through the high-level policy learning. In this section, we will test the effectiveness of our algorithm on a self-designed simulator which is an extension of the commonly-used benchmark – Push-Box [21].

### A. Simulator Setup

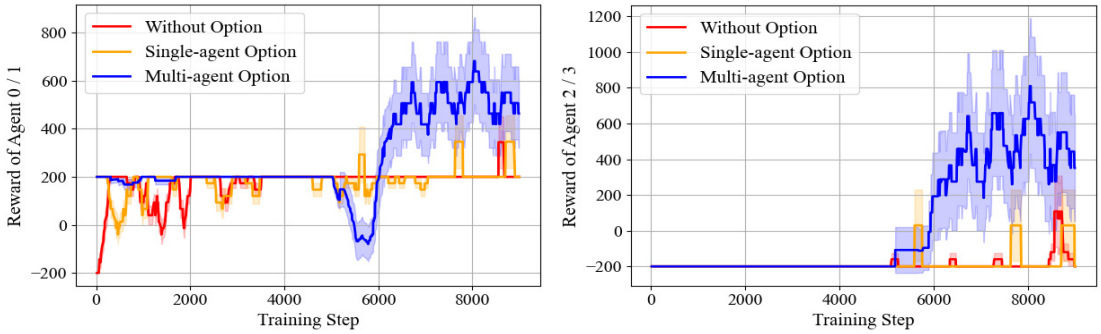
As shown in Figure 6, there are four agents in the environment and their task is to push the box to the target. During the




(a) Training reward of agent 0

(b) Training reward of agent 1

Fig. 7: Sub-setting 1.




(a) Training reward of agent 0 / 1

(b) Training reward of agent 2 / 3

Fig. 8: Sub-setting 2.

pushing process, they have to pass through a wall with a lock which is controlled by the switch, so they have to turn on the switch first (the wall in the middle will open), and then push the box to the target. Note that only when two or more agents contact the switch at the same time will the switch open, and the box won't move until two or more agents push it in the same direction at the same time. In this case, the whole task can be divided into two sub-tasks, and coordination among the agents is required for each sub-task.

We compare our algorithm with two baselines: (1) a multi-agent reinforcement learning algorithm with attention mechanism but without option discovery, such as MAAC [20] with the same network design as ours; (2) a multi-agent reinforcement learning algorithm with single-agent option discovery, where the option discovery algorithm is introduced in [4] and apparently, the coordination among the agents is not considered during the option discovery process.

The evaluation is based on the three sub-settings: (1) In this sub-setting, we simplify the task scenario to a two-agent version: agent 0 is able to turn on the switch, agent 1 is able to push the box, and both sub-tasks require only one agent. We will test the two baseline algorithms on this sub-setting and show the effectiveness of the single-agent option discovery. (2) Only two of the agents (agent 0 and 1) are able to turn on the switch and the other two agents (agent 2 and 3) are able to push the box. We will see whether the grouping (group 0: agent 1 and 2, group 1: agent 3 and 4) can be found through the attention mechanism, and whether the sub-tasks (i.e., group 0: turning on the switch, group 1: pushing the box to the target) can be learned through our algorithm. (3) We only have three agents in this sub-setting: agent 0 is only able to turn on the switch, agent 2 is only able to push the box, while agent 1 can do both. In this case, agent 1 should first group with agent 0 to turn on the switch and then group with agent 2 to push the box to the target. For agent 1, the grouping is dynamic, which brings more challenges to our task.

The reward function for the three sub-settings are the same, which is defined as:

$$r^i_{env} = -c_0 + c_1 e^i_{switch} + c_2 e^i_{move} + c_3 e_{target} \tag{18}$$

where $c_{0:3} > 0$ are the weights, $e^i_{move}$ represents whether the box is moved for one step by agent $i$, $e^i_{switch}$ represents whether the switch is turned on by agent $i$, and $e_{target}$ represents whether the box is pushed to the target area successfully (this term is shared by all the agents). Besides, the whole task is required to be completed within 400 steps.

### *B. Results and Discussion*

In this section, we compare MARL with the single-agent option, multi-agent option, and no option on the three sub-settings introduced above.

In the sub-setting 1, there are only two agents and each of them is responsible for a different task, i.e., agent 0: turn on the switch, agent 1: push the box. Figure 7(a)-7(b) plot the

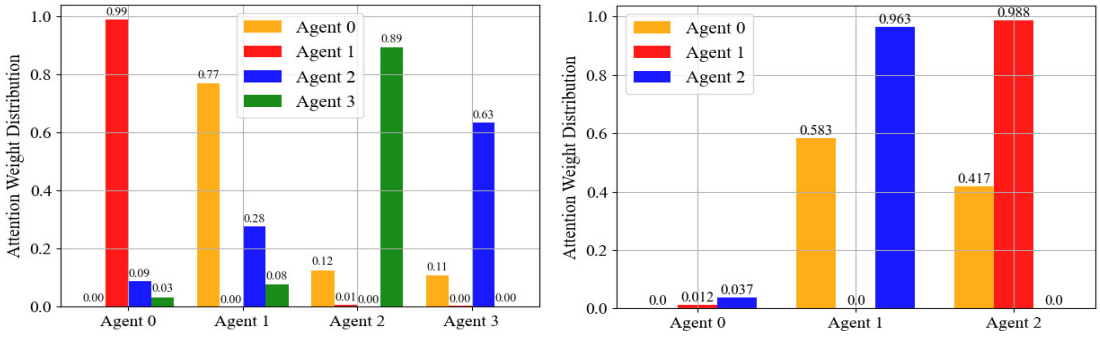
reward function of agent 0 and 1 in the training process. It can be observed that the agent without option can learn how to turn on the switch and get the corresponding reward term, however, it can't learn the harder task – pushing the box to the target, given the reward function is sparse. Then, we train a single-agent option for each agent at step 2000 with the transitions collected at the first 2000 steps, after which we start to train the high-level policy and low-level policy together within a hierarchical RL framework introduced above. The performance starts to improve at step 2000, which shows the effectiveness of MARL with the single-agent option. Agent 1 learns its sub-task through a denser reward function (Equation (3)), and also agent 0 and 1 learns how to cooperate with each other (turn on the switch first and then push the box through the gate to the target) through the training of the high-level policy.

In the sub-setting 2, agent 0 and 1 need to contact the switch at the same time to turn on it, and agent 2 and 3 need to push the box in the same direction to make it move, so collaboration is required for each sub-task, which is different from the sub-setting 1. Figure 8(a)-8(b) show that the agent without option or with four single-agent options (one for each agent) can learn to open the switch, while it can't get how to push the box to the target very well. Then, based on the transitions collected at the first 5000 steps, we can get the attention weight distribution of the agents. As shown in Figure 9(a), the attention weight that agent 0 pays to agent 1 is higher than the threshold 0.6 (Algorithm 3) and vice versa, so we can get a sub-group: [0, 1]; similarly, we can get the other sub-group: [2, 3]. Then, we can train the multi-agent option for each sub-group to complete its corresponding sub-task, and integrate the options in the whole learning process as shown in Algorithm 1. Results show that the agent with multi-agent options performs better, since it considers the collaboration among the agents in the option discovery process while the agent with single-agent options doesn't.

Different from the sub-setting 2, in the sub-setting 3, agent 1 should cooperate with agent 0 first to open the switch, and then push the box to the target area together with agent 2, which makes the task even harder. Results in Figure 10 show that agent 1 without options or with single-agent options (grouping: [0], [1], [2]) tend to be stuck by the local optimum, after it completes the easier task – open the switch. While, multi-agent options (grouping results shown as Figure 9(b): [0], [1, 2]) can avoid this, since the adoption of the multi-agent option for group [1, 2] can encourage the exploration of the joint state space of agent 1 and 2, and improve the overall performance. Note that the grouping results and multi-agent option discovery are based on the transitions collected at the first 5000 steps.

## VI. Conclusion

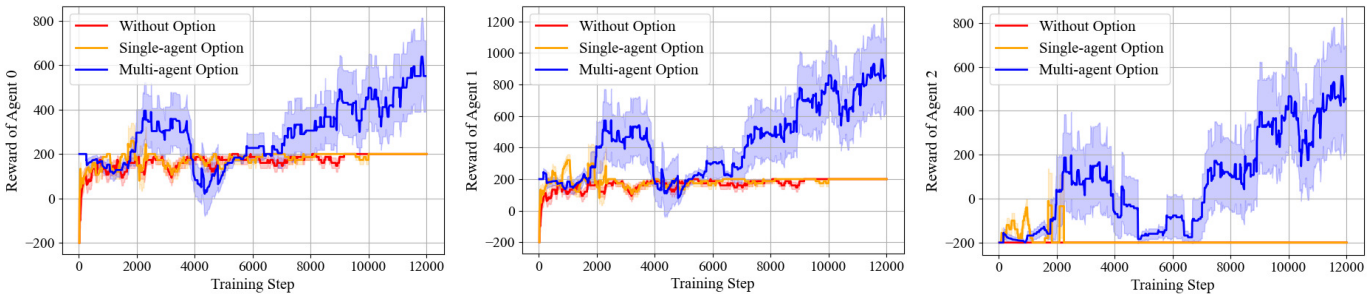
This paper proposes *Multi-agent Deep Covering Option Discovery* and a hierarchical multi-agent reinforcement learning algorithm based on soft actor-critic to integrate the options in a MARL setting – **H-MSAC**. This approach first divides all the agents into some sub-groups through the widely-used attention mechanism based on their interaction relationship, and then learns the multi-agent options for each sub-group to encourage the joint exploration of the multiple agents in a sub-group. Evaluation results on a self-designed simulator show reasonable sub-group division results with the attention mechanism, and superior performance of the MARL agents with multi-agent options as compared to the ones with single-agent options or no options.




(a) Attention weights of sub-setting 2

(b) Attention weights of sub-setting 3

Fig. 9: Attention weights of sub-setting 2 and 3.




(a) Training reward of agent 0

(b) Training reward of agent 1

(c) Training reward of agent 2

Fig. 10: Sub-setting 3.

## REFERENCES


[1] R. S. Sutton, D. Precup, and S. P. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artif. Intell.*, vol. 112, no. 1-2, pp. 181–211, 1999.

[2] Y. Jinnai, J. W. Park, D. Abel, and G. D. Konidaris, "Discovering options for exploration by minimizing cover time," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, vol. 97. PMLR, 2019, pp. 3130–3139.

[3] A. Ghosh and S. P. Boyd, "Growing well-connected graphs," in *45th IEEE Conference on Decision and Control, CDC 2006*. IEEE, 2006, pp. 6605–6611.

[4] Y. Jinnai, J. W. Park, M. C. Machado, and G. D. Konidaris, "Exploration in reinforcement learning with deep covering options," in *8th International Conference on Learning Representations, ICLR 2020*. OpenReview.net, 2020.

[5] J. Chakravorty, P. N. Ward, J. Roy, M. Chevalier-Boisvert, S. Basu, A. Lupu, and D. Precup, "Option-critic in cooperative multi-agent systems," in *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20*, 2020, pp. 1792–1794.

[6] Y. Lee, J. Yang, and J. J. Lim, "Learning to coordinate manipulation skills via skill behavior diversification," in *8th International Conference on Learning Representations, ICLR 2020*. OpenReview.net, 2020.

[7] J. Yang, I. Borovikov, and H. Zha, "Hierarchical cooperative multi-agent reinforcement learning with skill discovery," in *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20*, 2020, pp. 1566–1574.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, 2017, pp. 5998–6008.

[9] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, vol. 80. PMLR, 2018, pp. 1856–1865.

[10] I. Menache, S. Mannor, and N. Shimkin, "Q-cut—dynamic discovery of sub-goals in reinforcement learning," in *European Conference on Machine Learning*. Springer, 2002, pp. 295–306.

[11] G. Konidaris and A. Barto, "Skill discovery in continuous reinforcement learning domains using skill chaining," *Advances in neural information processing systems*, vol. 22, pp. 1015–1023, 2009.

[12] J. Harb, P.-L. Bacon, M. Klissarov, and D. Precup, "When waiting is not an option: Learning options with a deliberation cost," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[13] S. Tiwari and P. S. Thomas, "Natural option critic," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 5175–5182.

[14] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine, "Diversity is all you need: Learning skills without a reward function," in *7th International Conference on Learning Representations, ICLR 2019*. OpenReview.net, 2019.

[15] M. C. Machado, M. G. Bellemare, and M. H. Bowling, "A laplacian framework for option discovery in reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*, vol. 70. PMLR, 2017, pp. 2295–2304.

[16] C. Amato, G. D. Konidaris, and L. P. Kaelbling, "Planning with macro-actions in decentralized pomdps," in *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14*. IFAAMAS/ACM, 2014, pp. 1273–1280.

[17] C. Amato, G. Konidaris, L. P. Kaelbling, and J. P. How, "Modeling and planning with macro-actions in decentralized pomdps," *Journal of Artificial Intelligence Research*, vol. 64, pp. 817–859, 2019.

[18] J. Shen, G. Gu, and H. Liu, "Multi-agent hierarchical reinforcement learning by integrating options into maxq," in *First international multi-symposiums on computer and computational sciences (IMSCCS'06)*, vol. 1. IEEE, 2006, pp. 676–682.

[19] Y. Liu, W. Wang, Y. Hu, J. Hao, X. Chen, and Y. Gao, "Multi-agent game abstraction via graph attention neural network," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*. AAAI Press, 2020, pp. 7211–7218.

[20] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, vol. 97. PMLR, 2019, pp. 2961–2970.

[21] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Neural Information Processing Systems (NIPS)*, 2017.