Multi-agent Covering Option Discovery through Kronecker Product of Factor Graphs

Jiayu Chen
Purdue University
West Lafayette, United States
chen3686@purdue.edu

Jingdi Chen
The George Washington University
Washington, DC, United States
jingdic@gwu.edu

Tian Lan
The George Washington University
Washington, DC, United States
tlan@gwu.edu

Vaneet Aggarwal
Purdue University
West Lafayette, United States
vaneet@purdue.edu

ABSTRACT

Covering option discovery has been developed to improve the exploration of reinforcement learning in single-agent scenarios with sparse reward signals, through connecting the most distant states in the embedding space provided by the Fiedler vector of the state transition graph. However, due to exponentially-large state space resulted from multi-agent scenarios, existing researches on adopting options in multi-agent scenarios still rely on single-agent option discovery and fail to directly discover the joint options that can improve the connectivity of the joint state space of agents. In this paper, we show that it is indeed possible to directly compute multi-agent options with collaborative exploratory behavior while still enjoying the ease of decomposition. Our key idea is to approximate the joint state space as a kronecker graph – the kronecker product of individual agents' state transition graphs, based on which we can directly estimate the Fiedler vector of the joint state space using the Laplacian spectrum of individual agents' transition graphs. This decomposition enables us to efficiently construct multi-agent joint options by encouraging agents to connect the sub-goal joint states which are corresponding to the minimum or maximum values of the estimated joint Fiedler vector. The evaluation based on multi-agent collaborative tasks shows that the proposed algorithm can successfully identify multi-agent options, and significantly outperforms prior works using single-agent options or no options, in terms of both faster exploration and higher cumulative rewards.

ACM Reference Format:
Jiayu Chen, Jingdi Chen, Tian Lan, and Vaneet Aggarwal. 2022. Multi-agent Covering Option Discovery through Kronecker Product of Factor Graphs. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Auckland, New Zealand, May 9–13, 2022,* IFAAMAS, 15 pages.

1 INTRODUCTION

Option discovery [22] enables temporally-abstract actions to be constructed and utilized in the reinforcement learning (RL) process, which can significantly improve the performance of RL agents. Among recent developments on the topic, *Covering Option Discovery* [8, 9] has been shown to be an effective approach to improve the exploration in environments with sparse reward signals. In particular, it first computes the second smallest eigenvalue and the corresponding eigenvector (i.e., Fiedler vector [5]) of the Laplacian matrix extracted from the state transition process in RL. Then, options are built to connect the states corresponding to the minimum or maximum values in the Fiedler vector, which has been proved to greedily improve the algebraic connectivity of the state space [6] and thus accelerate exploration in the state space.

In this paper, we consider the problem of constructing and utilizing covering options in multi-agent reinforcement learning (MARL). Due to the exponentially-large state space in multi-agent scenarios, a commonly-adopted way to solve this problem [1, 2, 4, 11, 17] is to construct the single-agent options as if in a single-agent environment first, and then learn to collectively leverage these individual options to tackle multi-agent tasks. This method fails to consider the coordination among agents in the option discovery process, and thus can suffer from very poor behavior in multi-agent collaborative tasks. To this end, in our work, we propose a framework that makes novel use of kronecker product of factor graphs to directly construct the multi-agent options in the joint state space, and adopt them to accelerate the joint exploration of agents in MARL. We show through experiments that agents leveraging our multi-agent options significantly outperform agents with single-agent options or no options in MARL tasks. For some challenging tasks, the adoption of multi-agent options can improve the convergence speed by two orders of magnitude and the episodic cumulative reward by $\sim 100\%$. Also, instead of directly adopting the covering option discovery to the joint state space since its size grows exponentially with the number of agents, we build multi-agent options based on the individual state transition graphs, making our method much more scalable.

Specifically, the main contributions are as follows: (1) We propose *Multi-agent Covering Option Discovery* – it approximates the joint state transition graph as a kronecker product of the individual ones, so that we can estimate the Fiedler vector of the joint state space based on the Laplacian spectrum of the individual state spaces to enjoy the ease of decomposition. Then, the joint options composed of multiple agents' temporal action sequences can be directly constructed to connect the joint states corresponding to the minimum or maximum in the Fiedler vector, resulting in a greedy improvement of the joint state space's algebraic connectivity. (2) We propose that the multi-agent options can be adopted to MARL

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Auckland, New Zealand. © 2022 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

in either a decentralized or centralized manner. For the centralized manner, different agents jointly decide on their options. In contrast, for the decentralized manner, agents can choose their options independently and select different options to execute simultaneously. Further, we compare the decentralized or centralized approaches through experiments. (3) We propose group-based multi-agent option discovery that first groups the agents based on their sub-tasks, and then discover the multi-agent options within each sub-group. This sub-group division not only makes our approach more scalable but also makes the option discovery more accurate.

2 RELATED WORK

Option Discovery: The option framework was proposed in [22], which extends the usual notion of actions to include options — the closed-loop policies for taking actions over a period of time. In the literature, lots of option discovery algorithms have been proposed. Some of them are based on task-related reward signals, such as [7, 14–16]. Specifically, they directly define or learn through gradient descent the options that can lead the agent to the rewarding states in the environments, and then utilize these trajectory segments (options) to compose the completed trajectory toward the goal state. These methods rely on dense reward signals, which are usually hard to acquire in real-life tasks. Other works define the sub-goal states (termination states of the options) based on the visitation frequency of the states. For example, in [18–20], they discover the options by recognizing the bottleneck states in the environment, through which the agent can transfer between the sub-areas that are loosely connected in the state space, and they define these options as betweenness options. Recently, there are some state-of-the-art option generation methods based on the Laplacian spectrum of the state-transition graph, such as [8, 9, 12, 13], since the eigenvectors of the Laplacian of the state space can provide embeddings in lower-dimensional space, based on which we can obtain good measurements of the accessibility from one state to another. Especially, in [9], they propose covering options and prove that their option generation method has higher exploration speed and better performance compared with other Laplacian-based approaches and the betweenness options mentioned above.

Note that all the approaches mentioned above are for single-agent scenarios, and in this paper we will extend the construction and adoption of covering options to MARL.

Adopting options in multi-agent scenarios: Most of the researches about adopting options in MARL, such as [1, 2, 4, 11, 17], tried to first learn the options for each individual agent with the option discovery methods we mentioned above, and then learn to collaboratively utilize these individual options. Therefore, the coordination in the multi-agent system can only be shown/utilized in the option-choosing process while not the option discovery process. In this paper, we propose constructing multi-agent covering options based on the aforementioned Laplacian-based framework to encourage efficient exploration in the joint state space and explore how to utilize the multi-agent options in MARL tasks effectively.

3 BACKGROUND

3.1 Basic Conceptions and Notations

In this section, we will introduce the necessary conceptions and corresponding notations used in this paper.

Markov Decision Process (MDP): The RL problem can be described with an MDP, denoted by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the state transition function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to R^1$ is the reward function, and $\gamma \in (0, 1]$ is the discount factor.

State transition graph in an MDP: The state transitions in $\mathcal{M}$ can be modelled as a state transition graph $G = (V_G, E_G)$, where V_G is a set of vertices representing the states in $\mathcal{S}$, and E_G is a set of undirected edges representing state adjacency in $\mathcal{M}$. We note that there is an edge between state s and s' (i.e., s and s' are adjacent) if and only if $\exists\ a \in \mathcal{A},\ s.t.\ \mathcal{P}(s, a, s') > 0\ \vee\ \mathcal{P}(s', a, s) > 0$.

The adjacency matrix A of G is an $|\mathcal{S}| \times |\mathcal{S}|$ matrix whose (i, j) entry is 1 when s_i and s_j are adjacent, and 0 otherwise. The degree matrix D is a diagonal matrix whose entry (i, i) equals the number of edges incident to s_i. The Laplacian matrix of G is defined as $L = D - A$. Its second smallest eigenvalue $\lambda_2(L)$ is called the algebraic connectivity of the graph G, and the corresponding normalized eigenvector is called the Fiedler vector [5]. Last, the normalized Laplacian matrix is defined as $\mathcal{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$.

Kronecker product of graphs [24]: Let $G_1 = (V_{G_1}, E_{G_1})$ and $G_2 = (V_{G_2}, E_{G_2})$ be two state transition graphs, corresponding to the individual state space $\mathcal{S}_1$ and $\mathcal{S}_2$ respectively. The kronecker product of them denoted by $G_1 \otimes G_2$ is a graph defined on the set of vertices $V_{G_1} \times V_{G_2}$, such that: $A_1 \otimes A_2$ is an $|\mathcal{S}_1||\mathcal{S}_2| \times |\mathcal{S}_1||\mathcal{S}_2|$ matrix with elements defined by $(A_1 \otimes A_2)(I, J) = A_1(i, j)A_2(k, l)$ with Equation (1), where A_1 and A_2 are the adjacency matrices of G_1 and G_2, $A_1(i, j)$ is the element lies on the i-th row and j-th column of A_1 (indexed from 1).

$$I = (i - 1) \times |\mathcal{S}_2| + k, \quad J = (j - 1) \times |\mathcal{S}_2| + l \tag{1}$$

3.2 Covering Option Discovery

As defined in [22], an option ω consists of three components: an intra-option policy $\pi_\omega : \mathcal{S} \text{ x } \mathcal{A} \to [0, 1]$, a termination condition $\beta_\omega : \mathcal{S} \to \{0, 1\}$, 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 ω is taken, actions are selected according to π_ω until ω terminates stochastically according to β_ω. Therefore, in order to get an option, we need to train/define the intra-option policy, define the termination condition and initiation set.

The authors of [9] propose covering option discovery – discovering options by minimizing the upper bound of the expected cover time of the state space. First, they compute the Fiedler vector F of the Laplacian matrix of the state transition graph. Then, they collect the states s_i and s_j with the largest $(F_i - F_j)^2$ (F_i is the i-th element in F), based on which they construct two symmetric options: $\omega_{ij} = < I_{\omega_{ij}} = \{s_i\},\ \pi_{\omega_{ij}},\ \beta_{\omega_{ij}} = \{s_j\} >$ and $\omega_{ji} = < I_{\omega_{ji}} = \{s_j\},\ \pi_{\omega_{ji}},\ \beta_{\omega_{ji}} = \{s_i\} >$ to connect these two sub-goal states bidirectionally, where π_ω is defined as the optimal path between the initiation and termination state. This whole process is repeated until they get the required number of options.

The intuition of this method is as follows. The authors of [6] prove that $(F_i - F_j)^2$ gives the first order approximation of the increase in $\lambda_2(L)$ (i.e., algebraic connectivity) by connecting (s_i, s_j). Then, they propose a greedy heuristic to improve the algebraic connectivity of a graph: each time connecting (s_i, s_j) corresponding to the largest $(F_i - F_j)^2$. Further, in [9], they prove that the larger the algebraic connectivity of a graph is, the smaller upper bound of the expected cover time is and the easier the exploration tends to be. Therefore, applying this greedy heuristic to the state transition graph can improve the exploration in the state space.

4 PROPOSED ALGORITHM

4.1 System Model

In this paper, we consider to compute covering options in multi-agent scenarios, with n being the number of agents, $\widetilde{\mathcal{S}} = \mathcal{S}_1 \times \mathcal{S}_2 \times \cdots \times \mathcal{S}_n$ being the set of joint states, $\widetilde{\mathcal{A}} = \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_n$ being the set of joint actions. Note that $\mathcal{S}_i$ and $\mathcal{A}_i$ are the individual state space and action space of agent i. Apparently, the size of the joint state space, i.e., $|\widetilde{\mathcal{S}}| = \prod_{i=1}^{n} |\mathcal{S}_i|$, grows exponentially with the number of agents. Thus, it is prohibitive to directly compute the covering options based on the joint state transition graph using the approach introduced in Section 3.2 for a large n.

A natural method to tackle this challenging problem is to compute the options for each individual agent by considering only its own state transitions, and then learn to collaboratively leverage these individual options. However, it fails to directly recognize joint options (i.e., multi-agent options) composed of multiple agents' temporal action sequences for encouraging the joint exploration of all the agents. In this case, the algebraic connectivity of the joint state space may not be improved with these single-agent options. We will illustrate this with a simple example.

Illustrative example: Figure 1(a) shows a joint state transition graph $\widetilde{G}$ of two agents, where agent 1 has two states denoted by $\mathcal{S}_1 = \{1, 2\}$ and agent 2 has four states denoted by $\mathcal{S}_2 = \{1, 2, 3, 4\}$. In order to compute the individual options, we can restrict our attention to the state transition graph of each agent, namely G_1 and G_2, with Laplacian given by L_1 and L_2, respectively:

$$L_1 = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \; L_2 = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}. \tag{2}$$

To compute the options for each agent, we first compute the Fiedler vectors of G_1 and G_2 as:

$$v^{G_1} = \frac{1}{\sqrt{2}} \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \; v^{G_2} = \frac{1}{\sqrt{8 - 4\sqrt{2}}} \begin{bmatrix} -1 \\ -\sqrt{2}+1 \\ \sqrt{2}-1 \\ 1 \end{bmatrix}. \tag{3}$$

Then, according to the option discovery approach described in Section 3.2, we can get the individual options for agent 1 to connect its state 1 (minimum) and state 2 (maximum), and individual options for agent 2 to connect its state 1 (minimum) and state 4 (maximum). With these options, the updated joint state space would be like Figure 1(b). Apparently, the straightforward decomposition of option discovery for MARL fails to create a connected graph. It implies that for the purpose of encouraging efficient exploration, utilizing the single-agent options alone is not sufficient.

Therefore, we propose to build Multi-agent Covering Options to enhance the connectivity of the joint state space and accelerate the joint exploration of the agents within the scenario. Again, we can represent it as a tuple: $< I_\omega, \pi_\omega, \beta_\omega >$, where $I_\omega \subseteq \widetilde{\mathcal{S}}$ is the set of initiation joint states, $\beta_\omega : \widetilde{\mathcal{S}} \to \{0, 1\}$ indicates the joint states to terminate, $\pi_\omega = (\pi_\omega^1, \cdots, \pi_\omega^n)(\pi_\omega^i : \mathcal{S}_i \times \mathcal{A}_i \to [0, 1])$, is the joint intra-option policy that can lead the agents from the initiation states to the termination states. The key challenge is to calculate the Fiedler vector of the joint state space according to which we can define $< I_\omega, \pi_\omega, \beta_\omega >$ like Section 3.2. Given that $|\widetilde{\mathcal{S}}|$ grows exponentially with n, we propose to estimate the joint Fiedler vector based on individual state spaces in the next section.

4.2 Theory results

This section shows the theoretical foundations of our approach – computing multi-agent options directly based on the individual state transition graphs. The key idea is to approximate the joint state transition graph $\widetilde{G}$ as a kronecker graph $\otimes_{i=1}^{n} G_i = G_1 \otimes \cdots \otimes G_n$ and then estimate the Fiedler Vector of $\otimes_{i=1}^{n} G_i$ based on which we can build the multi-agent options.

First, we note that the kronecker product of individual state transition graphs $\otimes_{i=1}^{n} G_i$ can be used as a simple yet powerful approximation of the joint state transition graph $\widetilde{G}$ for the purpose of computing multi-agent options. This approximation becomes exact (i.e., $\widetilde{G} = \otimes_{i=1}^{n} G_i$) if the state transitions of an agent would not be influenced by the other agents. Also, we show through experiments in Section 5.2 that even if in conditions where agents can influence each other, such kronecker product approximation can still result in an effective representation of the original graph $\widetilde{G}$.

Next, we show how to effectively approximate the Fiedler vector of $\otimes_{i=1}^{n} G_i$ based on the Laplacian spectrum of the factor graphs, which enables an effective decomposition of multi-agent option discovery. Inspired by [3] which proposed an estimation of the Laplacian spectrum of the kronecker product of two factor graphs, we have the following THEOREM 4.1.

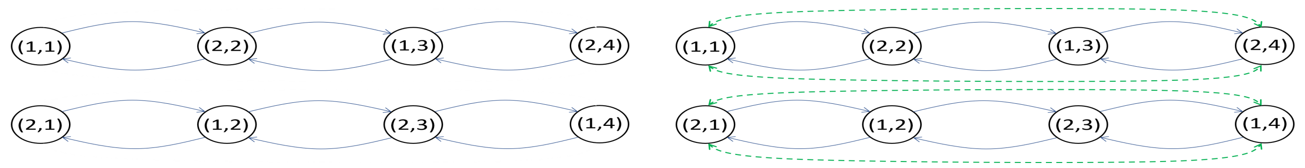
THEOREM 4.1. *For graph $\widetilde{G} = \otimes_{i=1}^{n} G_i$, we can approximate the eigenvalues μ and eigenvectors v of its Laplacian L by:*

$$\mu_{k_1,\ldots,k_n} = \left\{ \left[1 - \prod_{i=1}^{n} (1 - \lambda_{k_i}^{G_i}) \right] \prod_{i=1}^{n} d_{k_i}^{G_i} \right\} \tag{4}$$

$$v_{k_1,\ldots,k_n} = \otimes_{i=1}^{n} v_{k_i}^{G_i} \tag{5}$$

where $\lambda_{k_i}^{G_i}$ and $v_{k_i}^{G_i}$ are the k_i-th smallest eigenvalue and corresponding eigenvector of $\mathcal{L}_{G_i}$ (normalized Laplacian matrix of G_i), and $d_{k_i}^{G_i}$ is the k_i-th smallest diagonal entry of D_{G_i} (degree matrix of G_i).

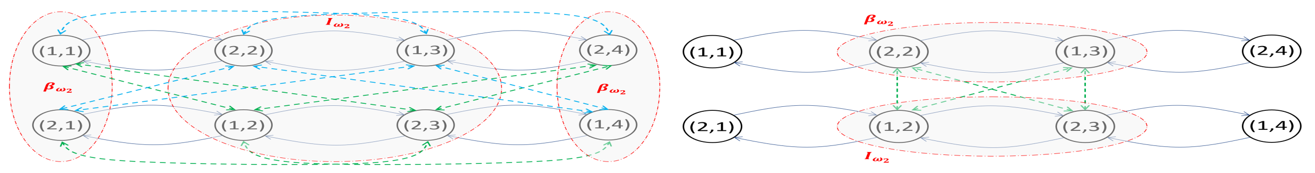
The proof of THEOREM 4.1 is provided in Appendix A.1, which also contains the proof that the estimated eigenvalues through this theorem are non-negative. Note that if G_i has K_i $(i = 1, \cdots, n)$ eigenvalues, there would be $\prod_{i=1}^{n} K_i$ eigenvalues for $\widetilde{G}$. Through enumerating $(k_1, \cdots, k_n)$, we can collect the eigenvalues of $\otimes_{i=1}^{n} G_i$ by Equation (4) and the corresponding eigenvectors by Equation (5), where the eigenvector $v_{\hat{k}_1,\cdots,\hat{k}_n}$ corresponding to the second smallest eigenvalue $\mu_{\hat{k}_1,\cdots,\hat{k}_n}$ is the estimated Fiedler vector of the joint state transition graph, namely $F_{\widetilde{G}}$. Then, we can define the joint states corresponding to the maximum or minimum in $F_{\widetilde{G}}$ as the initiation or termination joint states, which can be connected with joint options. As discussed in Section 3.2, connecting these two joint states with options can greedily improve the algebraic connectivity



(a) The joint state transition graph of agent 1 and agent 2

(b) The joint state transition graph after adding individual options

Figure 1: An illustrative example showing the limitations of utilizing single-agent options alone for MARL.



(a) The joint state transition graph updated with option ω_1

(b) The joint state transition graph updated with option ω_2

Figure 2: The joint state transition graph updated with the detected multi-agent options

of the joint state space and accelerate the joint exploration within it.

Illustrative example: Now we consider again the example in Figure 1(a), where $\widetilde{G} = G_1 \otimes G_2$. We can approximate the Fiedler vector of $\widetilde{G}$ using THEOREM 4.1. As a result, we get two approximations of the Fiedler vector (The computing details are shown in Appendix A.2):

$$v_{11} = \frac{1}{\sqrt{6}} \left[\frac{1}{\sqrt{2}},\ 1,\ 1,\ \frac{1}{\sqrt{2}},\ \frac{1}{\sqrt{2}},\ 1,\ 1,\ \frac{1}{\sqrt{2}} \right]^T \tag{6}$$

$$v_{24} = \frac{1}{\sqrt{6}} \left[-\frac{1}{\sqrt{2}},\ 1,\ -1,\ \frac{1}{\sqrt{2}},\ \frac{1}{\sqrt{2}},\ -1,\ 1,\ -\frac{1}{\sqrt{2}} \right]^T \tag{7}$$

Based on the two approximations and the indexing relationship between $\widetilde{G}$ and its factor graphs (Equation (1)), we can get two sets of multi-agent options: $\{I_{\omega_1} = \{(1,2), (1,3), (2,2), (2,3)\}$, $\beta_{\omega_1} = \{(1,1), (1,4), (2,1), (2,4)\}\}$ and $\{I_{\omega_2} = \{(1,2), (2,3)\}, \beta_{\omega_2} = \{(1,3), (2,2)\}\}$, where we set the joint states corresponding to the maximum and minimum as the initiation states and termination states respectively. For example, in v_{11} (6), the 7-th element (indexed from 1) is a maximum, so the 7-th joint state is within the initiation set I_{ω_1} and denoted as $(2,3)$ according to Equation (1). As shown in Figure 2, both of the two options can lead to a connected graph when applied to $\widetilde{G}$. Thus, the adoption of multi-agent options can encourage efficient exploration of the joint state space by improving its algebraic connectivity, and we can discover multi-agent options based on individual agents' state space, so that we can enjoy the ease of decomposition.

In the following sections, we will formalize our algorithm – Multi-agent Covering Option Discovery through Kronecker Product of Factor Graphs, and show empirically the significant performance improvement brought by integrating multi-agent options in MARL.

4.3 Multi-agent Covering Option Discovery

In this paper, we adopt Algorithm 1 to construct multi-agent options, based on the individual state transition graphs of each agent which is represented as a list of adjacency matrices $A_{1:n}$. First, in Line 5-9 of Algorithm 1, we acquire the estimation of the Fielder vector F of the joint state space through THEOREM 4.1 based on $A_{1:n}$, so that we can collect the joint states corresponding to the minimum or maximum of F. Then, in Line 10 of Algorithm 1, we split each joint state into a list of individual states. For example, after getting a pair of joint states (s_{min}, s_{max}), we convert them into $((s^1_{min}, \cdots, s^n_{min}),$ $(s^1_{max}, \cdots, s^n_{max}))$, so that we can connect (s_{min}, s_{max}) in the joint state space by connecting each (s^i_{min}, s^i_{max}) in the corresponding individual state space. According to THEOREM 4.1, we estimate F as the kronecker product of n eigenvectors, where the i-th vector is an eigenvector of agent i's normalized laplacian matrix. Therefore, we can get the relationship between the index of the joint state and the indexes of its corresponding individual states based on the definition of kronecker product, which is shown in Line 10.

After decentralizing the joint states, we can generate the multi-agent options through Algorithm 2. For each option ω, we define its initiation set I_ω as the explored joint states, and its termination set β_ω as a joint state in $MIN \cup MAX$ or the unexplored area. As mentioned in Section 3.2, an option ω is available in state s if and only if $s \in I_\omega$. Therefore, instead of constructing a point option between (s_{min}, s_{max}), i.e., setting $\{s_{min}\}$ ($\{s_{max}\}$) as I_ω and $\{s_{max}\}$ ($\{s_{min}\}$) as β_ω, we extend I_ω to the known area to increase the accessibility of ω without loss of the connectivity of the joint state space. As for the multi-agent intra-option policy π_ω used for connecting the initiation joint state and termination joint state, we divide it into a list of single-agent policies, where π^i_ω can be trained with any single-agent RL algorithm aiming at leading agent i from its own initiation state to the termination state s^i_{min} (s^i_{max}). At last, before entering the next loop, we adopt Algorithm 3 to update the individual state transition graphs with the newly-discovered options. This whole process (Line 5-13 in Algorithm 1) is repeated until we get a certain number of options.

To sum up, the proposed algorithm first discovers the joint states that need to be explored most, and then build multi-agent options to encourage agents to visit these sub-goals. More precisely, we project each sub-goal joint state into its corresponding individual state spaces and train the intra-option policy for each agent to visit the projection of the sub-goal state in its individual state space.

4.4 Adopting Multi-agent Options in MARL

In order to take advantage of options in the learning process, we adopt a hierarchical algorithm framework, shown in Figure 3. When making decisions, the RL agent first decides on which option ω to use according to the high-level policy (the primitive actions can be viewed as one-step options), and then decides on the action to take based on the corresponding intra-option policy π_ω. Note that the agent does not decide on a new option with the high-level policy until the current option terminates.

For a multi-agent option $\omega : < I_\omega = \{\textit{the explored joint states}\}, \pi_\omega = (\pi_\omega^1, \cdots, \pi_\omega^n), \beta_\omega = \{(s_1, \cdots, s_n)\} >$, it can be adopted either in a decentralized or centralized way. As shown by the purple arrows in Figure 3, the agents choose their own options independently, and they may choose different options to execute in the meantime. In this case, if agent i selects option ω, it will execute π_ω^i until it reaches its termination state s_i or an unknown individual state. On the other hand, we can force the agents to execute the same multi-agent option simultaneously. To realize this, as shown by the blue arrows in Figure 3, we view the n agents as a whole, which takes the joint state as the input and chooses primitive actions or the same multi-agent option to execute at a time. Once a multi-agent option ω is chosen, agent $1:n$ will execute $\pi_\omega^{1:n}$ until they reach the termination joint state $(s_1, \cdots, s_n)$ or an unexplored joint state. We note that if there are m primitive actions and k multi-agent options, the size of the search space would be $(m+k)^n$ for the decentralized approach and $m^n + k$ for the centralized approach. Therefore, the decentralized way is more flexible but has a larger search space. While, the centralized way fails to consider all the possible solutions but makes it easier for the agents to visit the sub-goal joint states, since the agents simultaneously select the same joint option which will not terminate until the agents arrive at a sub-goal state. In this paper, we use Independent Q-Learning (adopting Q-Learning to each individual agent) [23] to train the decentralized high-level policy, and Centralised Q-Learning (viewing the n agents as a whole and adopting Q-Learning to this joint agent) to train the centralised high-level policy. We will present the comparisons between these two ways in Section 5.

Further, we note that the centralized high-level policy may not be applicable when the number of agents n is large, since both the input space and output space will grow exponentially with n. However, in practice, a multi-agent task can usually be divided into some sub-tasks, each of which can be completed by a sub-group of the agents. For each sub-group, we can learn a list of multi-agent options, and then the agents within this group can make use of these options in a decentralized or centralized way as mentioned above. If there is no way to divide the (identical) agents based on sub-tasks, we can still group them randomly to a list of two-agent or three-agent sub-groups. Agents within the same sub-group will co-explore their joint state space using the algorithm framework shown as Figure 3. In Section 5, we show that the adoption of grouping techniques can not only accelerate the exploration but also can greatly improve the scalability of our algorithm.

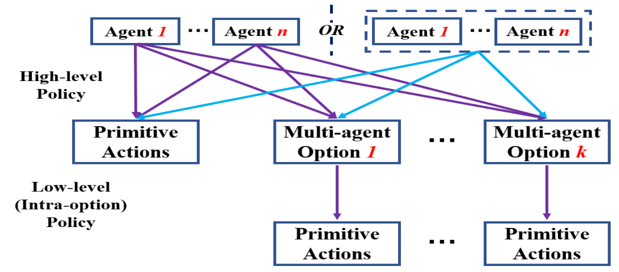


Figure 3: Hierarchical algorithm framework: When making decisions, the agent first decides on which option ω to use according to the high-level policy, and then decides on the primitive action to take based on the corresponding intra-option policy π_ω. The agents can decide on their options independently (the left side) or jointly (the right side).

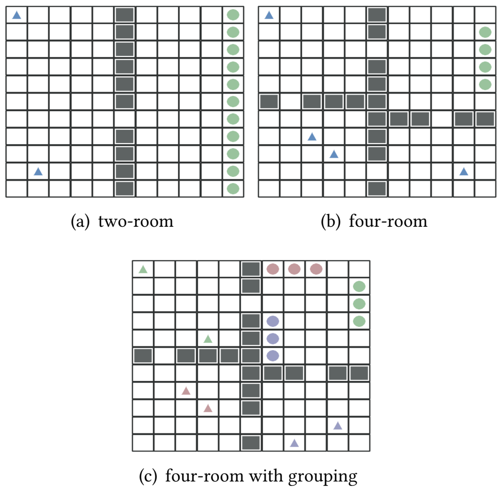


Figure 4: Simulators for Evaluation

5 EVALUATION AND RESULTS

5.1 Simulation Setup

As shown in Figure 4, the proposed approach is evaluated using three different settings. (1) Two-room task: two agents (the triangles) must reach the goal area (the circles) at the same time to complete this task, and they can't get through the walls (the squares). (2) Four-room task: n (2~8) agents must reach the goal area at the same time to complete this task. (3) Four-room with grouping task: there are m groups of agents, and each group contains n agents. Each group of agents has a special goal area. As shown in Figure 4(c), the agents and their related goal area are labeled with the same color. Note that all the $m \times n$ agents should get to their related areas at the same time to complete this task, and the agents don't know which goal area is related to them at first. For all the three tasks, different agents can share the same grid, and only when the agents complete

Algorithm 1 Multi-agent Covering Option Discovery

```
1: Input: number of agents $n$, list of adjacency matrices $A_{1:n}$, number of options to generate $tot\_num$
2: Output: list of multi-agent options $\Omega$
3: $\Omega \leftarrow \emptyset$, $cur\_num \leftarrow 0$
4: while $cur\_num < tot\_num$ do
5:     Collect the degree list of each individual state transition graph $D_{1:n}$ according to $A_{1:n}$
6:     Obtain the list of normalized laplacian matrices $\mathcal{L}_{1:n}$ corresponding to $A_{1:n}$
7:     Calculate the eigenvalues $U_i$ and corresponding eigenvectors $V_i$ for each $\mathcal{L}_i$ and collect them as $U_{1:n}$ and $V_{1:n}$
8:     Obtain the Fielder vector $F$ of the joint state space using THEOREM 4.1 based on $D_{1:n}$, $U_{1:n}$ and $V_{1:n}$
9:     Collect the list of joint states corresponding to the minimum or maximum in $F$, named $MIN$ and $MAX$ respectively
10:    Convert each joint state $s_{joint}$ in $MIN$ and $MAX$ to $(s_1, \cdots, s_n)$, where $s_i$ is the corresponding individual state of agent $i$, based on
       the equation:$ind(s_{joint}) = ((ind(s_1) * dim(A_2) + ind(s_2)) * dim_{A_3} + \cdots + ind(s_{n-1})) * dim(A_n) + ind(s_n)$, where $dim(A_i)$ is the
       dimension of $A_i$, $ind(s_i)$ is the index of $s_i$ (indexed from 0) in the individual state space of agent $i$
11:    Generate a new list of options $\Omega'$ through Algorithm 2
12:    $\Omega \leftarrow \Omega \cup \Omega'$, $cur\_num \leftarrow cur\_num + len(\Omega')$
13:    Update $A_{1:n}$ through Algorithm 3
14: end while
15: Return $\Omega$
```

Algorithm 2 Generate Multi-agent Options

```
1: Input: $MIN, MAX$: list of joint states corresponding to the minimum or maximum in the Fielder vector
2: Output: list of multi-agent options $\Omega'$
3: $\Omega' \leftarrow \emptyset$
4: for $s = (s_1, \cdots, s_n)$ in $(MIN \cup MAX)$ do
5:     Define the initiation set $I_\omega$ as the joint states in the known region of the joint state space
6:     Define the termination condition: $\beta_\omega(s_{cur}) \leftarrow \begin{cases} 1 & if\ (s_{cur} == s)\ or\ (s_{cur}\ is\ unknown) \\ 0 & otherwise \end{cases}$, where $s_{cur}$ is the current joint state
7:     Train the intra-option policy $\pi_\omega = (\pi_\omega^1, \cdots, \pi_\omega^n)$, where $\pi_\omega^i$ maps the individual state of agent $i$ to its action aiming at leading
       agent $i$ from any state in its initiation set to its termination state $s_i$
8:     $\Omega' \leftarrow \Omega' \cup \{< I_\omega, \pi_\omega, \beta_\omega >\}$
9: end for
10: Return $\Omega'$
```

Algorithm 3 Update Adjacency Matrices

```
1: Given: list of adjacency matrices $A_{1:n}$, list of joint states corresponding to the minimum or maximum in the Fielder vector
   $MIN, MAX$
2: for $s_{min} = (s^1_{min}, \cdots, s^n_{min})$ in $MIN$ do
3:     for $s_{max} = (s^1_{max}, \cdots, s^n_{max})$ in $MAX$ do
4:         for $i = 1$ to $n$ do
5:             $A_i[ind(s^i_{min})][ind(s^i_{max})] = 1$
6:             $A_i[ind(s^i_{max})][ind(s^i_{min})] = 1$
7:         end for
8:     end for
9: end for
```

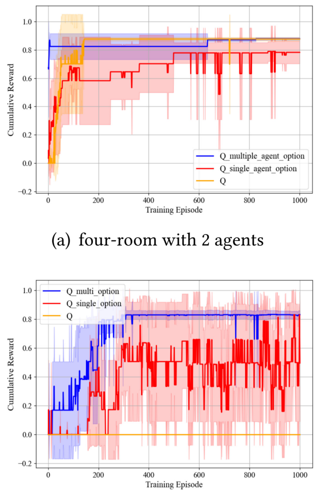
the task can they receive a reward signal $r = 1.0$, which is shared by all the agents; otherwise, they will receive $r = 0.0$. Hence, the joint reward function is very sparse, and will become more sparse as the number of agents grows. We use the episodic cumulative reward as the metric, which is defined as: $\sum_{i=0}^{l} \lambda^i r$, where $\lambda = 0.99$, and $l \leq 200$ is the length of each episode.

We compare our approach – agents with multi-agent options, with two baselines: (1) Agents with single-agent options: we construct covering options for each agent based on their individual state spaces, and then utilize these options in MARL with the framework shown as Figure 3. We use the SOTA option discovery method proposed in [9]. Also, we extend the initiation set of each single-agent option to the known area to increase their accessibility, like what we do with multi-agent options. (2) Agents without options: we adopt MARL algorithms directly on the evaluation tasks. The multi-agent options are constructed to improve the joint exploration of all the agents within the scenario (or a sub-group), which is expected to have superior performance than the baselines. To confirm the fairness, we set the number of single-agent options and multi-agent options for each agent to select as the same.

There are two kinds of policies in Figure 3: the high-level policy for selecting among options, and the low-level policy for selecting among primitive actions. We evaluate the performance of agents with multi-agent options using five different algorithms as the high-level policy: random policy, Independent Q-Learning, Distributed Q-Learning [10] (each agent decides on their own option based on the joint state), Centralized Q-Learning and Centralized Q-Learning

Algorithm	Input	Output	How to utilize multi-agent options
Random	–	Individual action	Decentralized
Independent Q-Learning [23]	Individual state	Individual action	Decentralized
Distributed Q-Learning [10]	Joint state	Individual action	Decentralized
Centralized Q-Learning	Joint state	Joint action	Decentralized
Centralized Q-Learning + Force	Joint state	Joint action	Centralized

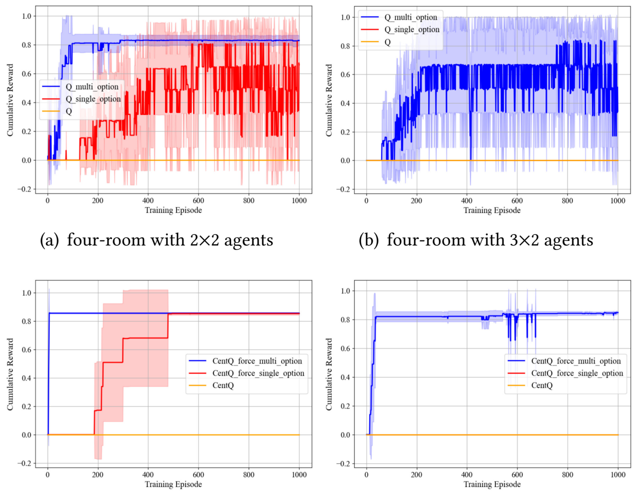
Table 1: Comparisons among different high-level policy algorithms



(a) four-room with 2 agents (b) four-room with 3 agents

(c) four-room with 4 agents (d) four-room with 5 agents

Figure 5: Evaluation on n-agent four-room tasks, using Independent Q-Learning as the high-level policy. The performance improvement of our approach are more and more significant as the number of agents increases.



(a) four-room with 2×2 agents (b) four-room with 3×2 agents

(c) four-room with 2×2 agents (d) four-room with 3×2 agents

Figure 6: Comparisons on the $m \times n$ four-room tasks: (a)-(b) Independent Q-Learning; (c)-(d) Centralized Q-Learning + Force. Agents with pairwise options can learn these tasks much faster than the baselines, even when both the baselines fail on the 3×2 four-room task. Also, agents trained with Centralized Q-Learning + Force have faster convergence speed and higher convergence value than those trained with Independent Q-Learning.

+ Force, to make sure that the performance improvement is not specific to a certain algorithm. Table 1 shows the comparisons among these algorithms. For “Centralized Q-Learning", agents can choose different options to execute simultaneously, while for “Centralized Q-Learning + Force" agents are forced to choose the same multi-agent option at a time. As for the low-level policy, we adopt Value Iteration [21] to find the optimal path between each pair of initiation and termination state for each agent i as π^i_ω. Compared with Baseline (1), our approach does not require extra cost for learning the low-level policy, since the number of single-agent options and multi-agent options are the same for each agent.

5.2 Main Results

For each experiment in this section, we present comparisons among the performance of agents with multi-agent options (the blue line), agents with single-agent options (the red line) and agents without options (the orange line). We run each experiment for five times and plot the change of the mean value and standard deviation of the cumulative reward during the training process (1000 episodes).

Two-room task & Two-agent four-room task: As shown in Appendix B.1 and B.2, we present comparisons on two simpler tasks: two-room task and two-agent four-room task respectively. It can be observed that for both tasks, no matter which algorithm we adopt as the high-level policy, agents with multi-agent options can converge faster than the baselines. However, when using Independent Q-Learning to train the high-level policy, the performance of our approach and the baselines are very close. Thus, in the follow-up experiments, we compare these approaches on more challenging tasks with Independent Q-Learning as the high-level policy to see if there will be more significant performance increase for our proposed method. Also, we will adopt Centralized Q-Learning + Force to train the high-level policy in the following experiments, to compare the two manners (decentralized or centralized) to utilize the multi-agent options.

***N*-agent four-room task:** In Figure 5, we test these methods on n-agent four-room tasks ($n = 2 \sim 5$), using Independent Q-learning as the high-level policy. We can observe that the performance improvement brought by our approach are more and more significant as the number of agents increases. When $n = 5$, both the baselines fail to complete the task, while agents with five-agent options can converge within ~ 200 episodes. On the other hand, in Appendix B.3, we show the results of using Centralized Q-Learning + Force as the high-level policy on the same tasks. We can see that the centralized way to utilize the n-agent options leads to faster convergence, since the joint output space of the agents is pruned.

Four-room task with sub-task grouping: The size of the joint state space and output space grows exponentially with the number of agents, making it infeasible to directly construct n-agent options and adopt Centralized Q-Learning for a large n. However, in real-life scenarios, a multi-agent task can usually be divided into sub-tasks, and the agents can be divided into sub-groups based on the sub-tasks they are responsible for. Thus, we test our proposed method on the $m \times n$ four-room tasks shown as Figure 4(c), where we divide the agents into m sub-groups, each of which contains n agents with the same goal area. Figure 6 shows comparisons between our method and the baselines on $m \times n$ four-room tasks. Note that, in the 2×2 (3×2) four-room task, we use two-agent (pairwise) options rather than four-agent (six-agent) options, and when using Centralized Q-Learning + Force, we only use the joint state space of the two agents as input to decide on their joint option choice. We can see that agents with pairwise options can learn to complete the tasks much faster than the baselines (e.g., improved by about two orders of magnitude in the 2×2 four-room task), even when both the baselines fail to complete the 3×2 four-room task. Also, we see that agents trained with Centralized Q-Learning + Force (Figure 6(c)-6(d)) have faster convergence speed and higher convergence value compared to training with Independent Q-Learning (Figure 6(a)-6(b)).

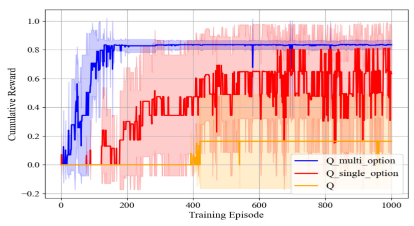
Four-room task with random grouping: Further, we note that our method also works with random grouping when sub-task grouping may not work. The intuition is that adopting two-agent or three-agent options can encourage the joint exploration of the agents in small sub-groups, which can increase the overall performance compared with only utilizing single-agent explorations. As shown in Figure 7, we compare the performance of agents with pairwise options, single-agent options and no options on the n-agent four-room tasks ($n = 4, 6, 8$). We can observe that when $n = 6$ *or* 8, agents with single-agent options or no options can't complete this task, while we can get a significant performance improvement with only pairwise options. On the other hand, agents with pairwise options can't complete the most challenging eight-agent four-room task, if we use Independent Q-Learning to train the high-level policy, shown as Figure 7(c). However, if we adopt Centralized Q-Learning + Force, agents with pairwise options can still complete this challenging task with satisfaction, shown as Figure 7(f). Further, in Appendix B.4, we show how the performance of agents with pairwise options would change with the number of options, based on the six-agent and eight-agent four-room tasks.

Four-room task with random grouping and dynamic influences among agents: Finally, we show that even if in environments where an agent's state transitions can be strongly influenced by the others, we can still obtain good approximations of the multi-agent options to encourage joint exploration using THEOREM 4.1. For this new setting, we make some modifications based on the n-agent four-room task – different agents cannot share the same grid so that an agent may be blocked by others when moving ahead, and this influence is dynamic. We use the Centralized Q-Learning + Force as the high-level policy, of which the results are shown as Figure 8. We can see that although this modification affects the performance of agents with single-agent options, we can still get significant performance improvement using the pairwise options.

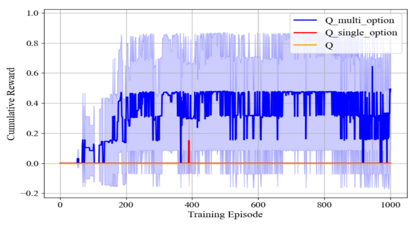
We present the numeric results of all the experiments in tabular in Appendix B.5, which show that agents with multi-agent options outperform the baselines in **ALL** the multi-agent tasks in terms of faster exploration and higher cumulative rewards.

6 CONCLUSION

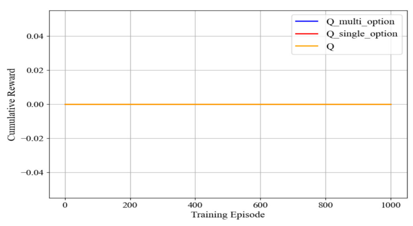
In this paper, we propose to approximate the joint state space in MARL as a kronecker graph and estimate its Fiedler vector using the Laplacian spectrum of the individual agents' state transition graphs. Based on the approximation of the Fiedler vector, multi-agent covering options are constructed, containing multiple agents' temporal action sequence towards the sub-goal joint states which are usually infrequently visited, so as to accelerate the joint exploration in the environment. Further, we propose algorithms to adopt these multi-agent options in MARL, using centralized, decentralized, and group-based strategies, respectively. We show through evaluation results that agents with multi-agent options have superior performance than agents relying on single-agent options or no options in three different multi-agent tasks. Future works will focus on scaling our algorithm for real-life applications with SOTA representation learning and deep learning techniques.



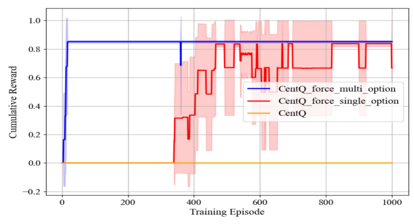
(a) four-room with 4 agents



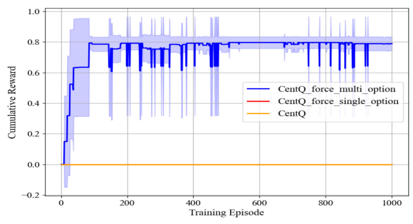
(b) four-room with 6 agents



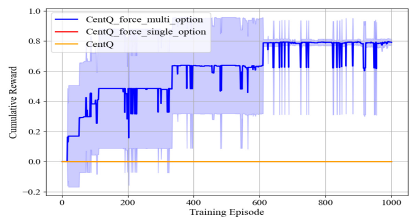
(c) four-room with 8 agents



(d) four-room with 4 agents

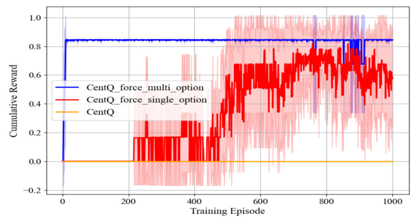


(e) four-room with 6 agents

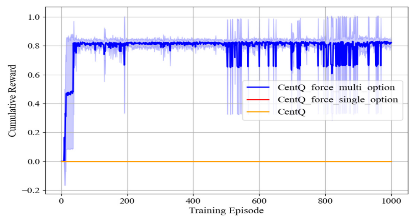


(f) four-room with 8 agents

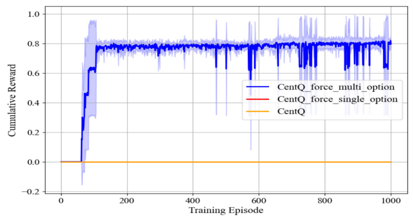
Figure 7: Comparisons on the n-agent four-room tasks: (a)-(c) Independent Q-Learning; (d)-(f) Centralized Q-Learning + Force. When n-agent options are not available, we can still get a significant performance improvement with only pairwise options. Adopting Centralized Q-Learning + Force within these sub-groups can further improve the convergence speed and value, compared with adopting Independent Q-Learning, e.g., when $n = 6$, the cumulative reward is improved by $\sim 100\%$.



(a) four-room with 4 agents



(b) four-room with 6 agents



(c) four-room with 8 agents

Figure 8: Comparisons on the n-agent four-room tasks where agent's state transitions can be influenced by the others, using Centralized Q-Learning + Force as the high-level policy. On this setting, we can still obtain good approximations of the multi-agent options based on the theory introduced in Section 4.2 and use them to get superior performance.

REFERENCES

[1] Christopher Amato, George Konidaris, Leslie P Kaelbling, and Jonathan P How. 2019. Modeling and planning with macro-actions in decentralized POMDPs. *Journal of Artificial Intelligence Research* 64 (2019), 817–859.

[2] Christopher Amato, George Dimitri Konidaris, and Leslie Pack Kaelbling. 2014. Planning with macro-actions in decentralized POMDPs. In *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14*. IFAAMAS/ACM, 1273–1280.

[3] Milan Bašić, Branko Arsić, and Zoran Obradović. 2021. Another estimation of Laplacian spectrum of the Kronecker product of graphs. arXiv:2102.02924 [cs.SI]

[4] Jhelum Chakravorty, Patrick Nadeem Ward, Julien Roy, Maxime Chevalier-Boisvert, Sumana Basu, Andrei Lupu, and Doina Precup. 2020. Option-Critic in Cooperative Multi-agent Systems. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20*. 1792–1794.

[5] Miroslav Fiedler. 1973. Algebraic connectivity of graphs. *Czechoslovak mathematical journal* 23, 2 (1973), 298–305.

[6] Arpita Ghosh and Stephen Boyd. 2006. Growing Well-connected Graphs.

[7] Jean Harb, Pierre-Luc Bacon, Martin Klissarov, and Doina Precup. 2018. When Waiting is not an Option : Learning Options with a Deliberation Cost. In *AAAI*.

[8] Yuu Jinnai, David Abel, David Ellis Hershkowitz, Michael L. Littman, and George Dimitri Konidaris. 2019. Finding Options that Minimize Planning Time. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 3120–3129. http://proceedings.mlr.press/v97/jinnai19a.html

[9] Yuu Jinnai, Jee Won Park, David Abel, and George Dimitri Konidaris. 2019. Discovering Options for Exploration by Minimizing Cover Time. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 3130–3139. http://proceedings.mlr.press/v97/jinnai19b.html

[10] Martin Lauer and Martin A. Riedmiller. 2000. An Algorithm for Distributed Reinforcement Learning in Cooperative Multi-Agent Systems. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*, Pat Langley (Ed.). Morgan Kaufmann, 535–542.

[11] Youngwoon Lee, Jingyun Yang, and Joseph J. Lim. 2020. Learning to Coordinate Manipulation Skills via Skill Behavior Diversification. In *8th International Conference on Learning Representations, ICLR 2020*. OpenReview.net.

[12] Marlos C. Machado, Marc G. Bellemare, and Michael H. Bowling. 2017. A Laplacian Framework for Option Discovery in Reinforcement Learning. *CoRR* abs/1703.00956 (2017). arXiv:1703.00956 http://arxiv.org/abs/1703.00956

[13] Marlos C. Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauro, and Murray Campbell. 2018. Eigenoption Discovery through the Deep Successor Representation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=Bk8ZcAxR-

[14] Daniel J. Mankowitz, Timothy A. Mann, and Shie Mannor. 2016. Adaptive Skills Adaptive Partitions (ASAP). In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (Eds.). 1588–1596.

[15] Amy McGovern and Andrew G. Barto. 2001. Automatic Discovery of Subgoals in Reinforcement Learning using Diverse Density. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, Carla E. Brodley and Andrea Pohoreckyj Danyluk (Eds.). Morgan Kaufmann, 361–368.

[16] Ishai Menache, Shie Mannor, and Nahum Shimkin. 2002. Q-Cut - Dynamic Discovery of Sub-goals in Reinforcement Learning. In *Machine Learning: ECML 2002, 13th European Conference on Machine Learning, volume 2430 of LectureNotes in Computer Science*. Springer, 295–306.

[17] Jing Shen, Guochang Gu, and Haibo Liu. 2006. 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, 676–682.

[18] Özgür Simsek and Andrew G. Barto. 2008. Skill Characterization Based on Betweenness. In *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou (Eds.). Curran Associates, Inc., 1497–1504.

[19] Özgür Simsek, Alicia P. Wolfe, and Andrew G. Barto. 2005. Identifying useful subgoals in reinforcement learning by local graph partitioning. In *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005 (ACM International Conference Proceeding Series, Vol. 119)*, Luc De Raedt and Stefan Wrobel (Eds.). ACM, 816–823. https://doi.org/10.1145/1102351.1102454

[20] Martin Stolle and Doina Precup. 2002. Learning options in reinforcement learning. In *International Symposium on abstraction, reformulation, and approximation*. Springer, 212–223.

[21] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[22] Richard S. Sutton, Doina Precup, and Satinder P. Singh. 1999. Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artif. Intell.* 112, 1-2 (1999), 181–211. https://doi.org/10.1016/S0004-3702(99)00052-1

[23] Ming Tan. 1993. Multi-Agent Reinforcement Learning: Independent versus Cooperative Agents. In *Machine Learning, Proceedings of the Tenth International Conference, University of Massachusetts, Amherst, MA, USA, June 27-29, 1993*, Paul E. Utgoff (Ed.). Morgan Kaufmann, 330–337. https://doi.org/10.1016/b978-1-55860-307-3.50049-6

[24] Paul M Weichsel. 1962. The Kronecker product of graphs. *Proceedings of the American mathematical society* 13, 1 (1962), 47–52.

A SUPPLEMENT MATERIALS FOR THE PROPOSED ALGORITHM

A.1 Proof of THEOREM 4.1

For convenience, we use G_i to represent the i-th factor graph and its adjacency matrix. Also, we denote the number of nodes in G_i as K_i and an identity matrix with K_i diagonal elements as I_{K_i}.

Proof. The normalized laplacian matrix of the kronecker product of n factor graphs $\otimes_{i=1}^{n} G_i$ can be written as:

$$\mathcal{L}_{\otimes_{i=1}^{n} G_i} = \otimes_{i=1}^{n} I_{K_i} - (\otimes_{i=1}^{n} D_{G_i}^{-\frac{1}{2}})(\otimes_{i=1}^{n} G_i)(\otimes_{i=1}^{n} D_{G_i}^{-\frac{1}{2}}). \tag{8}$$

Using the property of the kronecker product of matrices, $(A \otimes B)(C \otimes D) = AC \otimes BD$, we can obtain that:

$$\begin{aligned}\mathcal{L}_{\otimes_{i=1}^{n} G_i} &= \otimes_{i=1}^{n} I_{K_i} - \otimes_{i=1}^{n} (D_{G_i}^{-\frac{1}{2}} G_i D_{G_i}^{-\frac{1}{2}}) \\ &= \otimes_{i=1}^{n} I_{K_i} - \otimes_{i=1}^{n} (I_{K_i} - \mathcal{L}_{G_i}).\end{aligned} \tag{9}$$

Let $\{\lambda_{k_1}^{G_1}\}, \{\lambda_{k_2}^{G_2}\}, \dots, \{\lambda_{k_n}^{G_n}\}$ be the eigenvalues of matrices $\mathcal{L}_{G_1}, \mathcal{L}_{G_2}, \dots, \mathcal{L}_{G_n}$, with the corresponding orthonormal eigenvectors $\{v_{k_1}^{G_1}\}$, $\{v_{k_2}^{G_2}\}, \dots, \{v_{k_n}^{G_n}\}$, where $k_i = 1, 2, \dots, K_i$. Also, denote the diagonal matrices, whose diagonal elements are the values $\{1 - \lambda_{k_1}^{G_1}\}, \{1 - \lambda_{k_2}^{G_2}\}, \dots, \{1 - \lambda_{k_n}^{G_n}\}$, as $\Lambda_{G_1}, \Lambda_{G_2}, \dots, \Lambda_{G_n}$, and the square matrices containing the eigenvectors $\{v_{k_1}^{G_1}\}, \{v_{k_2}^{G_2}\}, \dots, \{v_{k_n}^{G_n}\}$ as the column vectors as $V_{G_1}, V_{G_2}, \dots, V_{G_n}$. Using the spectral decomposition of the matrix $I_{K_i} - \mathcal{L}_{G_i}$ $(i = 1, \dots, n)$, we can obtain that:

$$\begin{aligned}\mathcal{L}_{\otimes_{i=1}^{n} G_i} &= \otimes_{i=1}^{n} I_{K_i} - \otimes_{i=1}^{n} (V_{G_i} \Lambda_{G_i} V_{G_i}^{T}) \\ &= \otimes_{i=1}^{n} I_{K_i} - (\otimes_{i=1}^{n} V_{G_i})(\otimes_{i=1}^{n} \Lambda_{G_i})(\otimes_{i=1}^{n} V_{G_i})^{T} \\ &= (\otimes_{i=1}^{n} V_{G_i})(\otimes_{i=1}^{n} I_{K_i} - \otimes_{i=1}^{n} \Lambda_{G_i})(\otimes_{i=1}^{n} V_{G_i})^{T},\end{aligned} \tag{10}$$

since $\otimes_{i=1}^{n} I_{K_i} = \otimes_{i=1}^{n} [(V_{G_i})(V_{G_i})^{T}] = (\otimes_{i=1}^{n} V_{G_i})(\otimes_{i=1}^{n} V_{G_i})^{T}$. This implies that $\mathcal{L}_{\otimes_{i=1}^{n} G_i}$ has eigenvalues $\{[1 - \prod_{i=1}^{n}(1 - \lambda_{k_i}^{G_i})]\}$ and corresponding eigenvectors $\{\otimes_{i=1}^{n} v_{k_i}^{G_i}\}$.

Then, we let $\Lambda = \otimes_{i=1}^{n} I_{K_i} - \otimes_{i=1}^{n} \Lambda_{G_i}$ and $D = \otimes_{i=1}^{n} D_{G_i}$. Since the normalized Laplacian could be expressed in terms of Laplacian matrix as $\mathcal{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$, we can get $L_{\otimes_{i=1}^{n} G_i}(\otimes_{i=1}^{n} V_{G_i}) = D^{\frac{1}{2}} \mathcal{L}_{\otimes_{i=1}^{n} G_i} D^{\frac{1}{2}} (\otimes_{i=1}^{n} V_{G_i})$. By making assumption (used in [3]) that $D_{G_i}^{\frac{1}{2}} V_{G_i} \approx V_{G_i} D_{G_i}^{\frac{1}{2}}$, for $i = 1, 2, \dots, n$, we can derive that:

$$\begin{aligned}L_{\otimes_{i=1}^{n} G_i}(\otimes_{i=1}^{n} V_{G_i}) &\approx D^{\frac{1}{2}} \mathcal{L}_{\otimes_{i=1}^{n} G_i}(\otimes_{i=1}^{n} V_{G_i}) D^{\frac{1}{2}} \\ &= D^{\frac{1}{2}} \Lambda (\otimes_{i=1}^{n} V_{G_i}) D^{\frac{1}{2}}.\end{aligned} \tag{11}$$

After applying the same assumption again, we finally obtain that:

$$L_{\otimes_{i=1}^{n} G_i}(\otimes_{i=1}^{n} V_{G_i}) \approx (D\Lambda)(\otimes_{i=1}^{n} V_{G_i}). \tag{12}$$

Based on Equation (12), we can get an approximation of the Laplacian spectrum, including the eigenvalues and corresponding eigenvectors, of the kronecker product of n factor graphs, shown as THEOREM 4.1.

Next, we will prove that the estimated eigenvalues $\mu_{k_1 k_2, \dots, k_n}$ in THEOREM 4.1 are non-negative. It is obvious that $d_{k_i}^{G_i}$ and $\prod_{i=1}^{n} d_{k_i}^{G_i}$ are non-negative. Then, we need to prove $[1 - \prod_{i=1}^{n}(1 - \lambda_{k_i}^{G_i})]$ is non-negative. We know that if λ is an eigenvalue of a normalized Laplacian matrix, we can get $0 \le \lambda \le 2$. Hence, $-1 \le 1 - \lambda_{k_i}^{G_i} \le 1$, for $i = 1, 2, \dots, n$. Based on this, we can get that $\left|\prod_{i=1}^{n}(1 - \lambda_{k_i}^{G_i})\right| \le 1$ and thus $[1 - \prod_{i=1}^{n}(1 - \lambda_{k_i}^{G_i})]$ is non-negative. ◻

A.2 Finding the Fiedler vector for the illustrative example shown in Figure 1(a)

(1) Compute the normalized Laplacian matrix of G_1 and G_2, namely $\mathcal{L}_1$ and $\mathcal{L}_2$:

$$\mathcal{L}_1 = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \ \mathcal{L}_2 = \begin{bmatrix} 1 & -\frac{1}{\sqrt{2}} & 0 & 0 \\ -\frac{1}{\sqrt{2}} & 1 & -\frac{1}{2} & 0 \\ 0 & -\frac{1}{2} & 1 & -\frac{1}{\sqrt{2}} \\ 0 & 0 & -\frac{1}{\sqrt{2}} & 1 \end{bmatrix}. \tag{13}$$

(2) Compute the eigenvalues and eigenvectors of $\mathcal{L}_1$ and $\mathcal{L}_2$:

$$\lambda_1^{G_1} = 0, \ \ \lambda_2^{G_1} = 2, \ \ v_{1:2}^{G_1} = \frac{1}{\sqrt{2}} \left[\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right]. \tag{14}$$

$$\lambda_1^{G_2} = 0, \ \ \lambda_2^{G_2} = 0.5, \ \ \lambda_3^{G_2} = 1.5, \ \ \lambda_4^{G_2} = 2, \ \ v_{1:4}^{G_2} = \frac{1}{\sqrt{3}} \left[\begin{bmatrix} \frac{1}{\sqrt{2}} \\ 1 \\ 1 \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \begin{bmatrix} -1 \\ -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -\frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \\ 1 \end{bmatrix}, \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -1 \\ 1 \\ -\frac{1}{\sqrt{2}} \end{bmatrix} \right]. \tag{15}$$

(3) Compute the degree list of G_1 and G_2 (sorted in ascending order), namely d^{G_1} and d^{G_2}:

$$d^{G_1} = [1, \ 1]^T, \ d^{G_2} = [1, \ 1, \ 2, \ 2]^T. \tag{16}$$

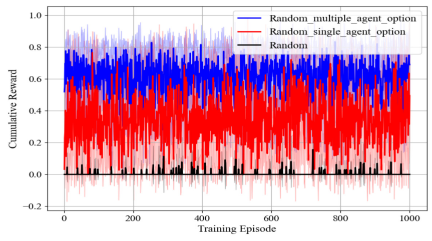
(4) According to THEOREM 4.1, we can get two approximations of the Fiedler vector:

$$v_{11} = v_1^{G_1} \otimes v_2^{G_2} = \frac{1}{\sqrt{6}} \left[\frac{1}{\sqrt{2}}, \ 1, \ 1, \ \frac{1}{\sqrt{2}}, \ \frac{1}{\sqrt{2}}, \ 1, \ 1, \ \frac{1}{\sqrt{2}} \right]^T, \tag{17}$$

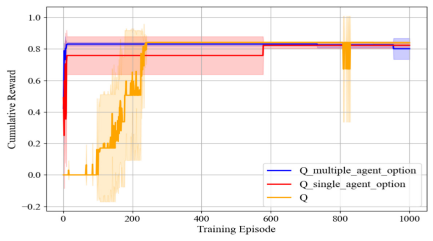
$$v_{24} = v_2^{G_1} \otimes v_4^{G_2} = \frac{1}{\sqrt{6}} \left[-\frac{1}{\sqrt{2}}, \ 1, \ -1, \ \frac{1}{\sqrt{2}}, \ \frac{1}{\sqrt{2}}, \ -1, \ 1, \ -\frac{1}{\sqrt{2}} \right]^T. \tag{18}$$

B SUPPLEMENT MATERIALS FOR THE EVALUATION PART

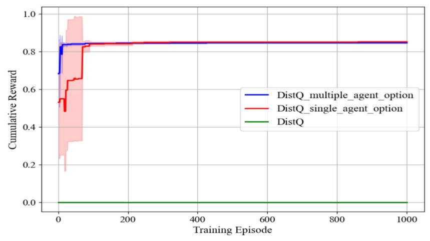
B.1 Evaluation results on the two-room task



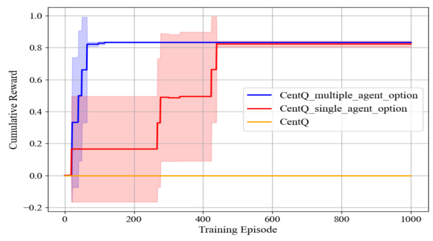
(a) Random



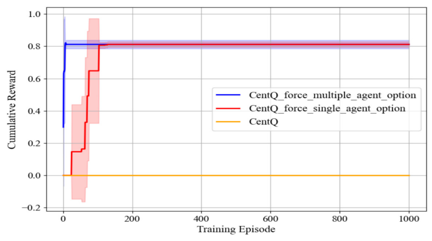
(b) Independent Q-Learning



(c) Distributed Q-Learning



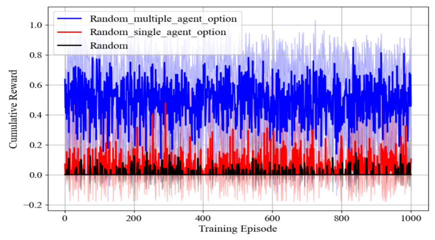
(d) Centralized Q-Learning



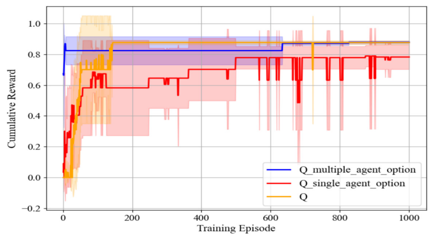
(e) Centralized Q-Learning + Force

Figure 9: Comparisons on the two-room task: (a)-(e) show the results of using different algorithms as the high-level policy. No matter which algorithm we adopt, the agents with multi-agent options can converge faster than the baselines.

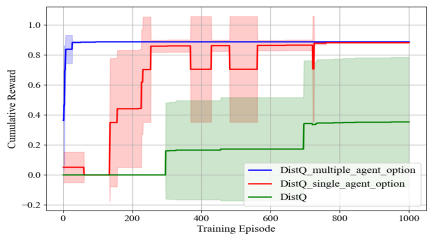
B.2 Evaluation results on the two-agent four-room task



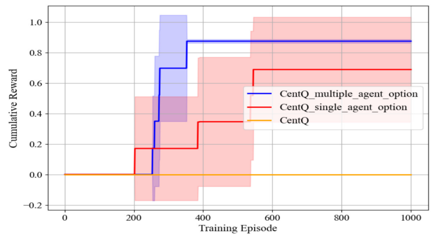
(a) Random



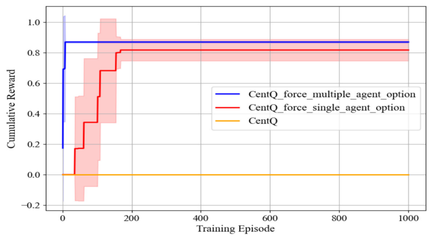
(b) Independent Q-Learning



(c) Distributed Q-Learning



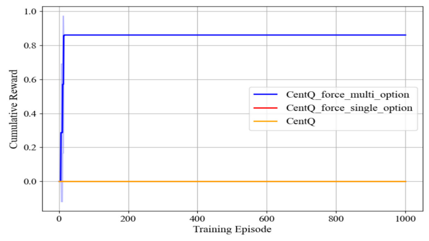
(d) Centralized Q-Learning



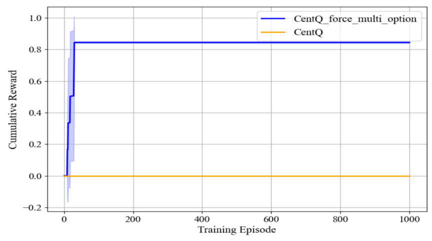
(e) Centralized Q-Learning + Force

Figure 10: Comparisons on the two-agent four-room task: (a)-(e) show the results of using different algorithms as the high-level policy. No matter which algorithm we adopt, agents with multi-agent options can converge faster than the baselines. Also, shown as (a)(d)(e), our approach converges to a higher cumulative reward.

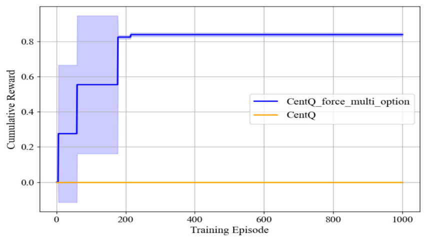
B.3 Comparisons on n-agent four-room tasks using Centralized Q-Learning + Force



(a) four-room with 3 agents



(b) four-room with 4 agents



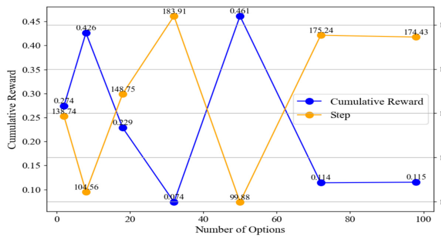
(c) four-room with 5 agents

Figure 11: Comparisons on a serious of four-room tasks: (a)-(c) show the results of using Centralized Q-Learning + Force as the high-level policy. Agents with single-agent options start to fail since the 3-agent case. Compared with the results of using decentralized way shown in Figure 5, the centralized way to utilize the *n*-agent options leads to faster convergence.

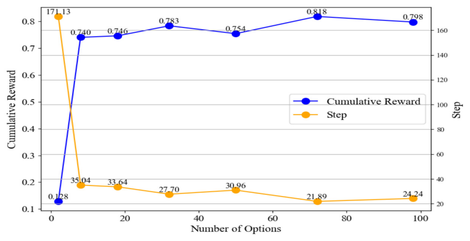
Note that when the number of agents is three, the agents with single-agent options already fail to complete the four-room task. We don't include the results of agents with single-agent options in Figure 11(b)-11(c), because it takes a tremendously long time to run those experiments and it can be predicted that the results will be the same as Figure 11(a).

B.4 Performance change with the number of options on six-agent and eight-agent four-room tasks

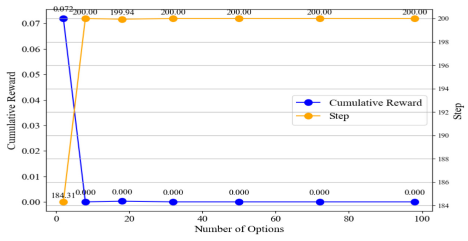
As shown in Figure 12 (the orange line: number of steps to complete the task; the blue line: episodic cumulative reward), if we increase the number of options, the performance of agents with pairwise options and using Centralized Q-Learning + Force as the high-level policy can be improved further. While, if using the Independent Q-Learning as the high-level policy, the agents' performance would go worse as the number of options increase. The reason is that, as mentioned in Section 4.4, the joint output space of the agents will grow exponentially with the number of options if we utilize the multi-agent options in a decentralized way. In contrast, the size of the joint output space is linear with the number of options when we use the multi-agent options in a centralized manner.



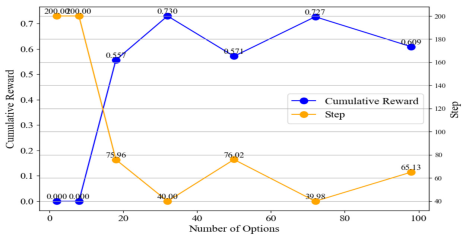
(a) Independent Q-Learning



(b) Centralized Q-Learning + Force



(c) Independent Q-Learning



(d) Centralized Q-Learning + Force

Figure 12: Performance change of the agents with pairwise options as the number of options increase: (a)-(b) six-agent four-room task; (c)-(d): eight-agent four-room task.

B.5 Numeric results of the experiments

In this section, we show the numeric results of our experiments in tabular with the following notations. IQL: Independent Q-Learning, DistQ: Distributed Q-Learning, CentQ: Centralized Q-Learning, CentQ+Force: Centralized Q-Learning + Force, Value: the mean of the episodic cumulative reward during the training process, Step: the mean of the number of steps to complete the task during the training process, Multiple: multi-agent options, Single: single-agent options.

Three-agent four-room task			Four-agent four-room task			Five-agent four-room task		
IQL	Value	Step	IQL	Value	Step	IQL	Value	Step
Multiple	**0.790**	**27.96**	Multiple	**0.716**	**42.95**	Multiple	**0.674**	**50.83**
Single	0.705	43.94	Single	0.386	115.1	Single	0.002	199.6
No options	0.038	191.7	No options	0.0	200.0	No options	0.0	200.0
CentQ+Force	Value	Step	CentQ+Force	Value	Step	CentQ+Force	Value	Step
Multiple	**0.852**	**16.73**	Multiple	**0.829**	**20.25**	Multiple	**0.769**	**32.47**
Single	0.0	200.0	Single	–	–	Single	–	–
No options	0.0	200.0	No options	0.0	200.0	No options	0.0	200.0

Table 2: Numeric results on the n-agent four-room tasks

2×2 four-room task			3×2 four-room task		
IQL	Value	Step	IQL	Value	Step
Multiple	**0.782**	**28.66**	Multiple	**0.504**	**89.34**
Single	0.398	111.2	Single	0.0	200.0
No options	0.0	200.0	No options	0.0	200.0
CentQ+Force	Value	Step	CentQ+Force	Value	Step
Multiple	**0.853**	**16.17**	Multiple	**0.809**	**23.25**
Single	0.614	67.33	Single	0.0	200.0
No options	0.0	200.0	No options	0.0	200.0

Table 3: Numeric results on the $m \times n$ four-room tasks

Two-agent two-room task			Two-agent four-room task		
Random	Value	Step	Random	Value	Step
Multiple	**0.609**	**57.58**	Multiple	**0.496**	**83.98**
Single	0.344	119.1	Single	0.045	189.5
No options	0.004	199.3	No options	0.010	197.9
IQL	Value	Step	IQL	Value	Step
Multiple	**0.828**	**18.90**	Multiple	**0.842**	**17.66**
Single	0.783	25.51	Single	0.685	44.48
No options	0.693	49.18	No options	0.827	23.46
DistQ	Value	Step	DistQ	Value	Step
Multiple	**0.844**	**16.89**	Multiple	**0.883**	**12.58**
Single	0.832	19.57	Single	0.686	52.92
No options	0.0	200.0	No options	0.174	162.7
CentQ	Value	Step	CentQ	Value	Step
Multiple	**0.800**	**25.29**	Multiple	**0.627**	**66.05**
Single	0.589	70.81	Single	0.402	113.7
No options	0.0	200.0	No options	0.0	200.0
CentQ+Force	Value	Step	CentQ+Force	Value	Step
Multiple	**0.811**	**21.06**	Multiple	**0.867**	**14.37**
Single	0.757	32.80	Single	0.743	36.81
No options	0.0	200.0	No options	0.0	200.0

Table 4: Numeric results on the two-room task and two-agent four-room task

Four-agent four-room task			Six-agent four-room task			Eight-agent four-room task		
IQL	Value	Step	IQL	Value	Step	IQL	Value	Step
Multiple	**0.774**	**30.82**	Multiple	**0.329**	**126.0**	Multiple	0.0	200.0
Single	0.427	105.5	Single	0.0	200.0	Single	0.0	200.0
No options	0.095	179.0	No options	0.0	200.0	No options	0.0	200.0
CentQ+Force	Value	Step	CentQ+Force	Value	Step	CentQ+Force	Value	Step
Multiple	**0.842**	**17.96**	Multiple	**0.753**	**31.82**	Multiple	**0.605**	**64.90**
Single	0.447	102.3	Single	0.0	200.0	Single	0.0	200.0
No options	0.0	200.0	No options	0.0	200.0	No options	0.0	200.0

Table 5: Numeric results on the *n*-agent four-room tasks with random grouping

Four-agent four-room task			Six-agent four-room task			Eight-agent four-room task		
CentQ+Force	Value	Step	CentQ+Force	Value	Step	CentQ+Force	Value	Step
Multiple	**0.835**	**18.90**	Multiple	**0.784**	**27.40**	Multiple	**0.717**	**39.52**
Single	0.351	122.3	Single	0.0	200.0	Single	0.0	200.0
No options	0.0	200.0	No options	0.0	200.0	No options	0.0	200.0

Table 6: Numeric results on the *n*-agent four-room tasks with random grouping and dynamic influence among the agents