Learning Multi-agent Skills for Tabular Reinforcement Learning using Factor Graphs

Jiayu Chen, Jingdi Chen, Tian Lan, and Vaneet Aggarwal

***Abstract*—Covering skill (a.k.a., option) discovery has been developed to improve the exploration of reinforcement learning in single-agent scenarios, where only sparse reward signals are available. It aims to connect the most distant states identified through the Fiedler vector of the state transition graph. However, the approach cannot be directly extended to multi-agent scenarios, since the joint state space grows exponentially with the number of agents thus prohibiting efficient option computation. Existing research adopting options in multi-agent scenarios still relies on single-agent algorithms and fails to directly discover joint options that can improve the connectivity of the joint state space.**

In this paper, we propose a new algorithm to directly compute multi-agent options with collaborative exploratory behaviors while still enjoying the ease of decomposition. Our key idea is to approximate the joint state space as the Kronecker product of individual agents' state spaces, 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 of the estimated joint Fiedler vector. Evaluation on multi-agent collaborative tasks shows that our 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.

***Impact Statement*—Multi-agent reinforcement learning (MARL) has become increasingly important due to growing complexity of real-world decision making problems. A key performance bottleneck for MARL is the lack of efficient coordinated exploration among multiple agents. The proposed multi-agent option discovery approach addresses this problem by alleviating the exponential complexity involved in multi-agent explorations. The approach achieves significantly improved exploration and higher cumulative rewards in challenging multi-agent decision making scenarios.**

***Index Terms*—Multi-agent Reinforcement Learning, Skill Discovery, Kronecker Product**

I. Introduction

REINFORCEMENT Learning (RL) has achieved impressive performance in a variety of scenarios, such as robotic control [1], [2] and games [3]–[5]. However, most of its applications rely on carefully-crafted, task-specific reward signals to drive exploration and learning, limiting its use in real-life scenarios often with sparse or no rewards. To this end, acquiring skills from the experience in a task-agnostic manner by extracting temporal action-sequence abstractions, i.e., option discovery [6], to support efficient exploration can be essential. The acquired skills/options can then be employed by a meta-controller to solve downstream tasks more effectively. For instance, in a robotic navigation task, the robot can first learn locomotion skills in the environment, and then an agent only needs to learn a controller to give out point-to-point navigation commands which would be implemented through these skills. Thus, given useful skills, the downstream task can be greatly simplified from a continuous control task to a discrete one. Among recent developments on option discovery, *Covering Option Discovery* [7], [8] has been shown to be effective to accelerate the exploration in sparse reward environments. In particular, it first computes the second smallest eigenvalue and the corresponding eigenvector (i.e., Fiedler vector [9]) 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 in the Fiedler vector, which has been proven to greedily improve the algebraic connectivity of the state space [10]. With these options, the accessibility from each state to the others will be enhanced, due to which the exploration in the state space can be accelerated a lot.

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 [11]–[15] 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 about 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.

Jiayu Chen and Vaneet Aggarwal are with the School of Industrial Engineering, Purdue University, West Lafayette IN 47907, USA, email: {chen3686, vaneet}@purdue.edu. V. Aggarwal is also with Computer Science, King Abdullah University Of Science And Technology, Thuwal 23955, Saudi Arabia. Jingdi Chen and Tian Lan are with the Department of Electrical and Computer Engineering, George Washington University, Washinton DC 20052, USA, email:{jingdic, tlan}@gwu.edu.

This work was supported in part by Meta Platforms, Inc., Cisco Systems, Inc., and the U.S. National Science Foundation under grant 2114415.

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 in either a decentralized or centralized manner and present the comparisons between these two approaches. 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.

II. Related Work

Option Discovery: Temporal abstraction allows representing knowledge about courses of action at different time scales, which is key to scaling up learning and planning in RL. The temporal abstraction in RL can be modeled with the option framework proposed in [6], which extends the usual notion of actions to include options — the closed-loop policies for taking actions over a period of time. While planning with options is well understood in researches about Semi-MDP [16], [17] and Hierarchical Reinforcement Learning [18], [19], constructing options autonomously from data, i.e., option discovery, has remained challenging. Literature on option discovery is summarized as follows.

Some works, such as [20]–[23], are based on task-related reward signals. 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 (i.e., termination states of the options) based on the visitation frequency of the states. For example, in [24]–[26], 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, which are denoted 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 [7], [8], [27], [28], 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/connectivity from one state to another. Through adding options between states with poor connectivity, the exploration in the state space can be accelerated a lot. Note that all the approaches mentioned above are for single-agent scenarios, and in this paper we will extend the construction and adoption of options to MARL.

Adopting options in multi-agent scenarios: Current researches on adopting options in MARL, such as [11]–[15], [29], try 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 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 use: the algorithms in [11], [12] directly define the options based on their task without the learning process; the algorithms in [13]–[15] learn the options based on the task-related reward signals from the environment; the algorithm in [29] trains the options based on a reward function that is a weighted sum of the environment reward and information-theoretic reward proposed in [30].

To the best of our knowledge, we are the first to propose multi-agent covering option discovery. Specifically, we propose algorithms for directly constructing multi-agent options based on Laplacian spectrum of the individual state transition graphs to encourage efficient exploration in the joint state space, and explore how to utilize the multi-agent options in MARL effectively, so as to leverage the coordination among the agents in both the option discovery and adoption process.

III. Background

A. Basic Conceptions and Notations

In this section, we will introduce the necessary conceptions and corresponding notations used in this paper. We provide a table of predefined symbols in Appendix A.

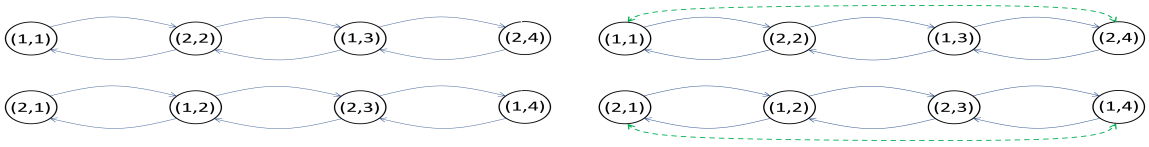
Markov Decision Process (MDP): The RL problem can be described with an MDP, denoted by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S}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

Remark 1. *There is an edge between state s and s' (i.e., s and s' are adjacent) if and only if $\exists\ a \in \mathcal{A},\ s.t.\ \mathcal{P}(s, a, s') > 0$ OR $\mathcal{P}(s', a, s) > 0$.*

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. $|\mathcal{S}|$ denotes the cardinality of $\mathcal{S}$. 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 eigenvector is called the Fiedler vector [9]. Further, the normalized Laplacian matrix is defined as $\mathcal{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$.

Kronecker product of graphs [31]: 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) The joint state transition graph of agent 1 and agent 2 (b) The joint state transition graph after adding individual options

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

Remark 2. *Two vertices of $G_1 \otimes G_2$, namely (g, h) and (g', h'), are adjacent if and only if g and g' are adjacent in G_1 and h and h' are adjacent in G_2.*

Thus, the Kronecker Product Graph can capture the joint transitions of the agents in their joint state space very well. In Section IV-B, we propose to use the Kronecker Product Graph as an effective approximation of the joint state transition graph, so that we can discover the joint options based on the factor graphs. Further, $A_1 \otimes A_2$ 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).

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

B. Covering Option Discovery

As defined in [6], an option ω consists of three components: an intra-option policy $\pi_\omega : \mathcal{S} \times \mathcal{A} \to [0, 1]$, 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 according to β_ω (i.e., $\beta_\omega = 1$). In order to get an option, we need to learn the intra-option policy, define the termination condition and initiation set.

The authors of [8] 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:

$$\begin{aligned} \omega_{ij} &= < I_{\omega_{ij}} = \{s_i\},\ \pi_{\omega_{ij}},\ \beta_{\omega_{ij}} = \{s_j\} > \\ \omega_{ji} &= < I_{\omega_{ji}} = \{s_j\},\ \pi_{\omega_{ji}},\ \beta_{\omega_{ji}} = \{s_i\} > \end{aligned} \tag{2}$$

to connect these two subgoal 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 is as follows.

The authors of [10] 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). Based on that, they propose a greedy heuristic to improve the algebraic connectivity of a graph: adding a certain number of edges one at a time, and each time connecting (s_i, s_j) corresponding to the largest $(F_i - F_j)^2$. Thus, applying this greedy heuristic to the state transition graph can effectively improve its connectivity, leading to a smaller upper bound of the expected cover time and accelerated exploration of the state space, as shown in [8].

IV. Proposed Algorithm

A. 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, $\mathcal{S}_i$ and $\mathcal{A}_i$ being 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 n. Thus, it is prohibitive to directly compute the covering options based on the joint state transition graph using the approach introduced in Section III-B 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 (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 connectivity of the joint state space may not be improved with these single-agent options. We illustrate this with a simple example.

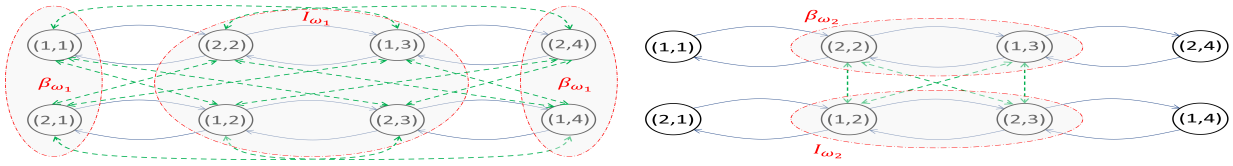
Illustrative example: Figure 1(a) shows a joint state transition graph $\widetilde{G}$ of two agents, where agent 1 has two states $\mathcal{S}_1 = \{1, 2\}$ and agent 2 has four states $\mathcal{S}_2 = \{1, 2, 3, 4\}$. In order to compute the individual options, we can restrict our attention to the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

$$L_1 = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \ L_2 = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}. \tag{3}$$

To compute the options for each agent, we first compute the Fiedler vectors of G_1 and G_2 (i.e., the eigenvectors corresponding to the second smallest eigenvalues of L_1 and L_2), namely F_1 and F_2:

$$F_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \ F_2 = \frac{1}{\sqrt{8 - 4\sqrt{2}}} \begin{bmatrix} -1 \\ -\sqrt{2} + 1 \\ \sqrt{2} - 1 \\ 1 \end{bmatrix}. \tag{4}$$

Then, according to the option discovery approach described in Section III-B, we can get the individual options for agent 1 to connect its state 1 (minimum) and state 2 (maximum), and



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

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

Fig. 2. The joint state transition graph updated with the detected multi-agent options

individual options for agent 2 to connect its state 1 (minimum) and state 4 (maximum). With these options, the joint transition from $(1,1)$ to $(2,4)$ (i.e., $(1,1) \rightarrow (2,4)$: agent 1 going from 1 to 2 and agent 2 going from 1 to 4) is possible, so are the transitions: $(2,4) \rightarrow (1,1)$ and $(2,1) \leftrightarrow (1,4)$. The newly updated transitions are shown as the green dashed lines in Figure 1(b). These options fail to create a connected graph. It implies that utilizing the single-agent options alone may not be sufficient for encouraging efficient joint exploration.

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. 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, $\beta_\omega : \widetilde{\mathcal{S}} \rightarrow \{0,1\}$ indicates the joint states to terminate, $\pi_\omega = (\pi_\omega^1, \cdots, \pi_\omega^n)(\pi_\omega^i : \mathcal{S}_i \times \mathcal{A}_i \rightarrow [0,1])$, 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 III-B. Given that $|\widetilde{\mathcal{S}}|$ grows exponentially with n, we propose to estimate the joint Fiedler vector based on the individual state spaces in the next section.

B. Theory results

We propose to use the Kronecker Product Graph to decompose the eigenfunction calculation to single-agent state spaces, making our approach much more scalable. This decomposition is based on the facts: (1) the Kronecker product of individual state transition graphs $\otimes_{i=1}^n G_i = G_1 \otimes \cdots \otimes G_n$ provides a good approximation of the joint state transition graph $\widetilde{G}$; (2) The Fielder vector of $\otimes_{i=1}^n G_i$ can be estimated with the Laplacian spectrum of $G_i (i = 1, \cdots, n)$.

We note that the use of $\otimes_{i=1}^n G_i$ as a factorized approximation of $\widetilde{G}$ introduces noise, since $\widetilde{G} = \otimes_{i=1}^n G_i$ becomes exact only in the case where agents' transitions are not influenced by the others. However, for the purpose of option discovery, we only need to identify areas in the state space with relatively low or high values in the Fielder vector, so an exact calculation of $\widetilde{G}$ and its Fiedler vector is not necessary. Moreover, the state transition influence among agents, e.g., collisions and blocking, would most likely result in local perturbations of the transition graph and thus is inconsequential to global properties of $\widetilde{G}$, like its algebraic connectivity and Fiedler vector. Therefore, approximating $\widetilde{G}$ by $\otimes_{i=1}^n G_i$ allows efficient options discovery. Further, in Section V-B, we empirically show in Figure 10 that superior exploration can still be achieved under such approximation noise, numerically validating the robustness of our proposed approach to the approximation error. Moreover, we provide a quantitative study on the approximation error in Section V-B, showing that $\otimes_{i=1}^n G_i$ can be used as a simple yet powerful approximation of $\widetilde{G}$ for option discovery.

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 [32] which proposed an estimation of the Laplacian spectrum of the Kronecker product of two factor graphs, we have the following THEOREM 1.

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

$$\lambda_{k_1,\ldots,k_n} = \left\{ \left[1 - \prod_{i=1}^{n} (1 - \lambda_{k_i}^{G_i}) \right] \prod_{i=1}^{n} d_{k_i}^{G_i} \right\} \tag{5}$$

$$v_{k_1,\ldots,k_n} = \otimes_{i=1}^n v_{k_i}^{G_i} \tag{6}$$

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 1 is provided in Appendix B. Through enumerating $(k_1, \cdots, k_n)$, we can collect the eigenvalues of $\otimes_{i=1}^n G_i$ by Equation (5) and the corresponding eigenvectors by Equation (6). Then, the eigenvector $v_{\hat{k}_1,\cdots,\hat{k}_n}$ corresponding to the second smallest eigenvalue $\lambda_{\hat{k}_1,\cdots,\hat{k}_n}$ is the estimated Fiedler vector of the joint state transition graph, namely $F_{\widetilde{G}}$. Based on it, 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 III-B, connecting these two joint states with options can greedily improve the algebraic connectivity 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 1. As a result, we get two approximations of the Fiedler vector: (Please refer to Appendix C for computing details.)

$$F_{\widetilde{G}}^1 = \frac{1}{\sqrt{6}} \left[\frac{1}{\sqrt{2}},\ 1,\ 1,\ \frac{1}{\sqrt{2}},\ \frac{1}{\sqrt{2}},\ 1,\ 1,\ \frac{1}{\sqrt{2}} \right]^T \tag{7}$$

$$F_{\widetilde{G}}^2 = \frac{1}{\sqrt{6}} \left[-\frac{1}{\sqrt{2}},\ 1,\ -1,\ \frac{1}{\sqrt{2}},\ \frac{1}{\sqrt{2}},\ -1,\ 1,\ -\frac{1}{\sqrt{2}} \right]^T \tag{8}$$

Based on the two approximations and the indexing relationship between $\widetilde{G}$ and its factor graphs (Equation (1)), we can get

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 Ω
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 $F_{\widetilde{G}}$ of the joint state space using THEOREM 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 $F_{\widetilde{G}}$, 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 state space of agent i
11: **Generate** a new list of options Ω' 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**

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 Ω'
3: $\Omega' \leftarrow \emptyset$
4: **for** $s = (s_1, \cdots, s_n)$ in $(MIN \cup MAX)$ **do**
5: **Define** the initiation set I_ω 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 π_ω^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**

Algorithm 3 Update Adjacency Matrices

1: **Given**: list of adjacency matrices $A_{1:n}$, MIN, MAX
2: **for** $s_{min} = (s_{min}^1, \cdots, s_{min}^n)$ in MIN **do**
3: **for** $s_{max} = (s_{max}^1, \cdots, s_{max}^n)$ in MAX **do**
4: **for** $i = 1$ to n **do**
5: $A_i[ind(s_{min}^i)][ind(s_{max}^i)] = 1$
6: $A_i[ind(s_{max}^i)][ind(s_{min}^i)] = 1$
7: **end for**
8: **end for**
9: **end for**

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 and termination states respectively. For example, in $F_{\widetilde{G}}^1$ (Equation (7)), 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), i.e., $7 = (2-1) \times |\mathcal{S}_2| + 3$ where $|\mathcal{S}_2| = 4$. As shown in Figure 2, the green dashed lines represent the joint options which connect the states in the initiation and termination set bidirectionally. It can be observed that both of the two options can lead to a connected graph when applied to $\widetilde{G}$. Thus, the adoption of multi-agent options has the potential to 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 spaces, so that we can enjoy the ease of decomposition. Next, we will formalize our algorithm.

C. 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 are 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 $F_{\widetilde{G}}$ of the joint state space through THEOREM 1 based on $A_{1:n}$, so that we can collect the joint states corresponding to the minimum or maximum of $F_{\widetilde{G}}$. 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 $((s_{min}^1, \cdots, s_{min}^n), (s_{max}^1, \cdots, s_{max}^n))$, so that we can connect (s_{min}, s_{max}) in the joint state space by connecting each (s_{min}^i, s_{max}^i) in the corresponding individual state space.

After decentralizing the joint states, we can define the multi-agent options as follows. For each option ω, we define I_ω as the explored joint states, and β_ω as a joint state in $MIN \cup MAX$ or the unexplored area. 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}), e.g., setting $\{s_{min}\}$ as I_ω and $\{s_{max}\}$ as β_ω, we extend I_ω to the known area to increase the accessibility of ω. As for the intra-option policy π_ω used for connecting the initiation and termination joint state, we divide it into a list of single-agent policies π_ω^i $(i = 1, \cdots, n)$, where π_ω^i can be trained with any single-agent RL algorithm aiming at leading agent i from its own initiation state to the termination state s_{min}^i (s_{max}^i). 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.

At last, we give out the computational complexity of our approach. Consider an MDP with n agents and r states for each agent. To compute the Fiedler vector directly from the joint state transition graph would require time complexity $\mathcal{O}(r^{3n})$, since there are in total r^n joint states and the time complexity of eigenvalue decomposition (Line 7 in Algorithm 1) is cubic with the size of the joint state space. While, with our Kronecker factor graph approach, we can decompose the original problem into computing eigenvectors of the individual state transition graphs, of which the overall time complexity is $\mathcal{O}(nr^3)$. Thus, our solution significantly reduces the problem complexity from $\mathcal{O}(r^{3n})$ to $\mathcal{O}(nr^3)$ for multi-agent problems. Also, note that for problems with continuous or large state space (i.e., r is large), our approach could be directly integrated with sample-based techniques for eigenfunction estimation (Line 7 in Algorithm 1), like [33], [34]. Hence, the bottleneck on computational complexity can be overcome. More precisely, the Laplacian spectrum of the factor graphs can be estimated using neural networks, and then leveraged by our proposed algorithm to find the Fiedler vector of the joint state transition graph, which is considered as future work.

D. 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. The agent does not decide on a new option with the high-level policy until the current option terminates.

For a multi-agent option $\omega : < I_\omega = \{the\ explored\ joint\ states\},\ \pi_\omega = (\pi_\omega^1, \cdots, \pi_\omega^n), \beta_\omega = \{(s_1, \cdots, s_n)\} >$, it can

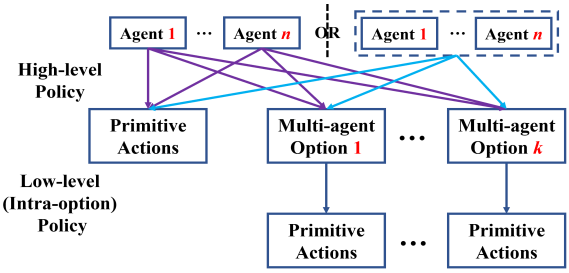
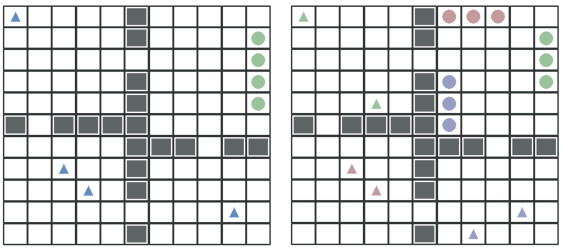


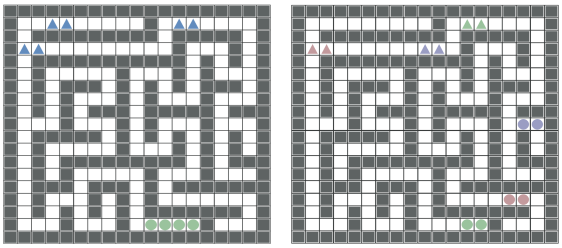
Fig. 3. Hierarchical algorithm framework: When making decisions, the agent first decides on which option ω to use according to the high-level policy, and then decides on the primitive action to take based on the corresponding intra-option policy π_ω. The agents can decide on their options independently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

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. 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. Note that if there are j primitive actions and k multi-agent options, the size of the search space would be $(j+k)^n$ for the decentralized approach and $j^n + k$ for the centralized approach. Therefore, the decentralized manner 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 reach the sub-goal. In this paper, we use Independent Q-Learning [35] (adopting Q-Learning [36] to each individual agent) to train the decentralized high-level policy, and Centralised Q-Learning (viewing the n agents as a whole and adopting Q-Learning to it) to train the centralised high-level policy. We present comparisons in Section V.

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 (i.e., joint states) will grow exponentially with n. Thus, we propose to partition the agents into some sub-groups first, and then learn the joint options within each sub-group. The intuition behind this is as follows. 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. Further, 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 V, we show that the adoption of grouping techniques can not only



(a) n-agent four-room task (b) $m \times n$-agent four-room task



(c) n-agent maze task (d) $m \times n$-agent maze task

Fig. 4. Simulators for Evaluation: All the agents (triangles) must reach the goal area (circles) simultaneously to complete the task, based on only their current locations. In (b) and (d), agents are assigned with different goals. The agents and their corresponding goals are labeled with the same color.

accelerate the exploration but also can greatly improve the scalability of our algorithm.

V. Evaluation and Results

A. Simulation Setup

As shown in Figure 4, the proposed approach is evaluated on four multi-agent goal-achieving tasks. (1) For tasks shown as Figure 4(a) and 4(c), n (2∼8) agents agents (triangles) must reach the goal area (circles) at the same time to complete this task, without going through the walls (squares). If some agents have reached the goal and the others haven't, the n agents will continue to move until all of them reach the goal in the meantime. (2) For tasks shown as Figure 4(b) and 4(d), there are m groups of agents, and each group contains n agents. Each group of agents has a special goal area labeled with the same color. Note that all the $m \times n$ agents should get to their related goal areas at the same time to complete this task, and the agents don't know which goal area is related to them at first. We only show the 4-agent and 3×2-agent cases for the four-room task in Figure 4. For the illustrations of other cases used in the following evaluations, please refer to Appendix D. For all the four tasks, different agents can share the same grid, and only when the agents complete the task can they receive a reward signal $r = 1.0$, which is shared by all the agents; otherwise, they will receive $r = 0.0$. In the following experiments, we use the episodic cumulative reward as the metric, which is defined as: $\sum_{i=0}^{l} \lambda^i r$. $\lambda = 0.99$ is the discount factor of the MDP, and l is the horizon of each episode of which the maximum is set as 200.

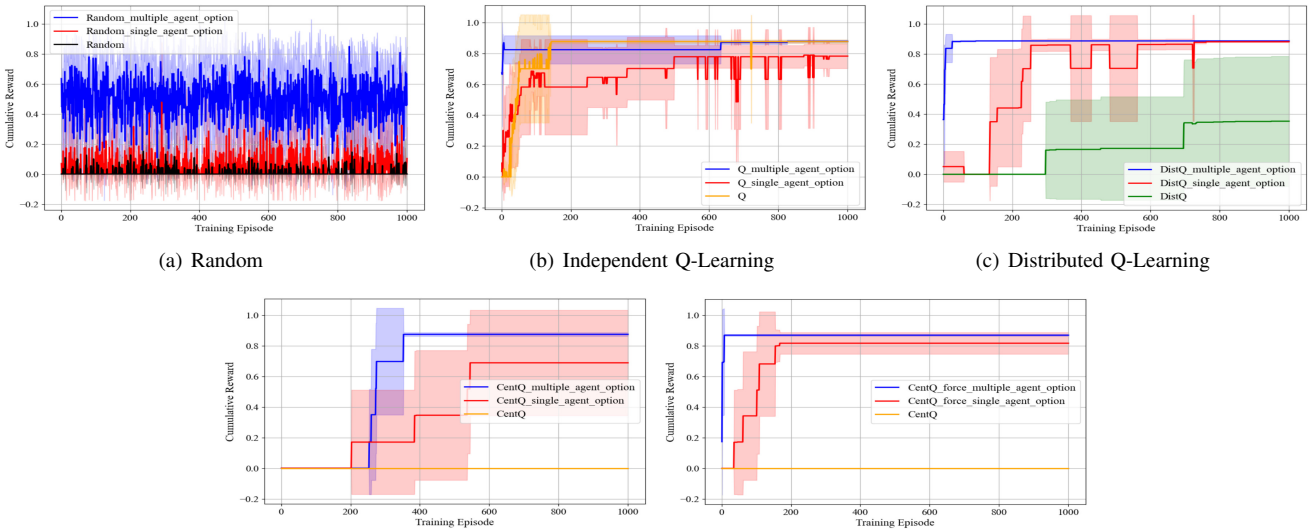
Note that these evaluation tasks are multi-agent versions of the simulators used in [8] (i.e., one of our baselines) which are quite challenging. On one hand, the agents need to make decisions based on only their current locations (i.e., without knowing where the goal area is). On the other hand, the reward space is very sparse: for example, in the eight-agent four-room task, there are in total 104 states (i.e., non-wall grids) for each agent and 4 of them are the rewarding states (i.e., goals), so the ratio of the rewarding joint states is $(4/104)^8 \approx 4.8 \times 10^{-12}$, which is also the probability that the 8 agents can complete this task through the random walk [38]. Hence, agents without highly-efficient exploration strategies cannot complete these tasks. In Section V-B, we evaluate on tasks of increasing complexity (e.g., Figure 6 and 8). The more difficult the task is, the more advantageous our approach becomes.

We compare our approach – agents with multi-agent options, with two baselines: (1) Agents without options: the high-level policy is directly used to choose primitive actions, rather than choosing the option first and then choosing the primitive action with the corresponding intra-option policy. Comparisons with this baseline can show the effectiveness of using options to aid the exploration. (2) Recent works on adopting options in MARL: As mentioned in Section II, these works [11]–[15], [29] first construct single-agent options for each agent based on their individual state spaces and then learn to collaboratively utilize them in MARL, so we denote these methods as agents with single-agent options in the following. However, they either rely on predefined options or adopt the option discovery methods that depend on dense task-related reward signals and suffer from poor performance in environments with only sparse rewards like ours. In this case, we adopt the state-of-the-art algorithm proposed in [8] to replace the option discovery algorithm component in these methods, which claims to outperform previous option discovery algorithms, including [26], [27], [30], for sparse reward scenarios. Comparisons with this baseline can show the superiority of our approach to directly identify and adopt joint options in multi-agent scenarios. To be fair, we set the number of single-agent and multi-agent options for each agent to select as the same. 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.

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. In the following experiments, we evaluate the performance of agents with five different algorithms as the high-level policy: random policy, Independent Q-Learning [35], Distributed Q-Learning [37] (each agent decides on their own option based on the joint state), Centralized Q-Learning and Centralized Q-Learning + Force, to make sure that the performance improvement is not specific to a certain algorithm. Table I shows the comparisons among these algorithms: (1) If adopting "Independent Q-Learning" as the high-level policy, agents need to make decisions based on only their local states; otherwise, agents within the same sub-group can share their views and make decisions based on their joint states. (2) For "Centralized Q-Learning + Force", agents are forced to choose the same multi-agent option at a time (centralized); while, for the others, agents can choose different options to execute simultaneously (decentralized). As for the low-level policy, we adopt Value Iteration [39] to find the optimal path between each pair of initiation and termination state for each agent i as π_ω^i. Compared with Baseline (2), our approach doesn't cost additionally for learning the low-level

Table I. Comparisons among different high-level policy algorithms

Algorithm	Input	Output	How to utilize multi-agent options
Random	–	Individual action	Decentralized
Independent Q-Learning [35]	Individual state	Individual action	Decentralized
Distributed Q-Learning [37]	Joint state	Individual action	Decentralized
Centralized Q-Learning	Joint state	Joint action	Decentralized
Centralized Q-Learning + Force	Joint state	Joint action	Centralized



(a) Random (b) Independent Q-Learning (c) Distributed Q-Learning

(d) Centralized Q-Learning (e) Centralized Q-Learning + Force

Fig. 5. 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, our approach converges to a higher cumulative reward.

policy, since the number of single-agent and multi-agent options are the same for each agent.

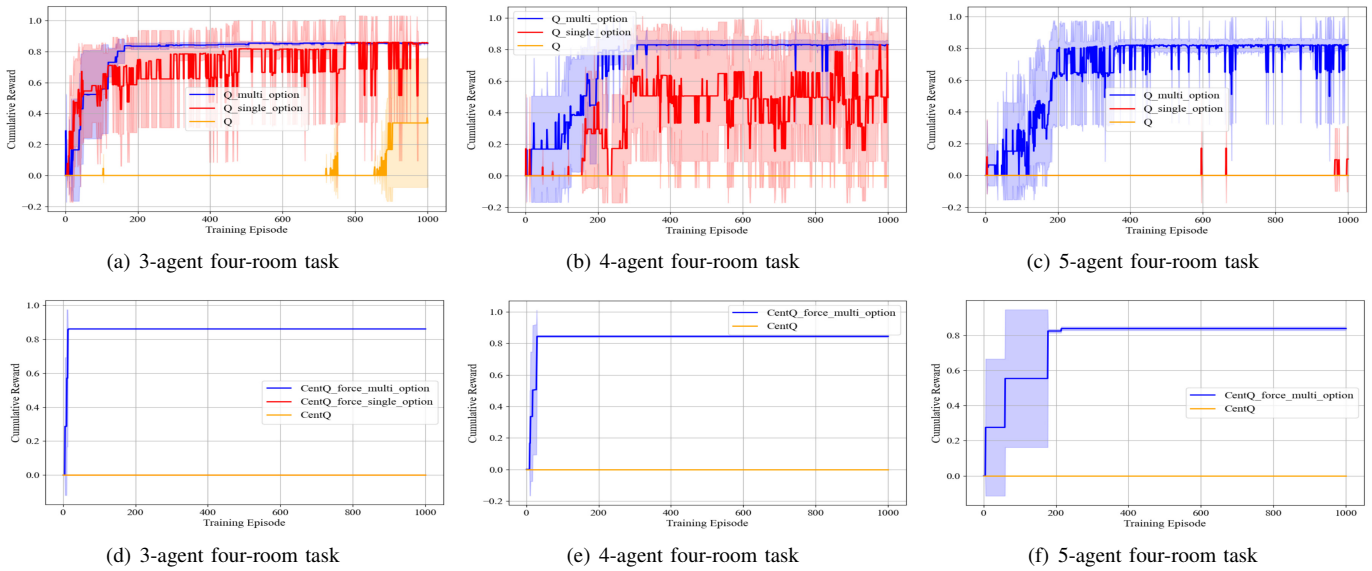
B. Main Results

For each experiment, we present comparisons among the performance of agents with multi-agent options (blue line), agents with single-agent options (red line) and agents without options. We run each experiment five times with different random seeds and plot the change of the mean (the solid line) and standard deviation (the shadow area) of the episodic cumulative reward during the training process (1000 episodes).

Two-agent four-room task: As shown in Figure 5, we present comparisons on the two-agent four-room task, with different algorithms (listed in Table I) as the high-level policy. It can be observed that 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. 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 6(a)-6(c), we test these methods on n-agent four-room tasks ($n = 3 \sim 5$), 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. Further, in Figure 6(d)-6(f), 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. As mentioned in Section IV-D, the size of the joint output space is $(j+k)^n$ for the decentralized manner and $j^n + k$ for the centralized manner if there are j primitive actions and k options for the n agents to select. 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 6(e)-6(f), 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 6(d).

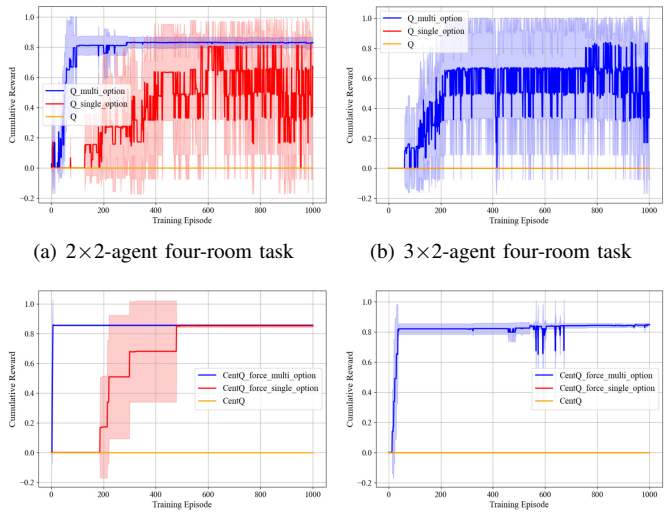
Four-room task with sub-task grouping: The size of the joint state space grows exponentially with the number of agents, making it infeasible to directly construct n-agent options and adopt Centralized Q-Learning for a large n. However, in real-life scenarios, a multi-agent task can usually be divided into



(a) 3-agent four-room task (b) 4-agent four-room task (c) 5-agent four-room task

(d) 3-agent four-room task (e) 4-agent four-room task (f) 5-agent four-room task

Fig. 6. Evaluation on n-agent four-room tasks: (a)-(c) 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. (d)-(f) using Centralized Q-Learning + Force as the high-level policy. Agents with single-agent options start to fail since the 3-agent case. Also, it can be observed that the centralized way to utilize the n-agent options leads to faster convergence.



(a) 2×2-agent four-room task (b) 3×2-agent four-room task

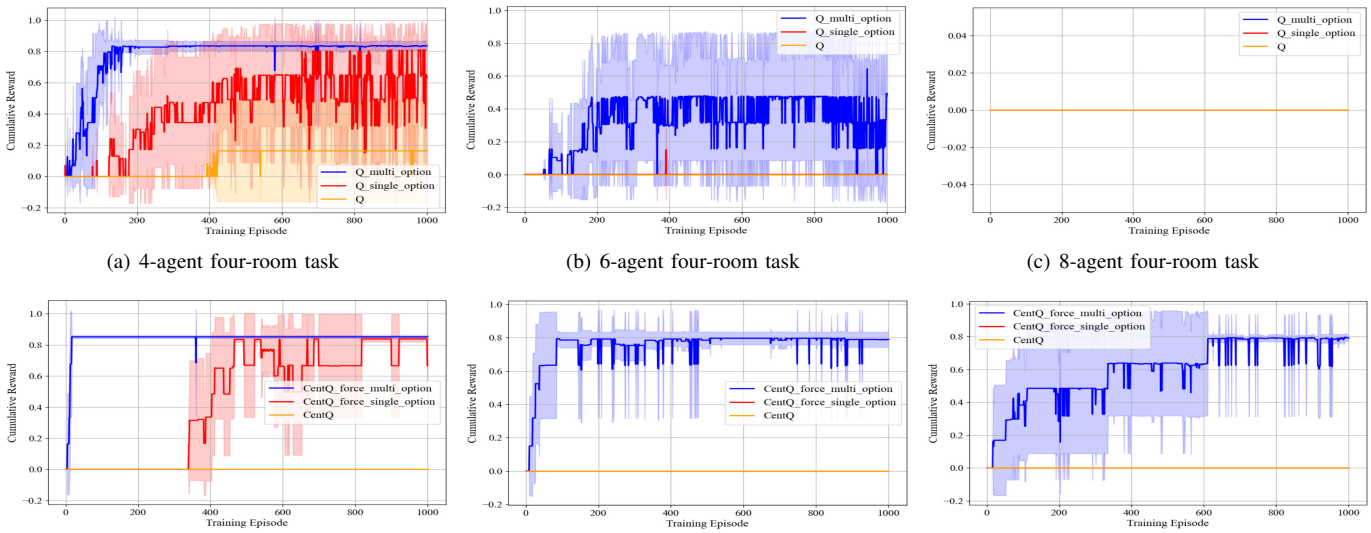
(c) 2×2-agent four-room task (d) 3×2-agent four-room task

Fig. 7. Comparisons on the $m \times n$ four-room tasks with sub-task grouping: (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.

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(b), where we divide the agents into m sub-groups, each of which contains n agents with the same goal area. Figure 7 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. Note that the red line is covered by the yellow line in Figure 7(d). Also, we see that agents trained with Centralized Q-Learning + Force (Figure 7(c)-7(d)) have faster convergence speed and higher convergence value.

Four-room task with random grouping: 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 8, 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 8(c). However, if we adopt Centralized Q-Learning + Force, agents with pairwise options can still complete this challenging task with satisfaction, shown as Figure 8(f).

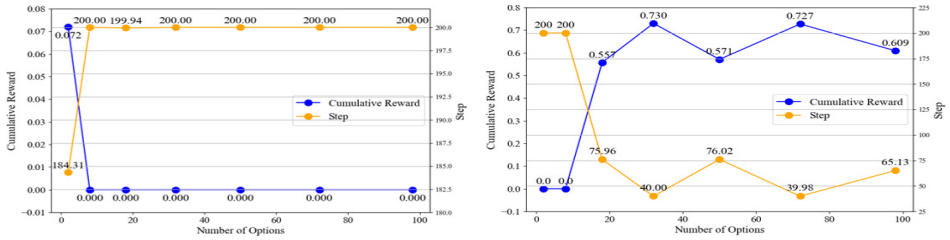
Further, in Figure 9, we show how the performance of agents using pairwise options would change with the number of options, based on the eight-agent four-room tasks (the orange line: number of steps to complete the task; the blue line: episodic cumulative reward). Note that for each step, every agent will make a decision to move one grid in any of the four directions (i.e., up, down, left, or right), and the maximum of the decision steps for each episode is 200. When increasing the number of options, the performance of agents



(a) 4-agent four-room task (b) 6-agent four-room task (c) 8-agent four-room task

(d) 4-agent four-room task (e) 6-agent four-room task (f) 8-agent four-room task

Fig. 8. Comparisons on the n-agent four-room tasks with random grouping: (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.



(a) Independent Q-Learning (b) Centralized Q-Learning + Force

Fig. 9. Performance change of the agents as the number of options increase, evaluated on the eight-agent four-room task with random grouping. As the number of options increases, the performance of agents with Centralized Q-Learning + Force as the high-level policy can be improved further. While, if for Independent Q-Learning, the agents' performance would go worse.

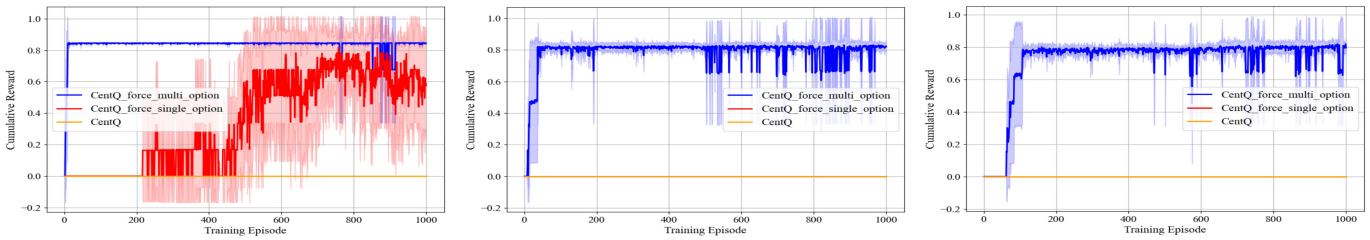
with pairwise options and using Centralized Q-Learning + Force as the high-level policy can be improved further. While, if using the Independent Q-Learning as the high-level policy, the agents' performance would go worse as the number of options increases. The reason is that, as mentioned in Section IV-D, 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 multi-agent options in a centralized manner.

Four-room task with random grouping and dynamic influences among agents: Further, 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 1. For this new setting, we make some modifications based on the n-agent four-room task (Figure 4(a)) – different agents cannot share the same grid so that an agent may be blocked by others when moving ahead, and this influence is highly dynamic. We use the Centralized Q-Learning + Force as the high-level policy, of which the results are shown as Figure 10. We can see that although this modification affects the performance of agents with single-agent options, we can still get significant performance improvement with pairwise options discovered with Theorem 1.

Table II. Quality of the estimated joint transition graph

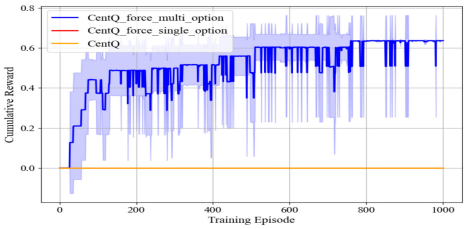
α	0.3	0.5	0.7
$\widehat{\lambda_2}$ $(\times 10^{-3})$	8.1131	8.1131	8.1131
λ_2 $(\times 10^{-3})$	8.1129	8.0988	8.0996
$\frac{\lvert\lambda_2 - \widehat{\lambda_2}\rvert}{\lambda_2}$ (%)	0.0025	0.1771	0.1662
$\frac{\lVert F - \widehat{F}\rVert_2}{\lVert F\rVert_2}$	0.0223	0.0989	0.1418

As mentioned in Section IV-B, the approximation error occurs when the state transitions of an agent are influenced by others. In Figure 10, we have evaluated on the case where an agent's state transitions will be influenced by others' states (i.e., blocking by other agents when going ahead). However, the transition influence for an agent may also come from the action choices of other agents. Thus, we further evaluate on a modified two-agent four-room task. We set agent 1 as the leading agent and agent 2 will follow the moving direction of agent 1 with the probability α, so the state transition of

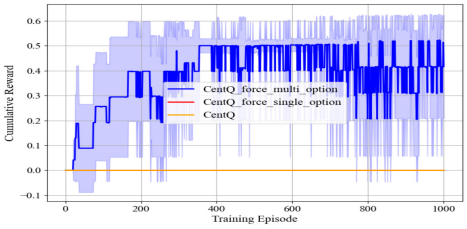


(a) 4-agent four-room task (b) 6-agent four-room task (c) 8-agent four-room task

Fig. 10. Comparisons on the n-agent four-room tasks with random grouping 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 IV-B and use them to get superior performance.



(a) 6-agent maze task



(b) 3×2-agent maze task

Fig. 11. Comparisons on the more challenging maze tasks using Centralized Q-Learning + Force as the high-level policy, where (a) and (b) are with random and sub-task grouping respectively. Although both baselines fail to complete the tasks, our approach can converge within 500 episodes with high rewards.

agent 2 can be influenced by the action choice of agent 1. With a certain α, we collect a million state transitions (i.e., (s, a, s')) through Monte Carlo sampling, based on which we can build the joint state transition graph $\widetilde{G}$ and the individual state transition graphs G_i $(i = 1, 2)$ and then get $\otimes_{i=1}^{2} G_i$. Then, as shown in Table II, we compare the algebraic connectivity and Fiedler vector of $\widetilde{G}$ (i.e., λ_2, F) and $\otimes_{i=1}^{2} G_i$ (i.e., $\widehat{\lambda_2}$, $\widehat{F}$) as α increases, which are closely related to the covering option discovery. We can see that the approximation error on these global properties of $\widetilde{G}$ caused by the transition influence among the agents is inconsequential. Thus, approximating $\widetilde{G}$ with $\otimes_{i=1}^{n} G_i$ allows accurate option discovery. There are in total 104^2 joint states which is also the size of F and $\widehat{F}$, and the complexity for the eigen decomposition is already $\mathcal{O}(10^{12})$ (i.e., $(104^2)^3$), so we limit the number of agents to 2.

Maze task with random grouping or sub-task grouping: Finally, in order to show the effectiveness of our approach on more challenging tasks, we compare it with the baselines on the maze tasks shown as Figure 4(c) and 4(d), of which the results are shown in Figure 11(a) and 11(b) respectively. Compared with the four-room task, the state space of the maze task is larger and the path finding toward the goal-area is much more difficult. Again, both baselines fail to complete the tasks, while our approach can converge within 500 episodes with a fairly high cumulative reward. Note that, for both tasks, we first group the agents based on sub-tasks (Figure 4(d)) or randomly (Figure 4(c)), then learn the pairwise options for each sub-group and utilize these options in a centralized manner to aid the exploration. To further show the difficulty of this task and significance of our algorithm, we apply SOTA MARL algorithms, including COMA [40], Weighted QMIX [41] and MAVEN [42], on the 6-agent maze task, each of which is repeated three times with different random seeds. The mean and standard deviation of the cumulative rewards in the training process (50000 episodes) of these baselines are shown in Table III, showing that none of these algorithms can learn to complete this task. The reason is that as a challenging cooperative search problem, the reward space is highly sparse since only when the six agents arrive at the goal area at the same time can they receive the reward signal, so efficient exploration strategies in the joint state space like ours is required. The code for reproducibility of these results has been made available at [43].

Table III. Performance of SOTA MARL baselines

Algorithm	Mean	Std.
COMA [40]	0.0	0.0
CWQMIX [41]	0.0	0.0
OWQMIX [41]	0.0	0.0
MAVEN [42]	0.0	0.0

VI. 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 options in MARL, using centralized, decentralized, and group-based strategies, respectively. We empirically show that agents with multi-agent options have significantly superior performance than agents relying on single-agent options or no options.

A future direction would be to scale our algorithm for real-life applications with SOTA representation learning techniques, like [33], [34]. On the other hand, there will be non-negligible differences between $\otimes_{i=1}^{n} G_i$ and the joint state transition graph $\widetilde{G}$, if the state transitions of an agent are hugely influenced by the others. Therefore, mechanisms to detect these situations in a task scenario and integrate them with $\otimes_{i=1}^{n} G_i$ for a better approximation of $\widetilde{G}$ will also be an interesting future direction.

REFERENCES

[1] F. Ebert, C. Finn, S. Dasari, A. Xie, A. X. Lee, and S. Levine, "Visual foresight: Model-based deep reinforcement learning for vision-based robotic control," *CoRR*, vol. abs/1812.00568, 2018.

[2] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *Proceedings of the 4th International Conference on Learning Representations, ICLR 2016*, 2016.

[3] N. Brown and T. Sandholm, "Superhuman ai for multiplayer poker," *Science*, vol. 365, pp. 885–890, 2019.

[4] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. P. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, 2016.

[5] I. Hosu and T. Rebedea, "Playing atari games with deep reinforcement learning and human checkpoint replay," *CoRR*, vol. abs/1607.05077, 2016.

[6] R. S. Sutton, D. Precup, and S. P. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial Intelligence*, vol. 112, pp. 181–211, 1999.

[7] Y. Jinnai, D. Abel, D. E. Hershkowitz, M. L. Littman, and G. D. Konidaris, "Finding options that minimize planning time," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, vol. 97, 2019, pp. 3120–3129.

[8] 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, 2019, pp. 3130–3139.

[9] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematical Journal*, vol. 23, pp. 298–305, 1973.

[10] 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.

[11] 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 2014*. IFAAMAS/ACM, 2014, pp. 1273–1280.

[12] 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.

[13] J. Shen, G. Gu, and H. Liu, "Multi-agent hierarchical reinforcement learning by integrating options into maxq," in *Proceedings of the 1st International Multi-Symposiums on Computer and Computational Sciences, IMSCCS 2006*, vol. 1, 2006, pp. 676–682.

[14] 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 2020*. IFAAMAS/ACM, 2020, pp. 1792–1794.

[15] 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*, 2020.

[16] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 1994.

[17] R. Fruit and A. Lazaric, "Exploration-exploitation in mdps with options," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*, vol. 54, 2017, pp. 576–584.

[18] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, "Feudal networks for hierarchical reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*, ser. Proceedings of Machine Learning Research, vol. 70, 2017, pp. 3540–3549.

[19] S. Pateria, B. Subagdja, A. Tan, and C. Quek, "Hierarchical reinforcement learning: A comprehensive survey," *ACM Computing Surveys*, vol. 54, pp. 109:1–109:35, 2021.

[20] A. McGovern and A. G. Barto, "Automatic discovery of subgoals in reinforcement learning using diverse density," in *Proceedings of the 8th International Conference on Machine Learning ICML 2001*, 2001, pp. 361–368.

[21] I. Menache, S. Mannor, and N. Shimkin, "Q-cut - dynamic discovery of sub-goals in reinforcement learning," in *Proceedings of the 13th European Conference on Machine Learning, ECML 2002*. Springer, 2002, pp. 295–306.

[22] D. J. Mankowitz, T. A. Mann, and S. Mannor, "Adaptive skills adaptive partitions (ASAP)," in *Advances of the 30th Conference on Neural Information Processing Systems, NIPS 2016*, vol. 29, 2016, pp. 1588–1596.

[23] J. Harb, P. Bacon, M. Klissarov, and D. Precup, "When waiting is not an option: Learning options with a deliberation cost," in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence, AAAI 2018*. AAAI Press, 2018, pp. 3165–3172.

[24] M. Stolle and D. Precup, "Learning options in reinforcement learning," in *Proceedings of the 5th International Symposium on Abstraction, Reformulation and Approximation, SARA 2002*, vol. 2371. Springer, 2002, pp. 212–223.

[25] Ö. Simsek, A. P. Wolfe, and A. G. Barto, "Identifying useful subgoals in reinforcement learning by local graph partitioning," in *Proceedings of the 22nd International Conference on Machine Learning, ICML 2005*, vol. 119, 2005, pp. 816–823.

[26] Ö. Simsek and A. G. Barto, "Skill characterization based on betweenness," in *Proceedings of the 22nd Conference on Neural Information Processing Systems, NIPS 2008*, vol. 21, 2008, pp. 1497–1504.

[27] 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, 2017, pp. 2295–2304.

[28] M. C. Machado, C. Rosenbaum, X. Guo, M. Liu, G. Tesauro, and M. Campbell, "Eigenoption discovery through the deep successor representation," in *Proceedings of the 6th International Conference on Learning Representations, ICLR 2018*, 2018.

[29] 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 2020*. IFAAMAS/ACM, 2020, pp. 1566–1574.

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

[31] P. M. Weichsel, "The kronecker product of graphs," in *Proceedings of the 13th American Mathematical Society, AMS 1962*, vol. 13. JSTOR, 1962, pp. 47–52.

[32] M. Basic, B. Arsic, and Z. Obradovic, "Another estimation of laplacian spectrum of the kronecker product of graphs," *CoRR*, vol. abs/2102.02924, 2021.

[33] Y. Wu, G. Tucker, and O. Nachum, "The laplacian in RL: learning representations with efficient approximations," in *Proceedings of the 7th International Conference on Learning Representations, ICLR 2019*, 2019.

[34] K. Wang, K. Zhou, Q. Zhang, J. Shao, B. Hooi, and J. Feng, "Towards better laplacian representation in reinforcement learning with generalized graph drawing," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021*, vol. 139, 2021, pp. 11 003–11 012.

[35] M. Tan, "Multi-agent reinforcement learning: Independent versus cooperative agents," in *Proceedings of the 10th International Conference on Machine Learning, ICML 1993*, 1993, pp. 330–337.

[36] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[37] M. Lauer and M. A. Riedmiller, "An algorithm for distributed reinforcement learning in cooperative multi-agent systems," in *Proceedings of the 17th International Conference on Machine Learning, ICML 2000*, 2000, pp. 535–542.

[38] K. Pearson, "The problem of the random walk," *Nature*, vol. 72, pp. 342–342, 1905.

[39] R. S. Sutton and A. G. Barto, *Reinforcement learning - an introduction*, ser. Adaptive computation and machine learning. MIT Press, 1998.

[40] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence, AAAI 2018*. AAAI Press, 2018, pp. 2974–2982.

[41] T. Rashid, G. Farquhar, B. Peng, and S. Whiteson, "Weighted QMIX: expanding monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proceedings of the 34th Conference on Neural Information Processing Systems, NIPS 2020*, vol. 33, 2020.

[42] A. Mahajan, T. Rashid, M. Samvelyan, and S. Whiteson, "MAVEN: multi-agent variational exploration," in *Proceedings of the 33th Conference on Neural Information Processing Systems, NIPS 2019*, vol. 32, 2019, pp. 7611–7622.

[43] J. Chen, J. Chen, T. Lan, and V. Aggarwal, "Multi-agent option discovery based on kronecker product," https://github.itap.purdue.edu/Clan-labs/MAOD_via_KP, 2022.

[44] H. Sayama, "Estimation of laplacian spectra of direct and strong product graphs," *Discrete Applied Mathematics*, vol. 205, pp. 160–170, 2016.

[45] D. B. West *et al.*, *Introduction to graph theory*. Prentice hall Upper Saddle River, 2001, vol. 2.

APPENDIX A
NOTATIONS

Table IV. Glossary of the predefined symbols

Variable	Definition	Variable	Definition
$\mathcal{M}$	$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$: MDP	L	$L = D - A$: Laplacian matrix of G
$\mathcal{S}$	Individual state space	$\mathcal{L}$	$\mathcal{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$: Normalized Laplacian matrix of G
$\widetilde{\mathcal{S}}$	$\widetilde{\mathcal{S}} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$: Joint state space, which is the Cartesian product of the individual ones of n agents	λ_i, v_i	The i-th smallest eigenvalue and corresponding eigenvector of L
$\mathcal{A}$	Individual action space	λ_2	Algebraic connectivity of G
$\widetilde{\mathcal{A}}$	$\widetilde{\mathcal{A}} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$: Joint action space of the n agents	F	Fielder vector of G: the eigenvector corresponding to λ_2
$\mathcal{P}$	$\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$: State transition function	$\otimes_i G_i$	$\otimes_{i=1}^n G_i = G_1 \otimes \cdots \otimes G_n$: Kronecker Product of the state transition graphs of the n agents
γ	$\gamma \in (0, 1]$: Discount factor	ω	$\omega =< I_\omega, \pi_\omega, \beta_\omega >$: Option
G	$G = (V_G, E_G)$: Individual state transition graph, where V_G and E_G denote the set of vertices and undirected edges	I_ω	Initiation set of ω
$\widetilde{G}$	Joint state transition graph, which is defined on $\widetilde{\mathcal{S}}$ and related to the joint state transitions of the agents	π_ω	Intra-option policy of ω
A	Adjacency matrix of G	β_ω	Termination condition of ω
D	Degree matrix of G	MIN, MAX	Joint states corresponding to the minimum or maximum in F

APPENDIX B
PROOF OF THEOREM 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:

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

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

$$\begin{aligned} \mathcal{L}_{\otimes_{i=1}^n G_i} &= \otimes_{i=1}^n I_{K_i} - \otimes_{i=1}^n (D_{G_i}^{-\frac{1}{2}} G_i D_{G_i}^{-\frac{1}{2}}) \\ &= \otimes_{i=1}^n I_{K_i} - \otimes_{i=1}^n (I_{K_i} - \mathcal{L}_{G_i}). \end{aligned} \tag{10}$$

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

$$\begin{aligned} \mathcal{L}_{\otimes_{i=1}^n G_i} &= \otimes_{i=1}^n I_{K_i} - \otimes_{i=1}^n (V_{G_i} \Lambda_{G_i} V_{G_i}^T) \\ &= \otimes_{i=1}^n I_{K_i} - (\otimes_{i=1}^n V_{G_i})(\otimes_{i=1}^n \Lambda_{G_i})(\otimes_{i=1}^n V_{G_i})^T \\ &= (\otimes_{i=1}^n V_{G_i})(\otimes_{i=1}^n I_{K_i} - \otimes_{i=1}^n \Lambda_{G_i})(\otimes_{i=1}^n V_{G_i})^T, \end{aligned} \tag{11}$$

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 and testified in [32], [44]) that $D_{G_i}^{\frac{1}{2}} V_{G_i} \approx V_{G_i} D_{G_i}^{\frac{1}{2}}$, for $i = 1, 2, \ldots, n$, we can derive that:

$$\begin{aligned} L_{\otimes_{i=1}^n G_i}(\otimes_{i=1}^n V_{G_i}) &\approx D^{\frac{1}{2}} \mathcal{L}_{\otimes_{i=1}^n G_i}(\otimes_{i=1}^n V_{G_i}) D^{\frac{1}{2}} \\ &= D^{\frac{1}{2}} \Lambda (\otimes_{i=1}^n V_{G_i}) D^{\frac{1}{2}}. \end{aligned} \tag{12}$$

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

$$L_{\otimes_{i=1}^n G_i}(\otimes_{i=1}^n V_{G_i}) \approx (D\Lambda)(\otimes_{i=1}^n V_{G_i}). \tag{13}$$

Based on Equation (13), 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 1.

Next, we will prove that the estimated eigenvalues $\lambda_{k_1 k_2,\ldots,k_n}$ in THEOREM 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 $0 \leq \lambda \leq 2$. Hence, $-1 \leq 1-\lambda_{k_i}^{G_i} \leq 1$, for $i = 1, 2, \ldots, n$. Based on this, we can get that $\left|\prod_{i=1}^{n}(1-\lambda_{k_i}^{G_i})\right| \leq 1$ and thus $[1-\prod_{i=1}^{n}(1-\lambda_{k_i}^{G_i})]$ is non-negative. ■

APPENDIX C
FINDING THE JOINT FIEDLER VECTOR FOR THE ILLUSTRATIVE EXAMPLE SHOWN IN FIGURE 1(A)

(1) According to the definitions [45], the adjacency (i.e., A_1, A_2) and degree matrices (i.e., D_1, D_2) of G_1 and G_2 are:

$$A_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},\ D_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ A_2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix},\ D_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{14}$$

(2) Compute the Laplacian matrix of G_1 and G_2, namely L_1 and L_2:

$$L_1 = D_1 - A_1 = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix},\ L_2 = D_2 - A_2 = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}. \tag{15}$$

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

$$\mathcal{L}_1 = D_1^{-\frac{1}{2}} L_1 D_1^{-\frac{1}{2}} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix},\ \mathcal{L}_2 = D_2^{-\frac{1}{2}} L_2 D_2^{-\frac{1}{2}} = \begin{bmatrix} 1 & -\frac{1}{\sqrt{2}} & 0 & 0 \\ -\frac{1}{\sqrt{2}} & 1 & -\frac{1}{2} & 0 \\ 0 & -\frac{1}{2} & 1 & -\frac{1}{\sqrt{2}} \\ 0 & 0 & -\frac{1}{\sqrt{2}} & 1 \end{bmatrix}. \tag{16}$$

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

$$\lambda_1^{G_1} = 0,\quad \lambda_2^{G_1} = 2,\quad v_{1:2}^{G_1} = \frac{1}{\sqrt{2}} \left[\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right]. \tag{17}$$

$$\lambda_1^{G_2} = 0,\quad \lambda_2^{G_2} = 0.5,\quad \lambda_3^{G_2} = 1.5,\quad \lambda_4^{G_2} = 2,\quad v_{1:4}^{G_2} = \frac{1}{\sqrt{3}} \left[\begin{bmatrix} \frac{1}{\sqrt{2}} \\ 1 \\ 1 \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \begin{bmatrix} -1 \\ -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -\frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \\ 1 \end{bmatrix}, \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -1 \\ 1 \\ -\frac{1}{\sqrt{2}} \end{bmatrix} \right]. \tag{18}$$

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

$$d^{G_1} = [1,\ 1]^T,\ d^{G_2} = [1,\ 1,\ 2,\ 2]^T. \tag{19}$$

(6) According to THEOREM 1, we can get two approximations of the Fiedler vector, namely $F_{\widetilde{G}}^1$ and $F_{\widetilde{G}}^2$: (Note that the eigenvalues corresponding to the two vectors, i.e., λ_{11} and λ_{24}, are the same and the second smallest among all the eigenvalues of $G_1 \otimes G_2$, so both v_{11} and v_{24} can be viewed as the Fielder vector.)

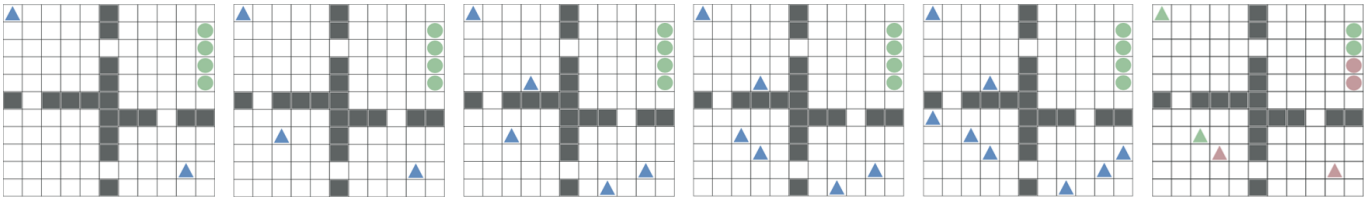
$$\lambda_{11} = \left[1-(1-\lambda_1^{G_1})(1-\lambda_1^{G_2})\right] d_1^{G_1} d_1^{G_2} = 0,\ \lambda_{24} = \left[1-(1-\lambda_2^{G_1})(1-\lambda_4^{G_2})\right] d_2^{G_1} d_4^{G_2} = 0, \tag{20}$$

$$F_{\widetilde{G}}^1 = v_{11} = v_1^{G_1} \otimes v_2^{G_2} = \frac{1}{\sqrt{6}} \left[\frac{1}{\sqrt{2}},\ 1,\ 1,\ \frac{1}{\sqrt{2}},\ \frac{1}{\sqrt{2}},\ 1,\ 1,\ \frac{1}{\sqrt{2}} \right]^T, \tag{21}$$

$$F_{\widetilde{G}}^2 = v_{24} = v_2^{G_1} \otimes v_4^{G_2} = \frac{1}{\sqrt{6}} \left[-\frac{1}{\sqrt{2}},\ 1,\ -1,\ \frac{1}{\sqrt{2}},\ \frac{1}{\sqrt{2}},\ -1,\ 1,\ -\frac{1}{\sqrt{2}} \right]^T. \tag{22}$$

APPENDIX D
MORE SIMULATION SETUP

Figure 12 shows the setup of the n-agent ($n = 2, 3, 5, 6, 8$) and 2×2-agent four-room tasks which are not illustrated in Figure 4. The triangles represent the initial positions of the agents and circles denote the goal area.



(a) 2-agent four-room (b) 3-agent four-room (c) 5-agent four-room (d) 6-agent four-room (e) 8-agent four-room (f) 2×2 four-room

Fig. 12. Simulators for Evaluation: All the agents (triangles) must reach the goal area (circles) simultaneously to complete the task, based on only their current locations. In (f), agents are assigned with different goals. The agents and their corresponding goals are labeled with the same color.