SELF-ORGANIZED POLYNOMIAL-TIME COORDINATION GRAPHS

Anonymous authors
Paper under double-blind review

ABSTRACT

Coordination graph is a promising approach to model agent collaboration in multi-agent reinforcement learning. It factorizes a large multi-agent system into a suite of overlapping groups that represent the underlying coordination dependencies. One critical challenge in this paradigm is the complexity of computing maximum-value actions for a graph-based value factorization. It refers to the decentralized constraint optimization problem (DCOP), which and whose constant-ratio approximation are NP-hard problems. To bypass this fundamental hardness, this paper proposes a novel method, named Self-Organized Polynomial-time Coordination Graphs (SOP-CG), which uses structured graph classes to guarantee the optimality of the induced DCOPs with sufficient function expressiveness. We extend the graph topology to be state-dependent, formulate the graph selection as an imaginary agent, and finally derive an end-to-end learning paradigm from the unified Bellman optimality equation. In experiments, we show that our approach learns interpretable graph topologies, induces effective coordination, and improves performance across a variety of cooperative multi-agent tasks.

1 INTRODUCTION

Cooperative multi-agent reinforcement learning (MARL) is a promising approach to a variety of real-world applications, such as sensor networks (Zhang & Lesser, 2011; Ye et al., 2015), traffic light control (Van der Pol & Oliehoek, 2016), and multi-robot formation (Alonso-Mora et al., 2017). One of the long-lasting challenges of cooperative MARL is how to organize coordination for a large multi-agent system. Coordination graph (Guestrin et al., 2001) is a classical approach to represent coordination relations in the reinforcement learning framework. It decomposes a multi-agent system into a suite of overlapping factors. Each factor is a hyper-edge covering a subset of agents that may involve coordinated behaviors. More specifically, the joint value function of the multi-agent system is factorized to the summation of local value functions based on factors or edges. Such a graph structure is a particular type of inductive bias that encodes task-specific coordination dependencies. It is known that conducting appropriate inductive bias can both reduce the model complexity and improve the generalizability (Battaglia et al., 2018).

One fundamental problem for coordination graphs is the trade-off between the representational capacity of value functions and the computational complexity of policy execution. To obtain value functions with high expressiveness, an advanced approach, deep coordination graphs (DCG) (Böhmer et al., 2020), considers a static complete graph connecting all pairs of agents. This graph structure has high representational capacity in terms of function expressiveness but raises a challenge for computation in the execution phase. The greedy action selection over a coordination graph can be formalized to a decentralized constraint optimization problem (DCOP), finding the maximum-value joint actions (Guestrin et al., 2001; Zhang & Lesser, 2013; Böhmer et al., 2020). Note that the DCOP and its any constant-ratio approximation are NP-hard problems, especially for the complete graph used in DCG (Dagum & Luby,

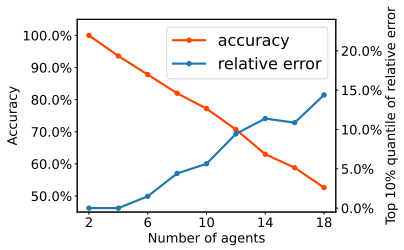

Figure 1: A motivating example with the accuracy and the relative joint Q error of max-sum algorithm w.r.t. the number of agents.

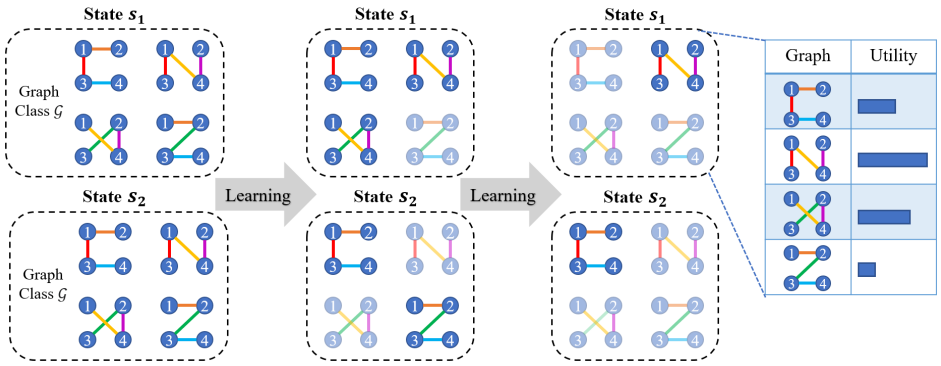

Figure 2: Illustration of our basic idea. For ease of viewing, we mark each edge with a unique color. Our algorithm conducts different coordination graphs for different environment states. For example, the selection of graph topologies on states s_1 and s_2 depends on the utility functions on corresponding states. These utility values are updated through the learning procedure.

1993; Park & Darwiche, 2004). From the perspective of theoretical computer science, any scalable heuristic algorithm (e.g., max-sum algorithm (Pearl, 1988)) for DCOP problem may have an unpredictable gap with the optimal solution. To visualize this phenomenon, we conduct a motivating example for generating a suite of complete-graph DCOPs with random edge-values. More detailed experiment setting are deferred to to Appendix A. We investigate the accuracy and the relative joint Q error of the max-sum algorithm in computing optimal joint actions, which is the default implementation of MARL methods based on coordination graphs (Stranders et al., 2009; Zhang & Lesser, 2013; Böhmer et al., 2020). The results are shown in Figure 1. As the number of agents increases, the accuracy of greedy action selection consistently decreases, and the relative joint Q error between the selected and optimal joint actions increases accordingly. How to incorporating polynomial-time complexity guarantee into coordination graphs with sufficient function expressiveness remains an open problem in MARL.

In this paper, we study a dilemma in coordination graphs, i.e., expressive graph structures lead to computationally intractable DCOPs but concise graph structures are in lack of function expressiveness. To tackle this problem, we propose a novel graph-based MARL method for multi-agent coordination, named *Self-Organized Polynomial-time Coordination Graphs* (SOP-CG), that utilizes structured graph topologies with polynomial-time guarantees for DCOPs and extends their representation expressiveness through a dynamic graph organization mechanism. More specifically, the topology of coordination graph G can depend on the environment state. Such a dynamic factorization structure can be self-organized, which is a major characteristic of our proposed approach. We introduce an imaginary coordinator agent, whose action is to select a graph topology in a given class at each time step. SOP-CG incorporates this imaginary agent into the Bellman optimality equation supporting end-to-end learning. Figure 2 illustrates the basic idea of SOP-CG. We integrate the search of graph structures into the MARL paradigm. The utilities of different graph topologies are learned along with the RL objective. As the graph utilities are updated through agent-environment interactions, the coordination graph will evolve to the topology characterizing the coordination dependencies among agents.

We evaluate the performance of SOP-CG in both grid-world and simulated physical environments. Empirical results on challenging tasks demonstrate that SOP-CG outperforms state-of-the-art baselines. By extensive ablation studies, we verify that harnessing specific structures with optimality guarantee of the induced DCOPs improves the sample quality and the accuracy of the one-step TD target. Furthermore, we show that SOP-CG can learn interpretable and context-dependent coordination graphs, which induces effective and dynamic coordination among agents.

2 Related Work

Multi-agent reinforcement learning (OroojlooyJadid & Hajinezhad, 2019) is challenged by the size of joint action space, which grows exponentially with the number of agents. Independent Q-learning

(Tan, 1993; Foerster et al., 2017) models agents as independent learners, which makes the environment non-stationary in the perspective of each agent. An alternative paradigm called centralized training and decentralized execution (CTDE; Kraemer & Banerjee, 2016) is widely used in both policy-based and value-based methods. Policy-based multi-agent reinforcement learning methods use a centralized critic to compute gradient for the local actors (Lowe et al., 2017; Foerster et al., 2018; Wen et al., 2019; Wang et al., 2020d). Value-based methods usually decompose the joint value function into individual value functions under the IGM (individual-global-max) principle, which guarantees the consistency between local action selection and joint action optimization (Sunehag et al., 2018; Rashid et al., 2020b; Son et al., 2019; Wang et al., 2021a; Rashid et al., 2020a). Other work also studies this problem from the perspective of agent roles and individuality (Wang et al., 2020a;b; Jiang & Lu, 2021) or communication learning (Singh et al., 2018; Das et al., 2019; Wang et al., 2020c). Compared to these methods, our work is built upon graph-based value decomposition, which explicitly models the interaction among agents.

Coordination graphs are classical technique for planning in multi-agent systems (Guestrin et al., 2001; 2002b). They are combined with multi-agent deep reinforcement learning by recent work (Castellini et al., 2019; Böhmer et al., 2020; Li et al., 2020; Wang et al., 2021b). Joint action selection on coordination graphs can be modeled as a decentralized constraint optimization problem (DCOP), and previous methods compute approximate solutions by message passing among agents (Pearl, 1988). As the work which is most closed to our method, Zhang & Lesser (2013) presents an algorithm based on coordination graph searching. They define a measurement to quantify the potential loss of the lack of coordination between agents and search for a coordination structure to minimize the communication cost within restricted loss of utilities. However, their induced DCOPs still remain NP-hard. In contrast, minimizing communication is not our core motivation, and we aim to use a structured graph class to maintain sufficient function expressiveness when bypassing the computational hardness of large-scale DCOPs (Dagum & Luby, 1993; Park & Darwiche, 2004).

3 BACKGROUND

In this paper, we consider about fully cooperative multi-agent tasks that can be modelled as a Dec-POMDP (Oliehoek et al., 2016) defined as $\mathcal{M} = \langle D, S, \{A^i\}_{i=1}^n, T, \{O^i\}_{i=1}^n, \{\sigma^i\}_{i=1}^n, R, h, b_0, \gamma\rangle$, where $D = \{1, \ldots, n\}$ is the set of n agents, S is a set of states, h is the horizon of the environment, $\gamma \in [0, 1)$ is the discount factor, and $b_0 \in \Delta(S)$ denotes the initial state distribution. At each stage t, each agent i takes an action $a_i \in A^i$ and forms the joint action $\boldsymbol{a} = (a_1 \ldots, a_n)$, which leads to a next state s' according to the transition function $T(s'|s, \boldsymbol{a})$ and an immediate reward $R(s, \boldsymbol{a})$ shared by all agents. Each agent i observes the state only partially by drawing observations $o_i \in O^i$, according to σ_i. The joint history of agent i's observations $o_{i,t}$ and actions $a_{i,t}$ is denoted as $\tau_{i,t} = (o_{i,0}, a_{i,0}, \ldots, o_{i,t-1}, a_{i,t-1}, o_{i,t}) \in \left(O^i \times A^i\right)^t \times O^i$.

Deep Q-Learning. Q-learning algorithms is a well-known algorithm to find the optimal joint action-value function $Q^*(s, \boldsymbol{a}) = r(s, \boldsymbol{a}) + \gamma \mathbb{E}_{s'}[\max_{\boldsymbol{a}'} Q^*(s', \boldsymbol{a}')]$. Deep Q-learning approximates the action-value function with a deep neural network with parameters $\boldsymbol{\theta}$. In Multi-agent Q-learning algorithms (Sunehag et al., 2018; Rashid et al., 2020b; Son et al., 2019; Wang et al., 2021a), a replay memory D is used to store the transition tuple $(\boldsymbol{\tau}, \boldsymbol{a}, r, \boldsymbol{\tau}')$, where r is the immediate reward when taking action $\boldsymbol{a}$ at joint action-observation history $\boldsymbol{\tau}$ with a transition to τ'. $Q(\boldsymbol{\tau}, \boldsymbol{a}; \boldsymbol{\theta})$ is used in place of $Q(s, \boldsymbol{a}; \boldsymbol{\theta})$, because of partial observability. Hence, parameters $\boldsymbol{\theta}$ are learnt by minimizing the following expect TD error:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{\tau}, \boldsymbol{a}, r, \boldsymbol{\tau}') \in D}\left[\left(r + \gamma V\left(\boldsymbol{\tau}'; \boldsymbol{\theta}^-\right) - Q\left(\boldsymbol{\tau}, \boldsymbol{a}; \boldsymbol{\theta}\right)\right)^2\right] \tag{1}$$

where $V(\boldsymbol{\tau}'; \boldsymbol{\theta}^-) = \max_{\boldsymbol{a}'} Q(\boldsymbol{\tau}', \boldsymbol{a}'; \boldsymbol{\theta}^-)$ is the one-step expected future return of the TD target and $\boldsymbol{\theta}^-$ are the parameters of the target network, which will be periodically updated with $\boldsymbol{\theta}$.

Coordination graphs. An undirect coordination graph (CG, Guestrin et al. (2001)) $\mathcal{G} = \langle \mathcal{V}, \mathcal{E}\rangle$ contains vertex $v_i \in \mathcal{V}$ for each agent $1 \leq i \leq n$ and a set of (hyper-)edges in $\mathcal{E} \subseteq 2^{\mathcal{V}}$ which represents coordination dependencies among agents. Prior work considers higher order coordination where the edges depend on actions of several agents (Guestrin et al., 2002a; Kok & Vlassis, 2006; Guestrin et al., 2002b). Such a coordination graph induces a factorization of global Q function:

$$Q_{tot}(s, \boldsymbol{a}) = \sum_{v^i \in \mathcal{V}} f^i(a^i|s) + \sum_{\boldsymbol{e}=\{e_1,\ldots,e_{|\boldsymbol{e}|}\} \in \mathcal{E}} f^{\boldsymbol{e}}(a^{e_1}, \ldots, a^{e_{|\boldsymbol{e}|}}|s) \tag{2}$$

The special case that $\mathcal{E}$ is an empty set yields VDN, but each additional edge enables the value representation of the joint actions of a pair of agents and can thus help to avoid relative-overgeneralization (Böhmer et al., 2020). In many coordination graph learning works (Zhang & Lesser, 2013; Böhmer et al., 2020; Wang et al., 2021b), the hyper-edges are simplified into pairwise edges. The graph is usually considered to be specified before training. Guestrin et al. (2002b) and Zhang & Lesser (2013) suggest that the graph could also depend on states, which means each state can have its own unique CG.

4 Self-Organized Polynomial-Time Coordination Graphs

A common method for multi-agent reinforcement learning is to decompose the joint value function into a linear combination of local value functions, each of which conditions on actions of a subset of agents (Guestrin et al., 2001; Böhmer et al., 2020). With this paradigm, computing joint action with maximum value can be modeled as a distributed constraint optimization problem (DCOP). General DCOP and any constant-ratio approximation have been proved to be NP-hard (Dagum & Luby, 1993; Park & Darwiche, 2004). Previous work adopts a variety of heuristic algorithms for action selection (Kok & Vlassis, 2006; Wang et al., 2021b). The imperfection of heuristics may lead to two side effects: (a) collecting bad samples and (b) errors in constructing one-step TD target, which hurt the learning performance.

To address these issues, we investigate *polynomial-time coordination graphs*, in which the induced DCOPs can be solved in polynomial time. We present a novel algorithm called *Self-Organized Polynomial-time Coordination Graphs* (SOP-CG), that utilizes a class of polynomial-time coordination graphs to construct a dynamic and state-dependent topology. In Section 4.1, we introduce our value factorization upon coordination graphs, which can hold the optimality guarantee of the induced DCOPs without significantly sacrificing the representational capacity by using a bottom-up design. To enable efficient topology self-organization, in Section 4.2, we unify the graph selection into the Q-learning framework by formulating the graphs as actions of an imaginary coordinator agent, and depict the whole framework of our algorithm. In Section 4.3, we further propose two polynomial-time graph class instances and discuss their theoretical advantages in SOP-CG.

4.1 Value Factorization on Polynomial-Time Coordination Graphs

To establish an algorithm with optimality guarantee in action selection, our approach first models coordination relations upon the graph-based value factorization specified by deep coordination graphs (DCG; Böhmer et al., 2020), in which the joint value function of the multi-agent system is factorized to the summation of *individual utility functions* q_i and *pairwise payoff functions* q_{ij} as follows:

$$Q(\boldsymbol{\tau}^{(t)}, \boldsymbol{a}; G) = \sum_{i \in [n]} q_i(\tau_i^{(t)}, a_i) + \sum_{(i,j) \in G} q_{ij}(\tau_i^{(t)}, \tau_j^{(t)}, a_i, a_j), \tag{3}$$

where the coordination graph G is represented by a set of undirected edges. The induced DCOP can be solved in polynomial time if G lies in $\mathcal{G}_{\text{uac}}$, the set of undirected acyclic graphs (Fioretto et al., 2018). However, an undirected acyclic graph can contain at most $n-1$ edges in an environment with n agents. For long-horizon complex multi-agent tasks, such a static sparse graph usually cannot represent all potential coordination dependencies.

To alleviate this problem, our approach allows the graph topology to dynamically evolve through the transitions of environment status. Given different environmental states, the joint values can be factorized with different coordination graphs chosen from a predefined graph class $\mathcal{G} \subseteq \mathcal{G}_{\text{uac}}$. This design is based on an assumption that, although a long-horizon task cannot be characterized by a static sparse coordination graph, the coordination relations at each single-time step are sparse and manageable. The employment of this graph class fills the lack of representational capacity as well as maintains a polynomial complexity of the induced DCOPs.

4.2 LEARNING SELF-ORGANIZED TOPOLOGY WITH AN EXPLICIT COORDINATOR

Now we present a novel framework to render the self-organized coordination graph. We introduce an imaginary coordinator agent whose action space refers to the selection of graph topologies, aiming to select a proper graph for minimizing the loss within restricted coordination. When using the graph-based value factorization stated in Eq. (3), the graph topology G can be regarded as an input of joint value function $Q(\boldsymbol{\tau}^{(t)}, \boldsymbol{a}; G)$. The objective of the coordinator agent is to maximize the joint value function over the specific graph class, which is a dual problem for finding a graph to minimize the loss within the different restricted coordination structures (Zhang & Lesser, 2013). Under this interpretation, we can integrate the selection of graph topologies into the trial-and-error loop of reinforcement learning. We handle the imaginary coordinator as a usual agent in multi-agent Q-learning framework and rewrite the joint action as $\boldsymbol{a}_{cg} = (a_1, \cdots, a_n, G)$.

Execution. Formally, at time step t, greedy action selection indicates the following joint action:

$$\boldsymbol{a}_{cg}^{(t)} \leftarrow \underset{(a_1,\cdots,a_n,G)}{\arg\max}\, Q(\boldsymbol{\tau}^{(t)}, a_1, \cdots, a_n; G). \tag{4}$$

Hence the action of coordinator agent $G^{(t)}$ is naturally the corresponding component in Eq. (4):

$$G^{(t)} \leftarrow \underset{G\in\mathcal{G}}{\arg\max} \left(\max_{\mathbf{a}} Q(\boldsymbol{\tau}^{(t)}, \boldsymbol{a}; G) \right). \tag{5}$$

After determining the graph topology $G^{(t)}$, the agents can choose their individual actions to jointly maximize the value function $Q(\boldsymbol{\tau}^{(t)}, \boldsymbol{a}; G^{(t)})$ upon the selected topology.

Training. With the imaginary coordinator, we can re-formulate the Bellman optimality equation and maximizing the future value over the coordinator agent's action:

$$Q^*(\boldsymbol{\tau}, \boldsymbol{a}; G) = \mathop{\mathbb{E}}_{\boldsymbol{\tau}'} \left[r + \gamma \max_{G'} \max_{\boldsymbol{a}'} Q^*(\boldsymbol{\tau}', \boldsymbol{a}'; G') \right] \tag{6}$$

Since the graph selection is a part of agent action, the associative value $Q(\boldsymbol{\tau}, \boldsymbol{a}; G)$ of graph G can be updated through temporal difference learning. The network parameters $\boldsymbol{\theta}$ can be updated by minimizing the standard Q-learning TD loss:

$$\mathcal{L}_{cg}(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{(\boldsymbol{\tau},\boldsymbol{a},G,r,\boldsymbol{\tau}')\sim\mathcal{D}} \left[\left(y_{cg} - Q(\boldsymbol{\tau}, \boldsymbol{a}; G; \boldsymbol{\theta}) \right)^2 \right] \tag{7}$$

where $y_{cg} = \max_{(\boldsymbol{a}',G')} Q(\boldsymbol{\tau}', \boldsymbol{a}'; G'; \boldsymbol{\theta}^-)$ is the one-step TD target and $\boldsymbol{\theta}^-$ are the parameters of target network. $G^{(t)}$ is regarded as part of the transition and is stored in the replay buffer together with the realistic agent actions. We include more implementation details of temporal difference learning in Appendix C.

The overall algorithm of our approach is summarized in Algorithm 1. The graph selection step at line 6 is the main characteristic of our method, which differs from the static graph representation used by prior work (Guestrin et al., 2002a; Kok & Vlassis, 2006; Böhmer et al., 2020). To support the self-organization of coordination graphs, the graph selection step raises a new computation demand, i.e., we need to find the best candidate from the graph class $\mathcal{G}$, which will be another non-trivial combinatorial optimization problem. We will show that, by designing a proper graph class $\mathcal{G}$, we can maintain both computational efficiency and function representational capacity in the next subsection.

4.3 HARNESSING STRUCTURED GRAPH CLASSES FOR EFFICIENT COORDINATION

To achieve the efficient graph search over the polynomial-time class instances, we investigate the following structures for the graph class $\mathcal{G}$, which are subsets of $\mathcal{G}_{\text{uac}}$:

- *Pairwise grouping* $\mathcal{G}_P$. This class is introduced by Castellini et al. (2019). n agents are partitioned to $\lfloor n/2 \rfloor$ non-overlapping pairwise groups. Only pairwise interactions are considered and two agents are connected with an edge if they lie in the same group.

Algorithm 1 Self-Organized Polynomial-Time Coordination Graphs

Require: Predefined graph class $\mathcal{G}$
Initialize replay buffer $\mathcal{D}$
Initialize network with random weights
for episode$= 1, \cdots, M$ **do**
  Receive initial observation $\boldsymbol{\tau}^{(0)}$
  for $t = 0, \cdots, T$ **do**
    Choose coordination graph $G^{(t)} \in \mathcal{G}$ according to Eq. (5) ▷ Section 4.3
    Select actions $\boldsymbol{a}^{(t)}$ w.r.t $G^{(t)}$ and an ϵ-greedy exploration
    Execute actions $\boldsymbol{a}^{(t)}$ and receive reward $r^{(t)}$ and observation $\boldsymbol{o}^{(t+1)}$
    Store transition $(\boldsymbol{\tau}^{(t)}, \boldsymbol{a}^{(t)}, r^{(t)}, \boldsymbol{\tau}^{(t+1)})$ in $\mathcal{D}$
  end for
  Sample random minibatch of transitions from $\mathcal{D}$
  Perform a gradient descent step on loss defined in Eq. (7)
end for

- *Hierarchical organization* $\mathcal{G}_H$. Overlapping edges are allowed in this class. In a system with n agents, the graph is organized as a tree with $n-1$ edges so that all agents form a connected component. Even if two agents are not directly connected by an edge, their actions are implicitly coordinated through the path on the tree. This provides the potential to approximate complex joint value functions.

$\mathcal{G}_H$ is actually the set of maximal undirected acyclic graphs. It is noteworthy that we do not further incorporate the smaller graphs into both classes, since their representational capacity are dominated by existing graphs in the classes. We summarize the hardness of computing these problems upon different graph class in Table 1. Although the representational capacity of $\mathcal{G}_P$ is relatively weaker, selecting $G^{(t)}$ from this class can be rigorously solved by polynomial-time algorithms (Edmonds, 1965), which highlights the key benefit of $\mathcal{G}_P$. In comparison, selecting $G^{(t)}$ from $\mathcal{G}_H$ is non-trivial, so we design a greedy algorithm to compute approximated solutions. Note that, although the graph selection of $\mathcal{G}_H$ may be imprecise, its represented values are formally optimized by temporal difference learning with accurate action selection. Thus, as we will show in the experiment section, $\mathcal{G}_H$ can still work as a promising graph class in absence of rigorous graph selection. Please refer to Appendix B for the implementation details of the induced combinatorial optimization for graph selection.

Graph Class	Select graph $G^{(t)}$	Select actions on a given graph
$\mathcal{G}_P$	$\surd$	$\surd$
$\mathcal{G}_H$	$-$	$\surd$
Complete graph	N/A	$-$

Table 1: Computational considerations in choosing graph class. A check mark "$\surd$" means that known algorithms for the problem are guaranteed to find the optimal solution in polynomial time. The mark "$-$" denotes that these settings use the specific heuristic methods.

Remark. In this paper, SOP-CG trades off the representational capacity of value function with the computational complexity within the polynomial-time boundary. Compared to fully connected graph, our sparse graphs restrict the value function class to a smaller space. This polynomial-time graph class enables SOP-CG to achieve accurate greedy action selection using limited message passing along the edges and achieve effective learning in the centralized training process. Moreover, if these graph classes are enough to express the coordination dependencies in a multi-agent task, using such topologies will be more sample-efficient by virtue of the concise function classes (Battaglia et al., 2018), which becomes another potential advantage of our algorithm.

5 EXPERIMENTS

In this section, we conduct experiments to answer the following questions: (1) Does the class of polynomial-time coordination graphs improve our performance (see Section 5.1)? (2) How well

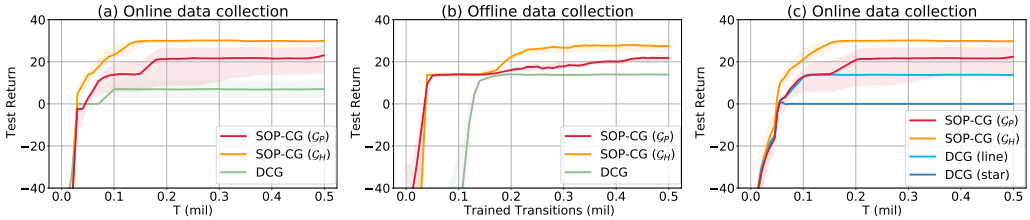

Figure 3: Ablation studies on Sensor-network: (a) comparison against DCG with online data collection; (b) comparison against DCG with offline dataset; (c) comparison with static topologies.

does SOP-CG perform on complex cooperative multi-agent tasks (see Fig. 5)? (3) Can SOP-CG extract interpretable dynamic coordination structures (see Fig. 6)?

5.1 Didactic Example

We first study a didactic example to demonstrate two main contributions of this paper: (a) harnessing polynomial-time coordination graphs to obtain optimality guarantee of the induced DCOPs; (b) maintaining function expressiveness by incorporating a class of structured graphs into the Q-learning framework. We conduct ablation studies on *Sensor-network*, which is a classic task for testing multi-agent coordination (Lesser et al., 2003; Zhang & Lesser, 2011). We adopt the same task configuration as Wang et al. (2021b). As illustrated in Fig. 4, 15 sensor agents are arranged as a 3×5 matrix while three targets randomly walk in the same grid space. At each timestep, an agent can choose to scan one of the eight nearby positions (inducing a cost) or perform no operation. If two or more agents scan the target simultaneously, the target is captured and produces a positive joint reward. We compare our algorithm with DCG (Böhmer et al., 2020) on this game. The default implementation of DCG uses a fully connected coordination graph. In addition, DCG provides alternative implementations on two static graphs *Line* (agent i is connected with agent $i-1$ and $i+1$) and *Star* (agent 1 is linked with others), which are two special instances of $\mathcal{G}_H$. SOP-CG ($\mathcal{G}_H$) has absolutely higher function expressiveness than DCG (line) and DCG (star).

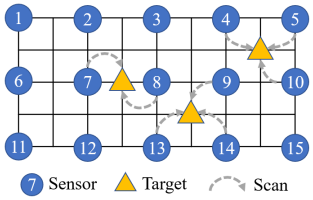

Figure 4: Sensor-network.

For the first contribution, we verify that the optimality guarantee of the induced DCOP in our algorithm improve the performance. We test these algorithms in both online (Fig. 3a) and offline learning (Fig. 3b) settings. For fair evaluation, we train DCG (line) and collect 15K experienced episodes throughout the training process to set up the diverse offline dataset. As shown in Fig. 3(a-b), SOP-CG outperforms DCG in both settings. In the offline setting, as the dataset is fixed for both algorithms, the main difference between SOP-CG and DCG lies in the construction of one-step TD target. Our algorithm is guaranteed to find the optimal solution of the induced DCOPs, which must play an important role in reducing the errors in computing TD target. We also observe that DCG sticks in a bad local optimum with online data collection, but its performance is improved in the offline setting. This result indicates that the hardness of greedy action selection causes DCG to collect low-quality samples in the online setting, which may hurt the final performance. In contrast, our method converges to the same superior value in both settings, highlighting the effectiveness of polynomial-time coordination graphs.

For the second contribution, we show the importance of learning a state-dependent structure. As illustrated in Fig. 3c, the performance of SOP-CG is much better than both DCG (line) and DCG (star) with online data collection. A static sparse topology usually cannot express the coordination dependencies in all steps, so these methods could not work well even if they are built upon polynomial-time coordination graphs.

5.2 Performance Comparison

To further demonstrate the effectiveness of our algorithm and discuss the difference among the graph classes, we evaluate SOP-CG on three large-scale complex tasks with highly dynamic coordination requirements: Pursuit, Chase and Postman.

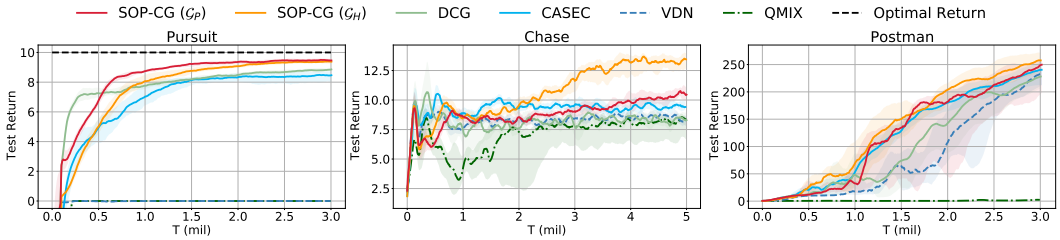

Figure 5: Learning curves on Pursuit, Chasing and Postman. The optimal return is demonstrated if it is available.

- *Pursuit*. This environment is also called predator-and-prey, which is widely used in recent work (Son et al., 2019; Böhmer et al., 2020; Wang et al., 2021b). 20 agents and 10 random walking prey interact on a 10×10 grid-world. Based on a limited sight range, an agent can move on the map or catch adjacent prey. One prey is captured only if at least two adjacent agents catch it simultaneously, inducing a joint reward of 1. After that, both agents and the captured prey are removed from the game. If only one agent tries to catch the prey, it will fail and receive a punishment of -1.
- *Chase*. This is a challenging task on the particle world with trained adversaries instead of built-in AIs created by Lowe et al. (2017). On the map with 3 randomly generated obstacles, 10 slower agents chase 3 faster adversaries. For each collision between an agent and an adversary, the agents receive a reward while the adversaries are punished. An agent or adversary can observe the units in a restricted circular area that is centered at its position. We train the agents by a specific algorithm, and the adversaries are concurrently trained by VDN (Sunehag et al., 2018).
- *Postman*. 12 postmen work in a vast community to deliver letters from residents to the post office. With the post office located in the northwest of the community, the whole community is separated into three parts according to the distance to the post office, and each postman will only work in one region. Postmen need to pick randomly generated letters in the community. While two postmen meet each other, any one of them can offer its letters to another one, so they can collaborate in order to deliver the letters to the post office. A postman can observe the information of other postmen and the letters in a sight range.

We compare our method with the state-of-the-art value-based MARL methods (VDN (Sunehag et al., 2018), QMIX (Rashid et al., 2020b)) and coordination graph learning methods (DCG (Böhmer et al., 2020), CASEC (Wang et al., 2021b)). The implementation details and hyperparameter settings are included in appendix D. As presented in Fig. 5, SOP-CG outperforms the baselines in all three environments.

Using a smaller graph class will be more efficient if such a concise graph class is sufficient to express the coordination dependency in the task of interest. Taking the task Chase as an example, DCG performs as poorly as the fully-decomposed methods, as their graph classes are too large and complex to learn. However, our method significantly outperforms VDN, QMIX, and coordination graph learning algorithms with a smaller graph class, which strongly confirms our statement. Furthermore, SOP-CG ($\mathcal{G}_H$) has better performance than SOP-CG ($\mathcal{G}_P$) on this task, which may be due to the lack of representational capacity. On the contrary, Pursuit mainly involves pairwise cooperation, so SOP-CG ($\mathcal{G}_P$) achieves better sample efficiency than SOP-CG ($\mathcal{G}_H$).

We will show that reducing the error of DCOP can also improve performance. On task Pursuit with 20 agents, our method achieves almost the optimal of this task while DCG has over two times larger gap with the optimal value than ours. The reason is that Pursuit requires precise collaborations. If only one agent tries to catch the prey, it will fail and induce a large punishment. When facing growing scales, DCG and CASEC have a poor accuracy on DCOP, as we showed before, which may lead to inaccurate greedy action selection, impeding DCG to better performance. In the super hard task Postman, which requires agents to collaborate during long horizon, our method also outperforms other methods.

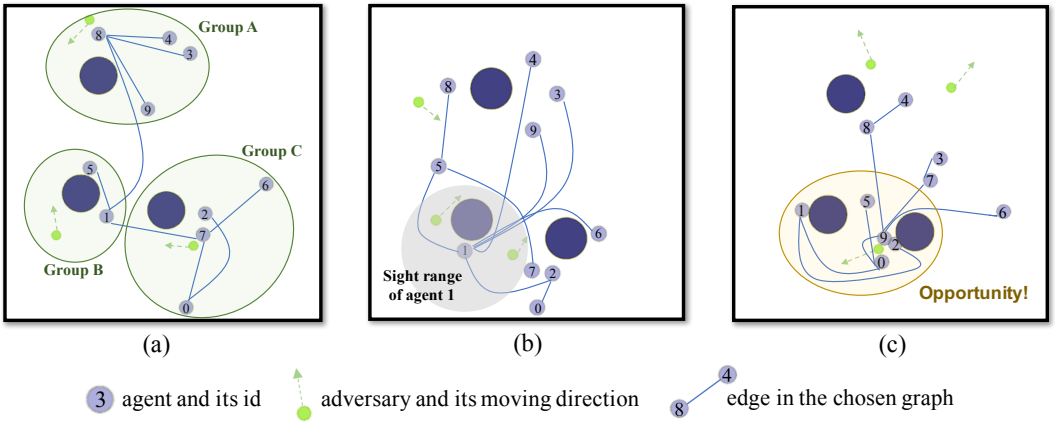

Figure 6: Various coordination graphs learned by our algorithm on Chase: (a) Self-organized grouping at initialization; (b) Connecting to agent with rich observation for better information sharing; (c) Concentrated collaboration structure around an enclosed adversary.

5.3 Visualization of Self-Organized Coordination

A major strength of our method is that SOP-CG can organize different coordination graphs to adapt to different situations. Such an adaptive organization mechanism can improve the efficiency of multi-agent collaboration. To illustrate the coordination relations organized by SOP-CG, we visualize the learned graph structures on Chase. In Fig. 6, we take three snapshots chronologically from one episode. This episode is collected by SOP-CG ($\mathcal{G}_H$) using the tree-based graph class. It shows that SOP-CG can produce interpretable graph structures that characterize the ground-truth coordination dependencies. The learned coordination behaviors are interpreted as follows:

1. Fig. 6a presents the initial status of the game, where agents and adversaries are generated randomly on the map. As the three adversaries distribute in separate positions, the agents naturally form three groups to chase different adversaries.
2. After several time steps, in Fig. 6b, two adversaries gather at the bottom left of the map. Note that, in this game, each agent can only observe objects within a small circular range. Only agent 1 can observe the two adversaries simultaneously, which makes it connect to the majority of the rest of the agents and helps the other agents recognize the position of the adversaries.
3. Finally, as illustrated by Fig. 6c, the bottom adversary is surrounded by multiple agents, yielding an opportunity for the agents to win considerable rewards, which makes the collaboration structure concentrate around the adversary.

As presented above, the graph structures learned by SOP-CG definitely match the ground-truth demands for effective collaboration. It demonstrates the ability of our approach to organizing coordination relations.

6 Conclusion

This paper introduces SOP-CG, a novel multi-agent graph-based method that guarantees the polynomial-time greedy policy execution with expressive function representation of a structured graph class. To enable end-to-end learning, SOP-CG introduces an imaginary agent to dynamically select the state-dependent graph and incorporates it into a unified Bellman optimality equation. We demonstrate that SOP-CG can learn interpretable graph topologies and outperform state-of-the-art baselines on several challenging tasks. Although the graph class is exponentially large, we conduct two specific polynomial-time graph class instances to achieve impressive performance. Designing effective graph exploration methods to deal with larger graph classes will be an interesting future direction.

ETHICS AND REPRODUCIBILITY STATEMENTS

The main contribution of our work locates on the research of a foundation theoretical problem in optimality theory and try to improve the problem with our method. Our work does not have a bad social impact. The source code of our method for all the experiments is attached in supplementary material, and the settings and parameters for all models and algorithms mentioned in the experiment section please refer to Appendix D.

REFERENCES

Javier Alonso-Mora, Stuart Baker, and Daniela Rus. Multi-robot formation control and object transport in dynamic environments via constrained optimization. *The International Journal of Robotics Research*, 36(9):1000–1021, 2017.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Wendelin Böhmer, Vitaly Kurin, and Shimon Whiteson. Deep coordination graphs. In *International Conference on Machine Learning*, pp. 980–991. PMLR, 2020.

Jacopo Castellini, Frans A Oliehoek, Rahul Savani, and Shimon Whiteson. The representational capacity of action-value networks for multi-agent reinforcement learning. In *AAMAS 2019: The 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1862–1864. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2019.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Paul Dagum and Michael Luby. Approximating probabilistic inference in bayesian belief networks is np-hard. *Artificial intelligence*, 60(1):141–153, 1993.

Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. Tarmac: Targeted multi-agent communication. In *International Conference on Machine Learning*, pp. 1538–1546, 2019.

Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17:449–467, 1965.

Ferdinando Fioretto, Enrico Pontelli, and William Yeoh. Distributed constraint optimization problems and applications: A survey. *Journal of Artificial Intelligence Research*, 61:623–698, 2018.

Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip HS Torr, Pushmeet Kohli, and Shimon Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. In *International conference on machine learning*, pp. 1146–1155. PMLR, 2017.

Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Carlos Guestrin, Daphne Koller, and Ronald Parr. Multiagent planning with factored mdps. In *NIPS*, volume 1, pp. 1523–1530, 2001.

Carlos Guestrin, Michail Lagoudakis, and Ronald Parr. Coordinated reinforcement learning. In *ICML*, volume 2, pp. 227–234. Citeseer, 2002a.

Carlos Guestrin, Shobha Venkataraman, and Daphne Koller. Context-specific multiagent coordination and planning with factored mdps. In *AAAI/IAAI*, pp. 253–259, 2002b.

Hado Hasselt. Double q-learning. *Advances in neural information processing systems*, 23:2613–2621, 2010.

Jiechuan Jiang and Zongqing Lu. The emergence of individuality. In *International Conference on Machine Learning*, pp. 4992–5001. PMLR, 2021.

Jelle R Kok and Nikos Vlassis. Collaborative multiagent reinforcement learning by payoff propagation. *Journal of Machine Learning Research*, 7:1789–1828, 2006.

Landon Kraemer and Bikramjit Banerjee. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190:82–94, 2016.

Victor Lesser, Charles L Ortiz Jr, and Milind Tambe. *Distributed sensor networks: A multiagent perspective*, volume 9. Springer Science & Business Media, 2003.

Sheng Li, Jayesh K Gupta, Peter Morales, Ross Allen, and Mykel J Kochenderfer. Deep implicit coordination graphs for multi-agent reinforcement learning. *arXiv preprint arXiv:2006.11438*, 2020.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Neural Information Processing Systems (NIPS)*, 2017.

Frans A Oliehoek, Christopher Amato, et al. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016.

Afshin OroojlooyJadid and Davood Hajinezhad. A review of cooperative multi-agent deep reinforcement learning. *arXiv preprint arXiv:1908.03963*, 2019.

James D Park and Adnan Darwiche. Complexity results and approximation strategies for map explanations. *Journal of Artificial Intelligence Research*, 21:101–133, 2004.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020a.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv preprint arXiv:2003.08839*, 2020b.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philiph H. S. Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft Multi-Agent Challenge. *CoRR*, abs/1902.04043, 2019.

Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. *arXiv preprint arXiv:1812.09755*, 2018.

Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 5887–5896, 2019.

Ruben Stranders, Alessandro Farinelli, Alex Rogers, and Nicholas R Jennings. Decentralised coordination of mobile sensors using the max-sum algorithm. In *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2085–2087, 2018.

Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pp. 330–337, 1993.

Elise Van der Pol and Frans A Oliehoek. Coordinated deep reinforcement learners for traffic light control. *Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016)*, 2016.

Jianhao Wang, Zhizhou Ren, Terry Liu, Yu Yang, and Chongjie Zhang. Qplex: Duplex dueling multi-agent q-learning. In *International Conference on Learning Representations*, 2021a.

Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. Multi-agent reinforcement learning with emergent roles. In *International Conference on Machine Learning*, 2020a.

Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, and Chongjie Zhang. Rode: Learning roles to decompose multi-agent tasks. *arXiv preprint arXiv:2010.01523*, 2020b.

Tonghan Wang, Jianhao Wang, Chongyi Zheng, and Chongjie Zhang. Learning nearly decomposable value functions via communication minimization. In *International Conference on Learning Representations*, 2020c.

Tonghan Wang, Liang Zeng, Weijun Dong, Qianlan Yang, Yang Yu, and Chongjie Zhang. Context-aware sparse deep coordination graphs. *arXiv preprint arXiv:2106.02886*, 2021b.

Yihan Wang, Beining Han, Tonghan Wang, Heng Dong, and Chongjie Zhang. Off-policy multi-agent decomposed policy gradients. *arXiv preprint arXiv:2007.12322*, 2020d.

Ying Wen, Yaodong Yang, Rui Luo, Jun Wang, and Wei Pan. Probabilistic recursive reasoning for multi-agent reinforcement learning. *arXiv preprint arXiv:1901.09207*, 2019.

Dayong Ye, Minjie Zhang, and Yun Yang. A multi-agent framework for packet routing in wireless sensor networks. *Sensors*, 15(5):10026–10047, 2015.

Chongjie Zhang and Victor Lesser. Coordinated multi-agent reinforcement learning in networked distributed pomdps. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

Chongjie Zhang and Victor Lesser. Coordinating multi-agent reinforcement learning with limited communication. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pp. 1101–1108, 2013.

A EXPERIMENTAL DETAILS OF THE MOTIVATING EXAMPLE

Algorithms. To get the correct result of DCOP, we use a brute force with complexity $O(|A|^n)$, while we choose Max-sum algorithm (has an alternative implementation called Max-plus) (Pearl, 1988), which is commonly used by most of the coordination graph learning methods (Zhang & Lesser, 2013; Böhmer et al., 2020; Wang et al., 2021b) to test the accuracy and relative Q error. The details of the algorithm are presented below (Zhang & Lesser, 2013).

Max-sum constructs a bipartite graph $\mathcal{G}_m = \langle \mathcal{V}_a, \mathcal{V}_q, \mathcal{E}_m \rangle$ according the coordination graph $\langle \mathcal{V}, \mathcal{E} \rangle$. Each node $v \in \mathcal{V}_m$ represents an agent who needs to do action selection, and each node $u \in \mathcal{V}_q$ represents a (hyper-)edge function. Edges in $\mathcal{E}_m$ connect u with the corresponding agent nodes. Max-sum algorithm will do multi-round message passing on this graph. The number of iterations is usually set smaller than 10 in previous works, while we set it 100 in this experiment.

Message passing on this bipartite graph starts with sending messages from node $v \in \mathcal{V}_a$ to node $u \in \mathcal{V}_q$:

$$m_{v \to u}(a_i) = \sum_{h \in \mathcal{F}_v \setminus u} m_{h \to v}(a_i) + c_{vu} \tag{8}$$

where $\mathcal{F}_v$ represents the node connected to v, and c_{vu} is the normalizing factor preventing the value of messages from growing arbitrarily large. The message from node u to v is:

$$m_{u \to v}(a_i) = \max_{\boldsymbol{a}_u \setminus a_v} \left[f(\boldsymbol{a}_u) + \sum_{h \in \mathcal{V}_u \setminus v} m_{h \to u}(a_h) \right] \tag{9}$$

where $\mathcal{V}_u$ is the set of nodes connected to node u, $\boldsymbol{a}_u = \{a_h | h \in \mathcal{V}_u\}$, $\boldsymbol{a}_u \setminus a_v = \{a_h | h \in \mathcal{V}_u \setminus \{v\}\}$, and f represents the value function of the (hyper-)edge. After iterations of message passing, each agent v can find its optimal action by calculating $a_v^* = \operatorname{argmax}_{a_v} \sum_{h \in \mathcal{F}_v} m_{h \to v}(a_v)$.

Test data generalization. We take $1,000$ random seeds to test the problem. We fix the number of actions 3 and move the number of agents from 2 to 18. For each setting, we generate a fully connected graph with each value a uniformly random number in $\{-1, 0, 1\}$ along with a noise drawn from the normal distribution.

B COMPUTATIONS WITH CLASSES $\mathcal{G}_P$ AND $\mathcal{G}_H$

Here we present the details of algorithms we used to select graph $G^{(t)}$ from the graph class $\mathcal{G}$.

When $\mathcal{G} = \mathcal{G}_P$, pairwise grouping. In this case, n agents are pairwise matched by $\lfloor n/2 \rfloor$ edges. One agent will be isolated if n is an odd number. With such a coordination graph, each agent coordinates with exactly one agent, except for the isolated one.

If node i is matched with node j, it will contribute $q_{i \to j} = q_i(\tau_i, a_i) + q_j(\tau_j, a_j) + q_{ij}(\tau_i, \tau_j, a_i, a_j)$ to the total value when choosing action a_i, which only contains the single utility function and one pairwise function. Thus the pair (i, j) can contribute at most

$$\max_{a_i, a_j} \left[q_i(\tau_i, a_i) + q_j(\tau_j, a_j) + q_{ij}(\tau_i, \tau_j, a_i, a_j) \right] \tag{10}$$

to the joint value without effect the action selection of other agents.

Hence, we construct a undirected weighted graph $G = \langle \mathcal{V}, \mathcal{E}_w \rangle$, where $\mathcal{V}$ is the vertex set, and

$$\mathcal{E}_w = \left\{ \left(i, j, w(i,j) = \max_{a_i, a_j} \left[q_i(\tau_i, a_i) + q_j(\tau_j, a_j) + q_{ij}(\tau_i, \tau_j, a_i, a_j) \right] \right) \middle| \forall (i,j) \in \mathcal{E} \right\}. \tag{11}$$

A matching on such graph is a function $mate(v)$ where $mate(mate(v)) = v$ for all $v \in \mathcal{V}$ with at most one v satisfies that $mate(v) = v$. The weight of the matching is defined as $\sum_v w(v, mate(v))$. Our goal is to find the maximum weighted matching on this graph.

This problem can be solved by the blossom algorithm (Edmonds, 1965) in $O(n^3)$ time. The brief steps of this algorithm are to iteratively find augmentation paths on the graph and contract the odd-cycle, which is called blossom in this algorithm.

When $\mathcal{G} = \mathcal{G}_H$, hierarchical organization. In this case, the graph is an acyclic graph with $n-1$ edges, and all agents form a connected component. The distributed constraint optimization problem (DCOP) on such a graph can always be solved in the polynomial time (Fioretto et al., 2018).

Choosing the best graph $G^{(t)}$ in $\mathcal{G}_H$ is non-trivial. Therefore, we present a greedy algorithm to select the graph from the graph class. The high level idea is to add edges greedily until all agents are connected. At each iteration, we add an edge across two different connected components to the coordination graph, which optimizes the increment of the maximal joint value. For an edge set $E \subseteq \mathcal{E}$, we define the value of a set $f(E)$ to be the result of DCOP on graph $\langle \mathcal{V}, E \rangle$. Initially we have E is an empty set. We do the iteration $n-1$ times. In each iteration, we find

$$e = \underset{e \in \mathcal{E}, E \cup \{e\} \text{ is acyclic}}{\arg\max} f(E \cup \{e\}), \tag{12}$$

and add this edge to the current graph, i.e., $E \leftarrow E \cup \{e\}$.

C Graph Relabeling Technique

As studied by Hasselt (2010), vanilla Q-learning already has a tendency to overestimate, since the TD target takes a max operator over a set of estimated action-values. Notice that our framework introduces an additional max operator over graphs in class $\mathcal{G}$, which will probably induce extra upward bias. To attenuate this problem, we use a graph relabeling technique to control the effects of overestimation errors. In our implementation, we modify the vanilla TD loss defined in Eq. (7) to the following form:

$$\mathcal{L}_{cg}(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{(\boldsymbol{\tau}, \boldsymbol{a}, r, \boldsymbol{\tau}') \sim \mathcal{D}} \left[\left(y_{cg} - \max_{G} Q(\boldsymbol{\tau}, \boldsymbol{a}; G; \boldsymbol{\theta}) \right)^2 \right] \tag{13}$$

where y_{cg} is the one-step TD target defined in Section 4.2 and the current value no longer takes the graph which is stored in the replay buffer with this transition. As G is the graph with largest estimated value, this loss achieves a soft constraint for the following target:

$$\forall G, Q(\boldsymbol{\tau}, \boldsymbol{a}; G; \boldsymbol{\theta}) \leq y_{cg} \tag{14}$$

which helps our method avoid overestimation.

D Implementation Details And Experiment Settings

A shared GRU (Cho et al., 2014) is used to process sequential inputs and then output the encoded observation history for each agent. A following fully-connected layer converts the observation histories to the individual utility function of each agent. The pairwise payoff function is computed by another multi-layer perceptron, which takes the concatenation of two agents' observation histories as input. Agents share parameters of the network for computing individual utility, and the parameters of the payoff network are also shared among different agent pairs.

All tasks in this paper use a discount factor $\gamma = 0.99$. We use ϵ-greedy exploration, and ϵ anneals linearly from 1.0 to 0.05 over 50000 time-steps. We use an RMSProp optimizer with a learning rate of 5×10^{-3} to train our network. A first-in-first-out (FIFO) replay buffer stores the experiences of at most 5000 episodes, and a batch of 32 episodes are sampled from the buffer during the training phase. The target network is periodically updated every 200 episodes.

Our method and all the baselines involved in this paper are implemented based on the open-sourced codebase PyMARL (Samvelyan et al., 2019), which ensures the fairness of comparison. For the baselines (VDN (Sunehag et al., 2018), QMIX (Rashid et al., 2020b), DCG (Böhmer et al., 2020), CASEC (Wang et al., 2021b)), we adopt the default hyperparameter settings provided by the authors. For evaluation, all learning curves in this paper are smoothed over 5 random seeds.

The experiments are finished on NVIDIA RTX 2080TI GPU. Training our algorithm on any task shown in this paper can be finished within 2.5 days.