Self-Organized Polynomial-Time Coordination Graphs

Qianlan Yang [*1] **Weijun Dong** [*1] **Zhizhou Ren** [*2] **Jianhao Wang** [1] **Tonghan Wang** [3] **Chongjie Zhang** [1]

Abstract

Coordination graph is a promising approach to model agent collaboration in multi-agent reinforcement learning. It conducts a graph-based value factorization and induces explicit coordination among agents to complete complicated tasks. However, one critical challenge in this paradigm is the complexity of greedy action selection with respect to the factorized values. It refers to the decentralized constraint optimization problem (DCOP), which and whose constant-ratio approximation are NP-hard problems. To bypass this systematic hardness, this paper proposes a novel method, named Self-Organized Polynomial-time Coordination Graphs (SOP-CG), which uses structured graph classes to guarantee the accuracy and the computational efficiency of collaborated action selection. SOP-CG employs dynamic graph topology to ensure sufficient value function expressiveness. The graph selection is unified into an end-to-end learning paradigm. In experiments, we show that our approach learns succinct and well-adapted 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. In recent years, the popular MARL algorithms achieve scalability by deploying fully decentralized control policies, i.e., the behavior of each agent only depends on its local observation history. Due to the lack of explicit coordination, these methods may suffer from a game-theoretic pathology called relative overgeneralization (Panait et al., 2006; Wei & Luke, 2016; Böhmer et al., 2020): if the values of coordinated and uncoordinated actions are not distinguished during exploration, the learning agents can jointly converge to suboptimal behaviors. Coordination graph (Guestrin et al., 2001) provides a promising class of algorithms to address this issue. It conducts a graph-based value factorization to explicitly represent coordination relations among agents. Each vertex in the graph corresponds to an agent, and each (hyper-) edge defines a local utility function over the joint action space of the connected agents. With such a higher-order value factorization, agents communicate to jointly optimize their actions, which helps to prevent relative overgeneralization. Despite this advantage, coordination graph would require more complex greedy action selection and extra communication cost than using fully decentralized policies. This paper mainly focuses on the challenges in greedy action selection of coordination graph.

One fundamental problem for coordination graphs is the trade-off between the representational capacity of value functions and the computational complexity of policy execution, where the latter would hinder the scalability of these algorithms. 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 approximation are NP-hard problems, especially for the complete graphs used in DCG (Dagum & Luby, 1993; Park & Darwiche, 2004). From the perspective of theoretical computer science, any scalable heuristic algorithm (e.g., max-sum algorithm (Pearl, 1988)) for DCOP may have an uncontrollable gap with the optimal solution.

To visualize this issue, we generate a suite of complete-graph DCOPs with random edge-values. Fig. 1 presents

[1]Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University [2]Department of Computer Science, University of Illinois at Urbana-Champaign [3]Harvard University. Correspondence to: Chongjie Zhang <chongjie@tsinghua.edu.cn>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

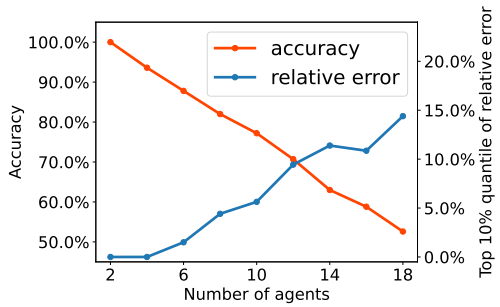

Figure 1. The accuracy and the relative joint Q-error of the max-sum algorithm w.r.t. the number of agents.

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). 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. More detailed experiment setting are deferred to to Appendix A. It further refers to 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. An open question in MARL raises:

How to incorporate accurate greedy action selection with sufficient function representational capacity for coordination graph algorithms?

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 (1) utilizes structured graph topologies with polynomial-time guarantees for DCOPs and (2) extends their representation expressiveness through a dynamic graph organization mechanism. These two characteristics open up a new family of MARL algorithms. More specifically, we first construct graph classes guaranteeing polynomial-time DCOPs and then conduct a state-dependent graph selection mechanism. Such a dynamic factorization structure is achieved by communications among agents and can be self-organized through end-to-end learning.

In experiments, we evaluate SOP-CG on the MACO benchmark (Wang et al., 2021b), particle environment (Lowe et al., 2017) and StarCraft II (Samvelyan et al., 2019). SOP-CG demonstrates superior performance on tasks characterizing relative overgeneralization pathology, highlighting the effectiveness of polynomial-time coordination graphs. By extensive ablation studies, we verify that the dynamic graph selection mechanism is critical for our performance. 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

We consider 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 $D = \{1, \dots, n\}$ 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 $\boldsymbol{a} = (a_1 \dots, a_n)$, 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 $\tau_{i,t} = (o_{i,0}, a_{i,0}, \dots, o_{i,t-1}, a_{i,t-1}, o_{i,t}) \in \left(O^i \times A^i\right)^t \times O^i$.

Deep Q-Learning. Q-learning 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 $\boldsymbol{\tau}'$. $Q(\boldsymbol{\tau}, \boldsymbol{a}; \boldsymbol{\theta})$ 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 undirected 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:

$$\begin{aligned} Q_{tot}(s, \boldsymbol{a}) = & \sum_{v_i \in \mathcal{V}} f_i(s, a_i) \\ & + \sum_{\boldsymbol{e} = \{e_1, \dots, e_{|\boldsymbol{e}|}\} \in \mathcal{E}} f_{\boldsymbol{e}}(s, a^{e_1}, \dots, a^{e_{|\boldsymbol{e}|}}), \end{aligned} \tag{2}$$

where f_i represents the individual utility of agent i and $f_{\boldsymbol{e}}$ specifies the payoff contribution for the actions of the agents connected by the (hyper-)edge $\boldsymbol{e}$, so that the global optimal solution can be found through maximizing this joint value. The special case that $\mathcal{E}$ is an empty set yields VDN (Sunehag et al., 2018), 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 actions 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 (e.g., message passing methods) 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 the *polynomial-time coordination graph class*, 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 specific coordination graphs, which can hold the accurate greedy action selection 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. Fig. 2 illustrates the overall framework of our approach.

4.1. Value Factorization on Polynomial-Time Coordination Graphs

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

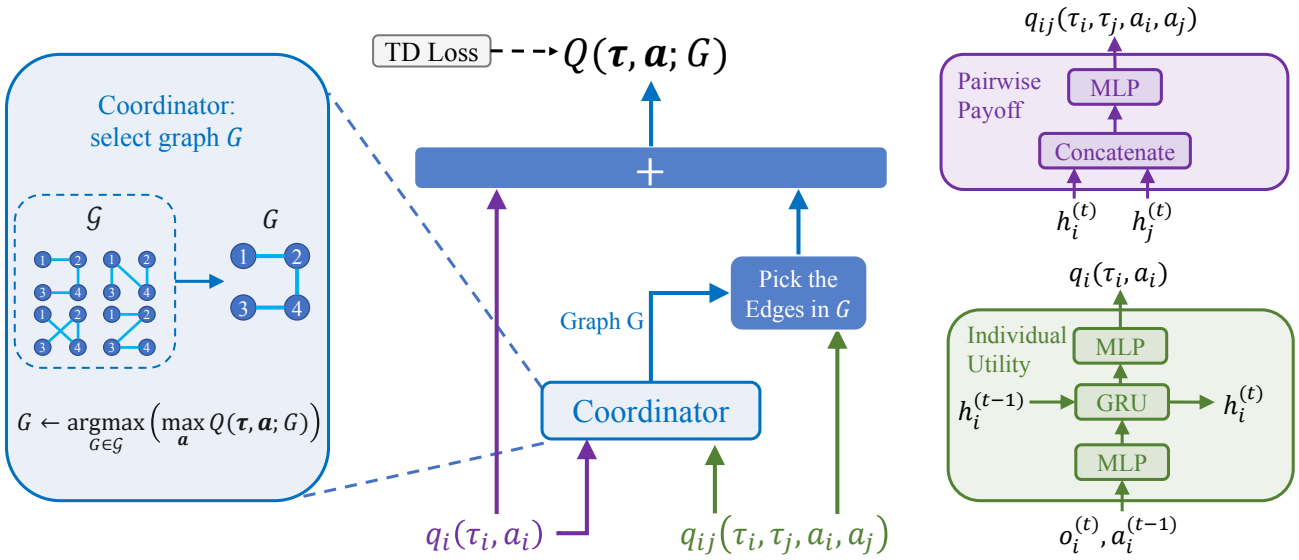

Figure 2. An overview of SOP-CG's framework. **Left**: An imaginary coordinator agent that selects graph G from the predefined graph class $\mathcal{G}$. **Middle**: The overall working flow. **Right**: Individual utility network structure (bottom) and pairwise payoff network structure (top).

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. With this second-order value decomposition, the hardness of greedy action selection is highly related to the graph topology. To provide a clear perspective, we discuss the theoretical complexity of the DCOPs induced by different topology classes.

Proposition 4.1. *Let n be the number of agents and $A = |\bigcup_{i=1}^{n} A^i|$. (i) Approximating the induced DCOPs of fully connected coordination graphs within a factor of $O\left(\frac{\log A}{\sqrt{A}}\right)$ is NP-hard. (ii) There exists an algorithm that can compute the accurate solution of any DCOP induced by an undirected acyclic graph in Poly(n, A) running time.*

Detailed proof can be found in Appendix B. Notebly, DCG adopts the fully connected coordination graph as the default implementation. Due to the theoretical hardness of its induced DCOP, using any polynomial-time approximation algorithm may result in unfavorable quality of the selected actions. To avoid this problem, we propose to deploy *Polynomial-Time Coordination Graph Classes*. Now we formulate the definition as follows:

Definition 4.2 (Polynomial-Time Coordination Graph Class)**.** We say a graph class $\mathcal{G}$ is a *Polynomial-Time Coordination Graph Class* if there exists an algorithm that can solve any induced DCOP of any coordination graph $G \in \mathcal{G}$ in Poly(n, A) running time.

As stated in Proposition 4.1, the set of undirected acyclic graphs $\mathcal{G}_{\text{uac}}$ is a polynomial-time coordination graph class. However, an undirected acyclic graph can contain at most $n-1$ edges in an environment with n agents, which suffers from the lack of function expressiveness.

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 the 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 the accuracy of greedy action selections.

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 suboptimality of performance 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 sub-optimality of performance 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 the 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):

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

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.

The imaginary agent is only introduced to better demonstrate our idea. In practice, it is not necessary to explicitly employ a coordinator. The agents can jointly select the graph topology and coordinate their actions through communications. This communication mechanism is consistent with previous coordination-graph-based methods like DCG and CASEC (Wang et al., 2021b).

Training. With the imaginary coordinator, we can reformulate the Bellman optimality equation and maximize 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 associated value $Q(\boldsymbol{\tau}, \boldsymbol{a}; G)$ of graph G can be updated through temporal difference learning. The network parameters $\boldsymbol{\theta}$ can be trained 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 $y_{cg} = r + \gamma \max_{(\boldsymbol{a}', G')} Q(\boldsymbol{\tau}', \boldsymbol{a}'; G'; \boldsymbol{\theta}^-)$ 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.

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)
      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 in replay buffer $\mathcal{D}$
    end for
    Sample random minibatch of transitions from $\mathcal{D}$
    Perform a gradient descent step on loss defined in Eq. (7)
  end for

The overall algorithm of our approach is summarized in Algorithm 1. The graph selection step 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.
- *Tree organization* $\mathcal{G}_T$. 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

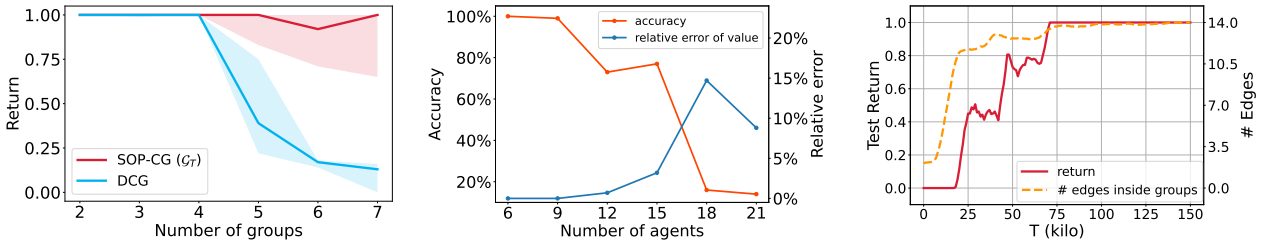

Figure 3. Illustrative example. **Left**: Final performance of SOP-CG ($\mathcal{G}_T$) and DCG when $k = 2$ to 7. The optimal return is 1. **Middle**: Accuracy of DCG's action selection, and the relative error between the joint values computed by the taken actions and the optimal actions. **Right**: Number of edges inside groups selected by SOP-CG ($\mathcal{G}_T$) when $k = 7$. A graph in $\mathcal{G}_T$ can include at most 14 edges which connects agents in the same group.

Graph Class	Select graph $G^{(t)}$	Select actions on a given graph
$\mathcal{G}_P$	$\surd$	$\surd$
$\mathcal{G}_T$	–	$\surd$
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.

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.

$\mathcal{G}_T$ 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 computations 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 $\mathcal{G}_T$ is non-trivial, so we design a greedy algorithm to compute approximated solutions. Note that, although the graph selection of $\mathcal{G}_T$ 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, $\mathcal{G}_T$ can still work as a promising graph class in absence of rigorous graph selection. Please refer to Appendix D for the implementation details of the induced combinatorial optimization for graph selection.

Remark. Regarding the integration of these graph classes and the maximization criterion introduced in Eq. (5), we have not derived strong convergence guarantees of the graph structure and the value functions. Despite this limitation, the criterion works well and can produce interpretable graph topologies that reflect the ground-truth problem structure, as we will show in Section 5. A future direction is to explore other graph organization criteria specializing in theoretical properties, e.g., developing systematical analysis and improving the convergence rate.

5. Experiments

In this section, we conduct experiments to answer the following questions: (1) Does the accurate greedy action selection improve our performance (see Section 5.1 and 5.2)? (2) Is the dynamic graph organization mechanism necessary for our algorithm (see Section 5.1 and 5.3)? (3) How well does SOP-CG perform on complex cooperative multi-agent tasks (see Fig. 4, 5 and 6)? (4) Can SOP-CG extract interpretable dynamic coordination structures in complex scenarios (see Section 5.4)? For evaluation, all results in this section are illustrated with the median performance over 5 random seeds as well as the 25%-75% percentiles. An open-source implementation of our algorithm is available online[1].

5.1. Illustrative Example

To illustrate the effects of the accuracy of greedy action selection in DCG and SOP-CG, we devise a simple one-step cooperative game for $3 \times k$ agents with two actions.

The $3 \times k$ agents will be randomly separated into k groups with each group 3 agents. Action A is available to all the agents, while action B is only available to a random set of agents. Choosing action B will yield a cost of 0.5. If three agents in the same group take action B, they will receive a global reward of 2.5. Therefore, agents in the same group should coordinate to decide whether to jointly take action A or action B.

We set the number of groups from 2 to 7 and examine the final performance of DCG and SOP-CG after 150000 episodes training. As shown in Fig. 3-Left, SOP-CG can

[1] https://github.com/yanQval/SOP-CG.

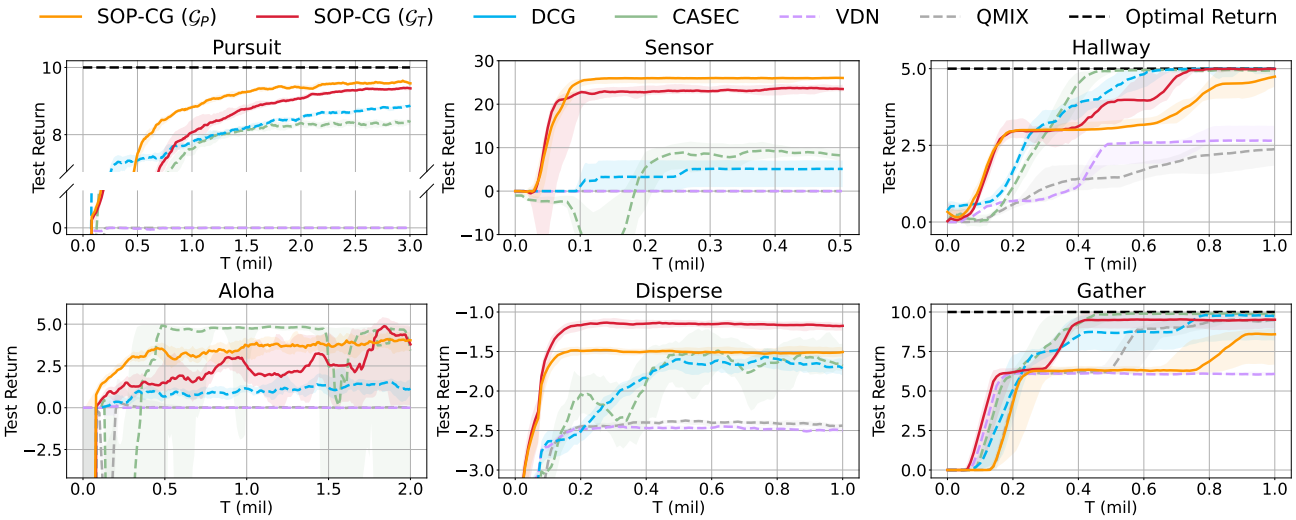

Figure 4. Learning curves on MACO benchmark. The optimal return is demonstrated if it is available.

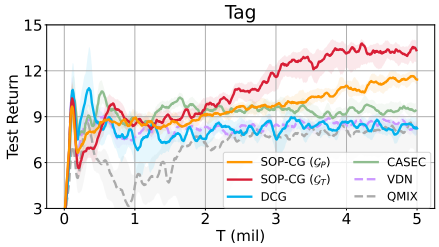

Figure 5. Learning curves on Tag.

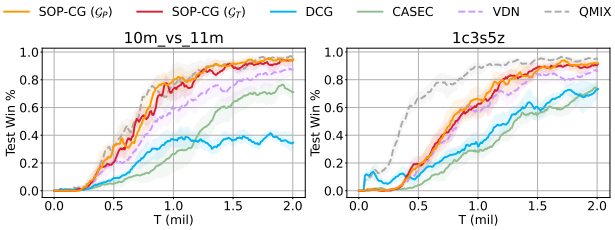

Figure 6. Learning curves on SMAC tasks.

keep achieving the optimal return, while DCG is more sensitive to the total number of agents and obtains low return when $k \geq 5$. For DCG, we further test the accuracy of its greedy action selection as well as the relative error between the joint values computed by the taken actions and the optimal actions. Fig. 3-Middle demonstrates that the accuracy plummets when the number of agents grows. This problem directly limits the performance of DCG, whereas SOP-CG does not suffer from it.

Meanwhile, we empirically show that our method could learn well-adapted topologies during the training. In this game, each agent only needs to coordinate with others in the same group. To better understand the contribution of self-organized graph selection, we visualize the number of edges which connects agents in the same group in Fig. 3-Right. It continually increases during the training, while the test performance rises along with it and finally reaches the optimal value after 14 such edges are chosen, which is the maximum number of edges inside groups that a graph in $\mathcal{G}_T$ can cover. This phenomenon proves the ability of our method to learn better policy through discovering the ground truth coordination requirements in the environment.

The full details of the illustrative example are provided in Appendix E.

5.2. Performance Comparison

In this section, we test SOP-CG on multi-agent coordination challenge (MACO; Wang et al., 2021b), multi-agent particle environment (MPE; Lowe et al., 2017) and StarCraft multi-agent challenge (SMAC; Samvelyan et al., 2019). We compare SOP-CG with the state-of-the-art coordination graph learning methods (DCG (Böhmer et al., 2020), CASEC (Wang et al., 2021b)) and fully decomposed value-based methods (VDN (Sunehag et al., 2018), QMIX (Rashid et al., 2020b)). The implementation details and hyperparameter settings are included in Appendix F.

MACO (Wang et al., 2021b) is a benchmark consisting of 6 classic coordination problems, raising challenges of relative overgeneralization and partial observability. Results are illustrated in Fig. 4. SOP-CG outperforms the baselines on three tasks, especially Pursuit and Sensor. Our method learns almost the optimal policy on Pursuit while baselines have over two times larger gap with the optimal return than ours. On Sensor, the return achieved by our method is over twice the best baseline's return. The major difficulties on these tasks lie in *relative overgeneralization pathology* (Panait et al., 2006; Wei & Luke, 2016; Böhmer et al., 2020), since the uncoordinated actions will lead to high punishments. Even if an agent takes the same action,

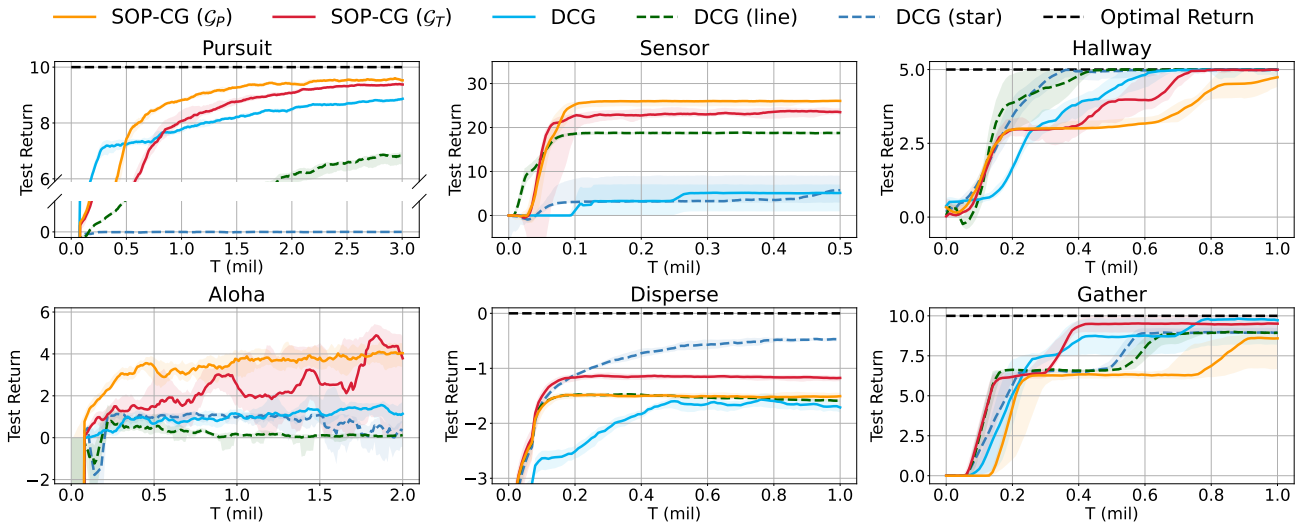

Figure 7. Ablation study on MACO benchmark. *Line* and *Star* are two special instances of the polynomial-time graph class. The optimal return is demonstrated if it is available.

the reward observed by it is highly related to the actions of other agents and may be drastically changeable due to their exploration, which hinders the credit assignment of fully decentralized methods. Although the explicit modeling of agent interactions helps the coordination-graph-based methods address this issue, accurate action selection is also critical for learning optimal policy. When facing growing scales (there are 20 agents on Pursuit[2] and 15 agents on Sensor), DCG and CASEC have a poor accuracy on DCOP, which may lead to inaccurate greedy action selection, impeding them to better performance. In contrast, our method converges to the superior values, highlighting the effectiveness of polynomial-time coordination graphs.

Tag is a task based on the particle world (Lowe et al., 2017). 10 agents chase 3 adversaries on the map with 3 randomly generated obstacles, and the agents receive a global reward for each collision between an agent and an adversary. Since the adversaries move faster, it requires fine-grained control and collaboration for the agents to surround the adversaries. As shown in Fig. 5, SOP-CG ($\mathcal{G}_T$) significantly outperforms the baselines on this task. In Section 5.4, we will visualize the coordination structures learned by our method to justify the adaptability of the dynamic graph organization mechanism.

We further test SOP-CG on the SMAC benchmark to demonstrate its scalability in large action-observation spaces. Notably, the number of output heads of pairwise payoff functions grows quadratically with the number of actions. In this environment, DCG and CASEC adopt different techniques (e.g., low-rank approximation or action representation learning) to improve the learning efficiency of the payoff functions, which can also be applied to our algorithm. For fair comparison, we equip SOP-CG and the coordination-graph-based baselines (i.e., DCG and CASEC) with action representation learning. Fig. 6 illustrates the results on the tasks 10m_vs_11m and 1c3s5z. SOP-CG outperforms both DCG and CASEC by a large margin.

5.3. Ablation Study

In this section, we carry out ablation studies to justify the contribution of the self-organized dynamic topology. We test alternative implementations of DCG on two static graphs *Line* (agent i is connected with agent $i-1$ and agent $i+1$) and *Star* (agent 1 is linked with others). These graphs are two special instances of $\mathcal{G}_{uac}$, implying polynomial-time solvable DCOPs. Results are shown in Fig. 7. Although the function expressiveness is restricted, it is observed that DCG (line) and DCG (star) can perform much better than DCG in some cases (i.e., DCG (line) on Sensor and DCG (star) on Disperse). We hypothesize that this is because these topologies induce efficient cooperation strategy or directly match the ground truth coordination demands on such domains, and the accurate greedy action selection ensures them to achieve superior performance. For example, agents in Sensor are arranged as a 3×5 matrix in order of their ids. As adjacent agents need to coordinate to scan a nearby target, *Line* covers the major coordination requirements and renders a satisfactory policy. However, these methods have poor performance on other tasks, which may not be characterized by such static coordination graphs. In contrast, SOP-CG

[2]Pursuit is also called Predator-Prey (Son et al., 2019; Rashid et al., 2020a). MACO benchmarks Pursuit with 10 agents and reward : penalty = 1 : 1. In this paper, we test a harder version with 20 agents.

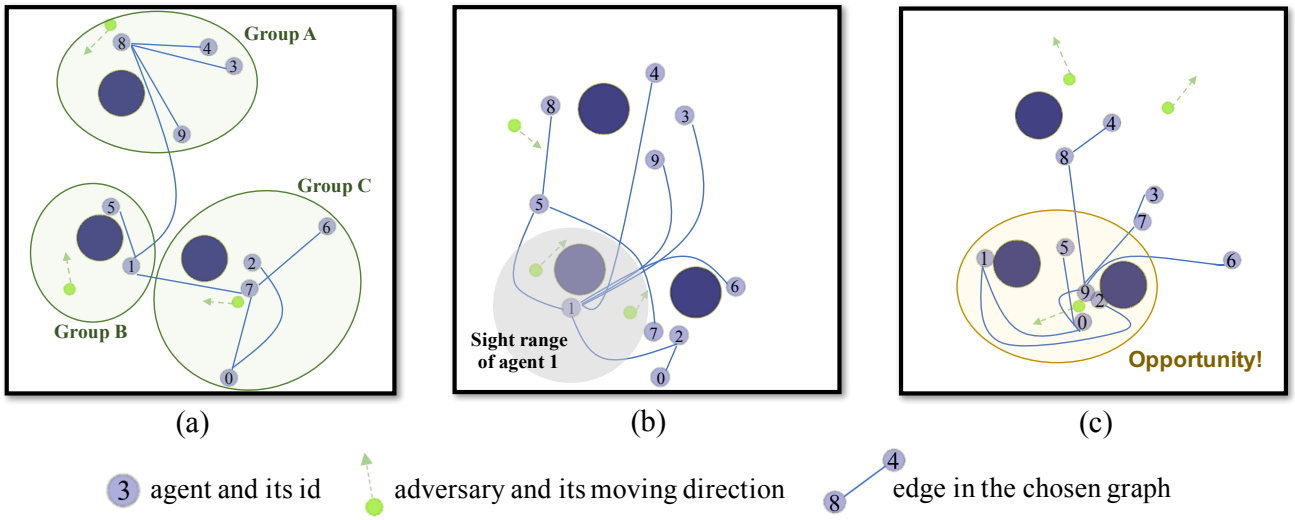

Figure 8. Various coordination graphs learned by our algorithm on Tag: (a) Self-organized grouping at initialization; (b) Connecting to agent with rich observation for better information sharing; (c) Concentrated collaboration structure around an enclosed adversary.

achieves good performance across all tasks in MACO by virtue of self-adaptive topologies.

5.4. 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 Tag. In Fig. 8, we take three snapshots chronologically from one episode. This episode is collected by SOP-CG ($\mathcal{G}_T$) 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. 8-a 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. 8-b, 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 them recognize the position of the adversaries.

3. Finally, as illustrated by Fig. 8-c, 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 can learn effective coordination patterns in Tag, which can be explained by the oracle collaboration strategy of human experts. It demonstrates the ability of our approach to organize 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.

7. Acknowledgement

This work is supported in part by Science and Technology Innovation 2030 – "New Generation Artificial Intelligence" Major Project (No. 2018AAA0100904) and National Natural Science Foundation of China (62176135).

References

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

Böhmer, W., Kurin, V., and Whiteson, S. Deep coordination graphs. In *International Conference on Machine Learning*, pp. 980–991. PMLR, 2020.

Castellini, J., Oliehoek, F. A., Savani, R., and Whiteson, S. 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.

Chan, S. O. Approximation resistance from pairwise-independent subgroups. *Journal of the ACM (JACM)*, 63(3):1–32, 2016.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

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

Das, A., Gervet, T., Romoff, J., Batra, D., Parikh, D., Rabbat, M., and Pineau, J. Tarmac: Targeted multi-agent communication. In *International Conference on Machine Learning*, pp. 1538–1546, 2019.

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

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

Foerster, J., Nardelli, N., Farquhar, G., Afouras, T., Torr, P. H., Kohli, P., and Whiteson, S. Stabilising experience replay for deep multi-agent reinforcement learning. In *International conference on machine learning*, pp. 1146–1155. PMLR, 2017.

Foerster, J. N., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. Counterfactual multi-agent policy gradients. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Guestrin, C., Koller, D., and Parr, R. Multiagent planning with factored mdps. In *NIPS*, volume 1, pp. 1523–1530, 2001.

Guestrin, C., Lagoudakis, M., and Parr, R. Coordinated reinforcement learning. In *ICML*, volume 2, pp. 227–234. Citeseer, 2002a.

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

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

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

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

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

Li, S., Gupta, J. K., Morales, P., Allen, R., and Kochenderfer, M. J. Deep implicit coordination graphs for multi-agent reinforcement learning. *arXiv preprint arXiv:2006.11438*, 2020.

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

Manurangsi, P., Nakkiran, P., and Trevisan, L. Near-optimal ugc-hardness of approximating max k-csp_r. *arXiv preprint arXiv:1511.06558*, 2015.

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

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

Panait, L., Luke, S., and Wiegand, R. P. Biasing coevolutionary search for optimal multiagent behaviors. *IEEE Transactions on Evolutionary Computation*, 10(6):629–645, 2006.

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

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

Rashid, T., Farquhar, G., Peng, B., and Whiteson, S. Weighted qmix: Expanding monotonic value function factorisation for deep multi- agent reinforcement learning. *NeurIPS Proceedings 2020*, 2020a.

Rashid, T., Samvelyan, M., De Witt, C. S., Farquhar, G., Foerster, J., and Whiteson, S. Monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv preprint arXiv:2003.08839*, 2020b.

Samvelyan, M., Rashid, T., de Witt, C. S., Farquhar, G., Nardelli, N., Rudner, T. G. J., Hung, C.-M., Torr, P. H. S., Foerster, J., and Whiteson, S. The StarCraft Multi-Agent Challenge. *CoRR*, abs/1902.04043, 2019.

Singh, A., Jain, T., and Sukhbaatar, S. Learning when to communicate at scale in multiagent cooperative and competitive tasks. *arXiv preprint arXiv:1812.09755*, 2018.

Son, K., Kim, D., Kang, W. J., Hostallero, D. E., and Yi, Y. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 5887–5896, 2019.

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

Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K., 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.

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

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

Wang, J., Ren, Z., Liu, T., Yang, Y., and Zhang, C. Qplex: Duplex dueling multi-agent q-learning. In *International Conference on Learning Representations*, 2021a.

Wang, T., Dong, H., Lesser, V., and Zhang, C. Multi-agent reinforcement learning with emergent roles. In *International Conference on Machine Learning*, 2020a.

Wang, T., Gupta, T., Mahajan, A., Peng, B., Whiteson, S., and Zhang, C. Rode: Learning roles to decompose multi-agent tasks. *arXiv preprint arXiv:2010.01523*, 2020b.

Wang, T., Wang, J., Zheng, C., and Zhang, C. Learning nearly decomposable value functions via communication minimization. In *International Conference on Learning Representations*, 2020c.

Wang, T., Zeng, L., Dong, W., Yang, Q., Yu, Y., and Zhang, C. Context-aware sparse deep coordination graphs. *arXiv preprint arXiv:2106.02886*, 2021b.

Wang, Y., Han, B., Wang, T., Dong, H., and Zhang, C. Off-policy multi-agent decomposed policy gradients. *arXiv preprint arXiv:2007.12322*, 2020d.

Wei, E. and Luke, S. Lenient learning in independent-learner stochastic cooperative games. *The Journal of Machine Learning Research*, 17(1):2914–2955, 2016.

Wen, Y., Yang, Y., Luo, R., Wang, J., and Pan, W. Probabilistic recursive reasoning for multi-agent reinforcement learning. *arXiv preprint arXiv:1901.09207*, 2019.

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

Yoon, S.-e., Song, H., Shin, K., and Yi, Y. How much and when do we need higher-order information in hypergraphs? a case study on hyperedge prediction. In *Proceedings of The Web Conference 2020*, pp. 2627–2633, 2020.

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

Zhang, C. and Lesser, V. 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. Experiment 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 work, 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 $u \in \mathcal{V}_q$:

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

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:

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

where $\mathcal{V}_u$ is the set of nodes connected to node u, $\boldsymbol{a}_u = \{a_h | h \in \mathcal{V}_u\}$, $\boldsymbol{a}_u \backslash a_v = \{a_h | h \in \mathcal{V}_u \backslash \{v\}\}$, 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 $a_v^* = \text{argmax}_{a_v} \sum_{h \in \mathcal{F}_v} m_{h \to v}(a_v)$.

Experiment setup. We take 1000 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. Proof of Proposition 4.1

In this section, we provide the proof of Proposition 4.1 (i). Proposition 4.1 (ii) is given in the literature (Fioretto et al., 2018).

We will reduce Max 2CSP-R (Manurangsi et al., 2015) to the DCOPs induced by fully connected graphs. Max 2CSP-R is a constraint satisfaction problem with two variables per constraint, where each variable can take values from a finite set Σ of size R, and the goal is to find an assignment to variables that maximizes the number of satisfied constraints.

For each variable in the Max 2CSP-R problem, we create an agent in the coordination graph with R actions. Furthermore, for each constraint in the Max 2CSP-R problem, we create an edge in the coordination graph, connecting the two agents of the variables, with a function value 1 only when the constraint is satisfied and otherwise 0. If we can approximate the induced DCOP problem within a factor of $c\frac{\log R}{\sqrt{R}}$, we can distinguish the $(1-\epsilon)$-satisfiable Max 2CSP-R instances and the instances in which at most a $c\frac{\log R}{\sqrt{R}}$ fraction of constraints are satisfiable. Chan (2016) proves that there is a constant c such that for every sufficiently large R and any $\epsilon > 0$, it is NP-hard to distinguish $(1-\epsilon)$-satisfiable instances of Max 2CSP-R from instances in which at most a $c\frac{\log R}{\sqrt{R}}$ fraction of constraints are satisfiable. Therefore, it is NP-hard to Approximate the induced DCOPs of fully connected coordination graphs within a factor of $O\left(\frac{\log A}{\sqrt{A}}\right)$. Results given by Manurangsi et al. (2015) also show that we can get a tighter approximation hardness bound of $O\left(\frac{\log A}{A}\right)$ if assuming the unique games conjecture.

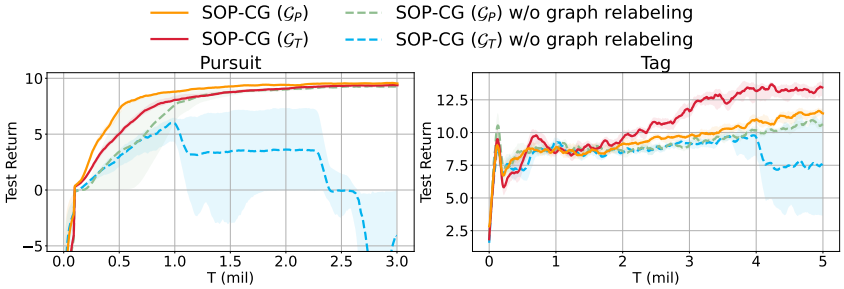

Figure 9. Ablation studies about the graph relabeling technique.

C. Implementation tricks of Temporal Difference Learning

As studied by previous work (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:

$$\mathcal{L}_{cg}(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{(\boldsymbol{\tau},\boldsymbol{a},r,\boldsymbol{\tau}')\sim\mathcal{D}}\left[\left(y_{cg} - \max_{G} Q(\boldsymbol{\tau},\boldsymbol{a};G;\boldsymbol{\theta})\right)^2\right] \tag{10}$$

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:

$$\forall G, Q(\boldsymbol{\tau},\boldsymbol{a};G;\boldsymbol{\theta}) \leq y_{cg} \tag{11}$$

which helps our method avoid overestimation.

We conduct ablation studies on the tasks Pursuit and Tag to demonstrate the necessity of this component. The results illustrated in Fig. 9. SOP-CG w/o graph relabeling is more likely to stick in suboptimal strategies and may even collapse on Pursuit. As analyzed before, these phenomenons are due to the dramatic overestimation caused by the additional max operator over graphs. By contrast, the performance of SOP-CG is significantly improved, highlighting the effectiveness of graph relabeling.

D. Computations with Classes $\mathcal{G}_P$ and $\mathcal{G}_T$

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, they will contribute at most

$$\max_{a_i,a_j}\left[q_i(\tau_i,a_i) + q_j(\tau_j,a_j) + q_{ij}(\tau_i,\tau_j,a_i,a_j)\right] \tag{12}$$

to the joint value without affecting 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

$$\mathcal{E}_w = \left\{ \left(i, j, w(i,j) = \max_{a_i,a_j}\left[q_i(\tau_i,a_i) + q_j(\tau_j,a_j) + q_{ij}(\tau_i,\tau_j,a_i,a_j)\right] \right) \middle| \forall (i,j) \in \mathcal{E} \right\}. \tag{13}$$

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 $\frac{1}{2}\sum_v w(v, mate(v))$. 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 $\mathcal{G} = \mathcal{G}_T$ (tree 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 graphs can always be solved in the polynomial time (Fioretto et al., 2018).

Choosing the best graph $G^{(t)}$ in $\mathcal{G}_T$ 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

$$e = \underset{e \in \mathcal{E}, E \cup \{e\} \text{ is acyclic}}{\arg\max} f(E \cup \{e\}), \tag{14}$$

and add this edge to the current graph, i.e., $E \leftarrow E \cup \{e\}$. Once a graph from $\mathcal{G}_T$ is selected, the joint action selection can be computed via dynamic programming.

Time complexity. We summarize the time complexity (in worst case) of each component in Table 2. For SOP-CG ($\mathcal{G}_T$), we use pruning techniques in our implementation, so the real-time cost of selecting graph $G^{(t)}$ is usually much smaller than the complexity reported in Table 2. The detailed training time cost of SOP-CG and DCG can be found in Appendix F.

Algorithm	Select graph $G^{(t)}$	Select actions on a given graph
SOP-CG ($\mathcal{G}_P$)	$O(n^3)$	$O(nA^2)$
SOP-CG ($\mathcal{G}_T$)	$O(n^3A^2)$ (heuristic)	$O(nA^2)$
DCG	N/A	$O(kn^2A^2)$ (heuristic)

Table 2. Time complexity of SOP-CG and DCG. n is the number of agents and $A = |\bigcup_{i=1}^{n} A^i|$. For DCG, k is the number of iterations in max-sum.

E. Settings and Results of the Illustrative Example

Experiment Setting. In this illustrative example, each agent performs independent ϵ-greedy exploration, with ϵ anneals linearly from 1.0 to 0.05 over 100000 time-steps. We set $\gamma = 0.99$. The replay buffer consists of the last 500 episodes, from which we uniformly sample a batch of size 32 for training.

The group allocation is drawn from the uniform distribution, with each time only one group having three agents can play action B. The group id is one-hot encoded. Each agent receives the group id and its available actions as input.

Detailed results. Fig. 11 shows the full result of our method and DCG (line, star and full) on this environment.

We also test an extended version of the illustrative example. In this version, each group has four agents. Choosing action B will still yield a cost of 0.5. If all agents in the same group take action B, they will receive a global reward of 3. Therefore, this version requires coordination among 4-tuples of agents. As shown in Figure 10, SOP-CG ($\mathcal{G}_T$) still works well.

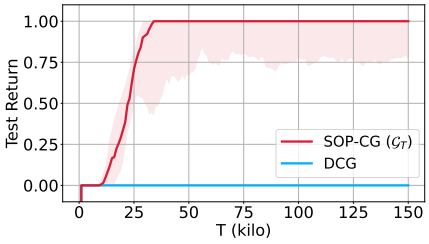

Figure 10. Extension of the illustrative example with 5 groups and each group 4 agents.

F. Experiment Settings and Hyperparameters

F.1. Settings of Experiment Tasks

MACO (Wang et al., 2021b) is a benchmark including 6 classic tasks: Aloha, Pursuit, Hallway, Sensor, Gather and Disperse. For the task Pursuit, MACO uses the version with 10 predators and 5 prey. In this paper, we use a harder version with 20

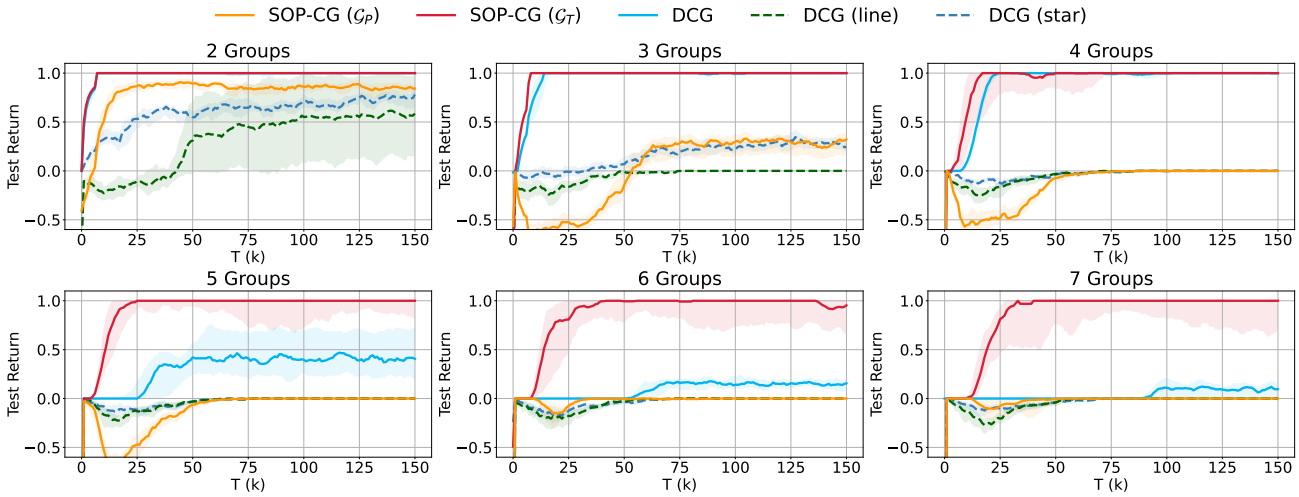

Figure 11. Learning curves on the illustrative example.

predators and 10 prey. For the other tasks, we test the vanilla version in MACO. The number of agents and actions are listed in Table 3.

	Aloha	Pursuit	Hallway	Sensor	Gather	Disperse
# agents	10	20	12	15	5	12
# actions	2	9	3	9	5	4

Table 3. Number of agents and actions in the tasks from MACO.

Tag is a challenging task on the particle world (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. Each agent has 5 actions to control its movement. 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, while the adversaries are concurrently trained by VDN (Sunehag et al., 2018).

SMAC (Samvelyan et al., 2019) is a popular benchmark based on the strategy game StarCraft II. In this paper, we benchmark our algorithm with StarCraft version of 2.4.6.2.69232, which is used by the SMAC paper.

F.2. Hyperparameters

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.

The experiments are finished on NVIDIA RTX 2080TI GPU. The training time of SOP-CG and DCG are listed in Table 4. SOP-CG runs faster than DCG.

	Pursuit	Tag	1c3s5z
Environmental time cost	2h	6h	1.5h
DCG	8.5h	12.5h	7h
SOP-CG ($\mathcal{G}_P$)	6h	9h	5h
SOP-CG ($\mathcal{G}_T$)	8h	9.5h	6.5h

Table 4. Time cost (hours) of training SOP-CG and DCG for 1M steps. Environmental time cost denotes the total time taken by the environment simulator and the training of the adversaries.

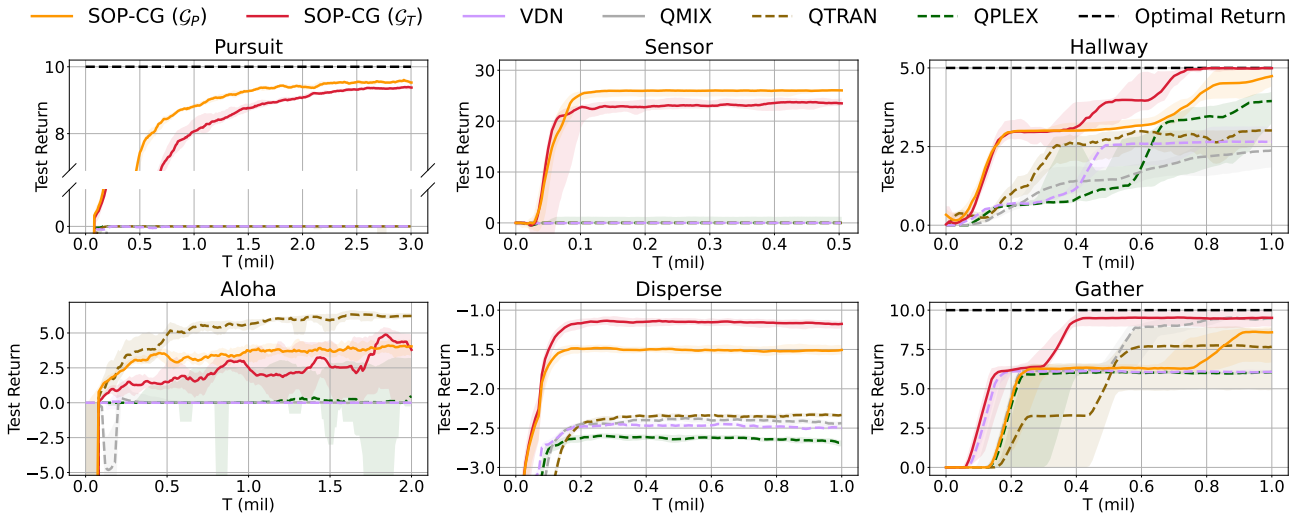

Figure 12. Comparison to fully decomposed value-based methods on MACO.

G. Additional Results on MACO

G.1. Comparison to Fully Decomposed Value-Based Methods

We compare our algorithm with two additional fully decomposed value-based methods: QTRAN (Son et al., 2019) and QPLEX (Wang et al., 2021a). These methods achieve more expressive value function classes than VDN and QMIX. The performance is shown in Figure 12.

G.2. Comparison to a Method Using the Underlying Problem Structure

We test DCG with a fixed static graph that reflects the underlying problem structure of Aloha. As shown in Figure 13-Left, the peak performance of this method is similar to SOP-CG's. Notice that max-sum algorithm is not guaranteed to get accurate DCOP solutions on this underlying graph. If we further replace max-sum with an exhaustive search (which will take much higher time complexity), it achieves better peak performance than our method.

G.3. Ablation of the Graph Selection Heuristic for $\mathcal{G}_T$

As mentioned in Appendix D, we use a greedy heuristic to select a graph from class $\mathcal{G}_T$. To understand the performance of this heuristic, we replace it with an exhaustive search (which will take much higher time complexity) and test it on Aloha. As shown in Figure 13-Right, the peak performance of this implementation is similar to the one with the greedy heuristic, but the precise graph selection brings better stability.

H. Connection to a Recent Work on Hyper-Graphs

We note that a recent work on the expressiveness of hyper-graphs (Yoon et al., 2020) is relevant to our work. The connections (i.e., the differences and relevance) between Yoon et al. (2020) and our paper are summarized as follows:

- **Different metric of expressiveness.** Yoon et al. (2020) investigates the functionality of graph expressiveness in terms

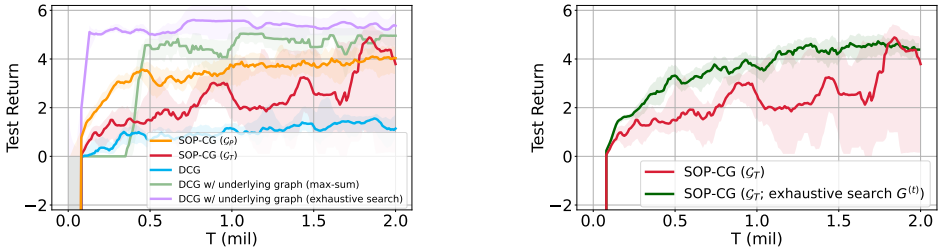

Figure 13. Additional results on Aloha. **Left**: Performance of DCG given a fixed static graph that reflects the underlying problem structure. **Right**: Performance of SOP-CG ($\mathcal{G}_T$) that selects $G^{(t)}$ by exhaustive search.

of the order of hyper-edges. By contrast, our paper focuses on the expressiveness of different graph topologies.

- **Similar motivation.** Both Yoon et al. and our paper aim to study the trade-off between the graph expressiveness and the complexity of solving downstream tasks. In our paper, the downstream task specifically refers to the DCOPs induced by graph-based value factorization.
- **Different workflow and contributions.** Yoon et al. (2020) establish a systematic experimental study to provide an empirical guideline on how to trade-off between the graph expressiveness and downstream costs. In comparison, our work proposes an algorithm to break this trade-off for improving MARL performance. Our algorithm is able to improve the function expressiveness without largely increasing the time complexity of downstream DCOPs.

Despite the expressiveness metric being different, several conclusions reached by Yoon et al. (2020) can be observed in our experiments. More specifically, the key findings of Yoon et al. (2020) (namely, three points, i.e, *Diminishing returns*, *Troubleshooter*, and *Irreducibility*) correspond to the following observations in our paper:

- **Diminishing returns.** When the task coordination structure is simple, a concise graph topology (e.g., pairwise matching) can achieve comparable performance with topologies using larger edge sets. As presented in Fig. 4, SOP-CG ($\mathcal{G}_P$) performs better than SOP-CG ($\mathcal{G}_T$) and DCG in Pursuit, since pairwise matching is sufficient to express the underlying coordination relations in this task. (Recall that $\mathcal{G}_P$ uses pairwise matching and $\mathcal{G}_T$ uses tree-based topology.)
- **Troubleshooter and Irreducibility.** When the task requires a complicated coordination structure, the graph topology needs to have higher function expressiveness. This finding can also be seen in Fig. 5. SOP-CG ($\mathcal{G}_T$) significantly outperforms SOP-CG ($\mathcal{G}_P$) in Tag where pairwise matching cannot express the underlying coordination relations.