Journal of Automation and Intelligence 5 (2026) 46–56

Contents lists available at ScienceDirect

Journal of Automation and Intelligence

journal homepage: www.keaipublishing.com/en/journals/journal-of-automation-and-intelligence/

Research article

GRA: Graph-based reward aggregation for cooperative multi-agent reinforcement learning

Jingcheng Tang [a], Peng Zhou [a], He Bai [b], Gangshan Jing [a,*]

[a] School of Automation, Chongqing University, Chongqing, 400044, China
[b] School of Mechanical and Aerospace Engineering, Oklahoma State University, Stillwater, OK, 74078, USA

ARTICLE INFO

Keywords:
Networked system
Multi-agent reinforcement learning
Graph-based RL

ABSTRACT

Multi-agent reinforcement learning (MARL) has proven its effectiveness in cooperative multi-agent systems (MASs) but still faces issues on the curse of dimensionality and learning efficiency. The main difficulty is caused by the strong inter-agent coupling nature embedded in an MARL problem, which is yet to be fully exploited in existing algorithms. In this work, we recognize a learning graph characterizing the dependence between individual rewards and individual policies. Then we propose a graph-based reward aggregation (GRA) method, which utilizes the inherent coupling relationship among agents to eliminate redundant information. Specifically, GRA passes information among cooperating agents through graph attention networks to obtain aggregated rewards that contribute to the fitting of the value function, making each agent learn a decentralized executable cooperation policy. In addition, we propose a variant of GRA, named GRA-decen, which achieves decentralized training and decentralized execution (DTDE) when each agent only has access to information of partial agents in the learning process. We conduct experiments in different environments and demonstrate the practicality and scalability of our algorithms.

1. Introduction

Multi-agent systems (MASs) in practice usually exhibit structural inter-agent couplings that can be represented as graphs [1], such as cellular networks [2], telecommunication networks [3,4], supply chain, and distribution networks [5–7]. These structural properties are critical because they determine how agents interact and coordinate with each other. Explicitly modeling such couplings allows algorithms to exploit the underlying dependencies, thereby improving learning efficiency and scalability in large-scale MASs.

In general multi-agent reinforcement learning (MARL) problems, each agent has to learn its optimal policy in an environment including other learning agents, which is non-stationary due to the couplings between the agents. To overcome this challenge, the centralized training and decentralized execution (CTDE) [8] paradigm has been proposed. Unfortunately, existing references, e.g., [8,9], usually construct the centralized state value function via global information, which always causes the learning algorithm to exhibit extremely poor performance when the number of the agents is large [10].

Extensive efforts have been made to improve the learning efficiency of cooperative MARL algorithms. For instance, methods such as communication [11], reward shaping [12], and mean field theory [13,14] have shown their unique advantages in facilitating multi-agent cooperation. Nevertheless, less attention has been paid to structure properties of MASs. Specifically, in a large-scale MAS, the inter-agent coupling relationship is usually not all to all, but rather follows a structure that can be described by a graph. Therefore, the aforementioned works still have the potential to be improved, since the structure property is not exploited. Independent learning [15] was shown to be highly efficient when different agents are weakly coupled, but becomes invalid when strong couplings exist due to the neglect of the information from other agents.

Indeed, some references have exploited certain coupling relationships among agents to design RL algorithms. For instance, the scalable actor critic proposed in Ref. [16] leverages the couplings between different agents' states to truncate the global value function. Additionally, in Refs. [17,18], different agents are coupled in the way that the reward of each agent involves its neighbors' states and actions. Although they all exploit the inherent coupling relationships among agents, they have not taken all types of couplings into full consideration, which may lead to inefficiency when scaling to larger systems. To address this gap, we aim to design an approach that can leverage the structural relationships

Peer review under responsibility of Chongqing University.
* Corresponding author.
E-mail addresses: 202213131165@stu.cqu.edu.cn (J. Tang), zhoupeng@stu.cqu.edu.cn (P. Zhou), he.bai@okstate.edu (H. Bai), jinggangshan@cqu.edu.cn (G. Jing).

https://doi.org/10.1016/j.jai.2025.10.006
Received 4 July 2025; Received in revised form 12 October 2025; Accepted 24 October 2025
Available online 30 October 2025

among agents in observations, transitions, and rewards, so that learning can be guided by the most relevant information.

In this paper, we consider a general case where different agents have mutual couplings in local observations, state transition functions, and reward functions. We propose an algorithm named graph-based reward aggregation (GRA) to effectively utilize these inherent relationships among agents, thereby enabling more efficient collaborative learning. The underlying idea is identifying cooperative relationships among the agents and excluding the influence of irrelevant agents during policy iteration. Specifically, we recognize a learning graph describing the relationship between individual policies and individual rewards. In the GRA algorithm, we use a reward graph neural network (GNN) and an observation GNN to form the "critic" for each agent, which aggregates information from neighboring agents in the learning graph. In addition, we propose a variant of GRA, named GRA-decen, which achieves decentralized training by message passing only within the one-hop neighboring agents. Our contributions are as follows.

(i). We recognize the learning graph embedded in general MASs, based on which we construct a local value function for each agent to guide the policy learning. Graph attention networks are utilized for each agent to aggregate neighbors' rewards, which enhances the approximation of the local value function. In Section 4.1.3, we provide ablation experiments to illustrate the benefit of using the aggregated reward in critic.

(ii). In contrast to many GNN-based RL algorithms, e.g., [19,20], the actor network learned by GRA for each agent solely requires the acquisition of its observations during the execution phase without additional communication. Moreover, GRA allows different agents to be heterogeneous (see our experiments in Section 4).

(iii). GRA-decen is designed by reducing the number of the observation GNN layers to one, and thus follows the decentralized training and decentralized execution (DTDE) framework. Benefiting from the graph-based learning architecture, the experiments show that despite using limited information, GRA-decen outperforms CTDE algorithms MADDPG [8] and MAPPO [9].

2. Background

2.1. Preliminaries

We consider a cooperative multi-agent task where the agents operate in a partially observable environment. The task can be modeled as a decentralized partially observable Markov decision process (DecPOMDP) [21], which is defined as a tuple: $\langle \mathcal{V}, S, \mathcal{O}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where $\mathcal{V} = \{1 \dots N\}$ is a set of N agents; $\boldsymbol{s}^t = \{s_1^t \dots s_N^t\} \in S$ is the joint state for all the agents at time step t; $o_i^t \sim \mathcal{O}(\boldsymbol{s}^t, i)$ is the local observation for agent i, and $\boldsymbol{o}^t = \{o_1^t \dots o_N^t\}$ is the joint observation; each agent i employs a stochastic policy $\pi_i \in \Pi$ to select its action $a_i^t \sim \pi_i(\cdot|o_i^t) \in \mathcal{A}_i$, where Π is the policy space; we use the policy network $\pi(\cdot|o_i^t; \theta_i)$ parameterized by θ_i to approximate the policy function $\pi_i(\cdot|o_i^t)$, and $\boldsymbol{\pi} = \{\pi_1 \dots \pi_N\}$; $\boldsymbol{a}^t = \{a_1^t \dots a_N^t\} \in \mathcal{A} = \cup_{i \in \mathcal{V}} \mathcal{A}_i$ is the joint action; $P(\boldsymbol{s}^{t+1}|\boldsymbol{s}^t, \boldsymbol{a}^t)$ is the transition probability from $\boldsymbol{s}^t$ to $\boldsymbol{s}^{t+1}$ given the joint action $\boldsymbol{a}^t$, $R(\boldsymbol{s}^t, \boldsymbol{a}^t) = \sum_{i=1}^N r_i^t$ is the joint reward function, $r_i^t = r_i(\boldsymbol{s}^t, \boldsymbol{a}^t)$ is the reward of agent i at time step t, and $r_i(\boldsymbol{s}^t, \boldsymbol{a}^t)$ is the reward function of agent i; $\gamma \in [0, 1)$ is the discount factor. In cooperative MARL, each agent learns a policy π_i to collaboratively maximize the total expected discounted return, denoted as $J(\boldsymbol{\pi}) = \mathbb{E}_{\boldsymbol{\pi}}[\sum_{t=0}^{\infty} \sum_{i=1}^N \gamma^t r_i^t]$.

2.2. Related work

Multi-Agent Actor Critic. In the actor-critic framework, CTDE algorithms have been widely adopted for MARL. For instance, in MADDPG [8], each agent learns its own centralized critic using joint observation and action, but the learned individual policy induces an action based only on its local observations. COMA [22] shares a critic across all agents and uses a counterfactual baseline to estimate the contribution of each agent's behavior to the team's reward. MAPPO [9] incorporates clipped surrogate objective to stabilize the learning process and demonstrates the effectiveness of PPO [23] in a wide variety of cooperative multi-agent settings. The above methods all require collecting the vectors of observations and actions from all agents as the input to the centralized critic, resulting in poor scalability due to the high-dimensional input. To resolve this issue, we utilize the coupling relationship among agents to design a local value function for each agent, which filters out useless information, therefore reduces the dimension of the critic input and improves the training efficiency.

Value Function Decomposition (VFD). Existing VFD algorithms have also employed local value functions to make use of local information. Ref. [24] proposed the VDN algorithm to solve the credit assignment problem by writing the joint value function as a sum of individual local action value functions. Unlike VDN, QMIX [25] uses a mixing network to aggregate the outputs of the local action value functions to achieve credit assignment. The global action value function obtains the joint value by aggregating the outputs of the local action value functions. Ref. [26] proposed QTRAN, which represents the joint action value function as the sum of the action value function of each agent plus an error term, allowing the algorithm to be used in a wider range of scenarios. However, when the network is of large scale, the VFD algorithms become invalid due to the difficulty of credit assignment. Unlike these value-based methods, our local value functions are designed based on the different types of inter-agent couplings, which may be clearly defined in specific application scenarios even if a large-scale problem is encountered (see Section 3.4).

Graph-Based MARL. In recent years, graph neural network (GNN) has gained attention in the field of MASs due to its ability to effectively tackle the graph-structured applications. For instance, DGN [19] uses the graph attention network (GAT) [27] to aggregate the information of the three nearest neighboring agents for each agent and uses the aggregated information as the input to the action value function, but they did not indicate whether there is a significant cooperation among these agents. GraphComm [28] categorizes relationships among agents into explicit and implicit categories. The exchange of information about both static and dynamic relationships among agents is facilitated through GAT. InforMARL [20] treats all entities in the environment, including target points, obstacles, etc. as nodes in the graph, and uses the GNN to aggregate information about the local neighborhoods of the agents and input it into the actor and critic. However, these graph-based methods always require communications among neighbors to pass information during decentralized execution.

Ref. [29] proposed a scalable MARL framework that fully utilizes the coupling relationship among agents. However, their algorithm can only be implemented within the REINFORCE framework [30], necessitating the collection of data until the end of an episode before the policy updates. In contrast, we employ the actor-critic framework, which leads to broader applicability.

3. Methodology

3.1. The learning graph in MARL

Given a MAS with a cooperative task described by the sum of all individual rewards, the policy of each agent may only influence the rewards of a subset of other agents. Denote by π_i the policy of agent i, and let $\boldsymbol{\pi} = \{\pi_1 \dots \pi_N\}$. The cooperative task goal is defined as:

$$J(\boldsymbol{\pi}) = \mathbb{E}_{s \sim D} V^{\boldsymbol{\pi}}(\boldsymbol{s}), \tag{1}$$

where D is the distribution that the initial state follows, and

$$V^{\boldsymbol{\pi}}(\boldsymbol{s}) = \mathbb{E}\left[\sum_{j \in \mathcal{V}} \sum_{t=0}^{\infty} \gamma^t r_j\left(\boldsymbol{s}^t, \boldsymbol{a}^t\right) | \boldsymbol{s}^0 = \boldsymbol{s}\right] \tag{2}$$

is the global value function.

Suppose that agent j's reward is independent of agent i's policy. It follows that

$$r_j\left(\boldsymbol{s},\{a_i+\varepsilon,a_{i-}\}\right)=r_j\left(\boldsymbol{s},\boldsymbol{a}\right),\forall \boldsymbol{s}\in\mathcal{S},\forall \boldsymbol{a}\in\mathcal{A}, \tag{3}$$

where $a_{i-}=\{a_j,j\neq i,j\in\mathcal{V}\}$, and ε is an arbitrary nonzero action perturbation with an appropriate dimension. Eq. (3) implies that there is no cooperative relationship between agents i and j.

We employ the learning graph $\mathcal{G}_{\mathrm{L}}=(\mathcal{V},\mathcal{E}_{\mathrm{L}})$ to reflect the inter-agent coupling relationship, which will be exploited in our RL algorithm. More specifically, let $(j,i)\in\mathcal{E}_{\mathrm{L}}$ if and only if i and j do not satisfy (3). In the following sections, we assume the existence of a learning graph $\mathcal{G}_{\mathrm{L}}$. The identification of $\mathcal{G}_{\mathrm{L}}$ for networked systems is discussed in Section 3.4.

Now we construct a local value function for agent i, $\hat{V}_i^{\boldsymbol{\pi}}(\boldsymbol{s})$, by focusing on the agents whose rewards are influenced by the policy of agent i, i.e.,

$$\hat{V}_i^{\boldsymbol{\pi}}(\boldsymbol{s})=\sum_{j\in\mathcal{N}_i^{\mathrm{L}}}\sum_{t=0}^{\infty}\mathbb{E}\left[\gamma^t r_j\left(\boldsymbol{s}^t,\boldsymbol{a}^t\right)|\boldsymbol{s}^0=\boldsymbol{s}\right], \tag{4}$$

where $\mathcal{N}_i^{\mathrm{L}}=\{j\in\mathcal{V}:(j,i)\in\mathcal{E}_{\mathrm{L}}\}\cup\{i\}$ is the set of neighbors (including itself) of agent i in graph $\mathcal{G}_{\mathrm{L}}$. By virtue of (3), the following holds:

$$\begin{aligned}
&V^{\{\pi_i+\zeta,\pi_{i-}\}}(\boldsymbol{s})-V^{\boldsymbol{\pi}}(\boldsymbol{s})\\
&=\sum_{i=1}^{N}\sum_{t=0}^{\infty}\gamma^t\left[r_i^t\left(\boldsymbol{s},\{a_i+\varepsilon,a_{i-}\}\right)-r_i^t(\boldsymbol{s},\boldsymbol{a})\right]\\
&=\sum_{j\in\mathcal{N}_i^L}\sum_{t=0}^{\infty}\gamma^t\left[r_j^t\left(\boldsymbol{s},\{a_i+\varepsilon,a_{i-}\}\right)-r_j^t(\boldsymbol{s},\boldsymbol{a})\right]\\
&=\hat{V}_i^{\{\pi_i+\zeta,\pi_{i-}\}}(\boldsymbol{s})-\hat{V}_i^{\boldsymbol{\pi}}(\boldsymbol{s}),
\end{aligned} \tag{5}$$

where $\pi_{i-}=\{\pi_j,j\neq i,j\in\mathcal{V}\}$, ζ is an arbitrary nonzero policy perturbation with appropriate dimension, and the perturbed action $\{a_i+\varepsilon,a_{i-}\}$ is determined by the perturbed policy $\{\pi_i+\zeta,\pi_{i-}\}$. Eqs. (5) implies that the impact of a policy change of agent i in the global value function is captured exactly in the change of the local value function of agent i, which fortunately consists of only partial agents.

We then replace the global value function V by a local value function $\hat{V}_i$ for each agent i during the learning procedure. The main advantage of employing $\hat{V}_i$ instead of V is that it excludes redundant information for each agent, thereby improving the learning efficiency. Note that throughout this paper, the learning graph is considered a non-complete graph, so that $\hat{V}_i\neq V$.

3.2. Graph-based learning architecture

We adopt an actor-critic framework incorporated with the learning graph to achieve scalable and efficient MARL. The main idea is to replace the global value function $V(\boldsymbol{s})$ in the critic by the local value function $\hat{V}_i(\boldsymbol{s})$, which only involves information of partial agents.

Considering that the state information is usually unavailable under the DecPOMDP setting, we use the joint observation $\boldsymbol{o}$ to represent global information instead of the joint state $\boldsymbol{s}$, which is similar to MAPPO [9]. Different from MAPPO, we further exploit the aggregated reward in training the critic of each agent. As a result, the local value function of each agent i is defined as

$$\hat{V}_i^{\boldsymbol{\pi}}(\boldsymbol{o},\boldsymbol{r})=\mathrm{GNN}\left(\boldsymbol{o}_{\mathcal{N}_i^\kappa},\boldsymbol{r}_{\mathcal{N}_i^{\mathrm{L}}},\mathcal{G}_{\mathrm{L}}\right), \tag{6}$$

where $\mathcal{N}_i^\kappa$ is the set of κ-hop neighbors of agent i in the learning graph, $\boldsymbol{o}_{\mathcal{N}_i^\kappa}=\{o_j,j\in\mathcal{N}_i^\kappa\}$, $\boldsymbol{r}_{\mathcal{N}_i^{\mathrm{L}}}=\{r_j,j\in\mathcal{N}_i^{\mathrm{L}}\}$ and $\mathcal{G}_{\mathrm{L}}$ is the learning graph. The form of $\hat{V}_i^{\boldsymbol{\pi}}(\boldsymbol{o},\boldsymbol{r})$ indicates that each agent only takes the reward aggregation of partial agents (determined by the learning graph) into account, therefore removing redundant reward information and improving the learning efficiency. The reward aggregation enables the agents to perceive changes in the rewards of their neighbors and stabilize the learning process, as shown in our ablation experiments in Section 4.1.3.

Within the framework of the multi-agent actor-critic paradigm [8, 9], each agent has its own actor and critic. For each agent, an observation GNN and a reward GNN are employed in the critic to accomplish message aggregation and instruct the actor to learn a cooperative policy. The observation GNN is primarily employed for aggregating observations, expanding the receptive field through computations across multiple layers to acquire more concentrated information. On the other hand, the reward GNN aggregates rewards from neighboring agents, facilitating the extraction of the aggregated rewards and furnishing instructive cues for fitting the local value function. Fig. 1 illustrates the overall architecture of GRA, from which we have:

$$\hat{V}_i^{\boldsymbol{\pi}}(\boldsymbol{o},\boldsymbol{r})=\mathrm{FC}\left(\hat{\boldsymbol{r}}_i,\hat{\boldsymbol{o}}_i\right), \tag{7}$$

where $\hat{\boldsymbol{r}}_i$ is the *aggregated reward* of agent i:

$$\hat{\boldsymbol{r}}_i=\mathrm{RGNN}\left(\boldsymbol{r}_{\mathcal{N}_i^{\mathrm{L}}},\mathcal{G}_{\mathrm{L}}\right) \tag{8}$$

and $\hat{\boldsymbol{o}}_i$ is the highly concentrated *aggregated observation* of agent i:

$$\hat{\boldsymbol{o}}_i=\mathrm{OGNN}\left(\boldsymbol{o}_{\mathcal{N}_i^\kappa},\mathcal{G}_{\mathrm{L}}\right), \tag{9}$$

which extracts observations within the κ-hop neighbors of agent i. We next briefly explain how RGNN and OGNN are constructed.

3.2.1. Attention-based aggregation GNN design

Motivated by Ref. [27], we use a GAT for each agent to accomplish the reward aggregation, which allows the agent to selectively prioritize its neighbors based on the importance of their rewards. As explained in Section 3.1, the learning graph $\mathcal{G}_{\mathrm{L}}$ describes the inter-agent couplings exactly. The neighbor set $\mathcal{N}_i^{\mathrm{L}}$ collects all the agents influenced by agent i. Therefore, we design the adjacency matrix of the GNN based on the learning graph.

Note that the local value function exclusively incorporates the reward information from the one-hop neighbors of each agent, implying that each agent collaborates solely with its one-hop neighbors. Therefore, we restrict the number of network layers in the reward GNN to one. Moreover, the critic input of each agent is composed of the rewards of agents in $\mathcal{N}_i^{\mathrm{L}}$. Accordingly, the output of each reward GNN is defined as:

$$\boldsymbol{x}_i'=\boldsymbol{W}_1\cdot\boldsymbol{x}_i+\sum_{j\in\mathcal{N}_i^{\mathrm{L}}\setminus\{i\}}\alpha_{i,j}\boldsymbol{W}_2\cdot x_j \tag{10}$$

with the attention coefficient $\alpha_{i,j}$ designed by the multi-head dot product:

$$\alpha_{i,j}=\mathrm{softmax}\left(\frac{(\boldsymbol{W}_3\boldsymbol{x}_i)^{\top}(\boldsymbol{W}_4\boldsymbol{x}_j)}{\sqrt{d}}\right). \tag{11}$$

Here $\boldsymbol{x}_i$ consists of the node features of agent i, which are induced by the individual reward r_i, $\{\boldsymbol{W}_k,k=1\ldots4\}$ is the set of learnable weight matrices, and d is the output dimension of the layer. In our experiments, the message passing among agents in (10) is achieved by the unified message passing model (UniMP) [31].

We employ a similar GNN structure for the aggregation of agents' observations. Unlike the reward GNN, the observation GNN needs to perform multi-layer information aggregation to acquire the global information, where the output of each layer is calculated by (10). Therefore, the overall input of the critic for agent i is $\boldsymbol{o}_{\mathcal{N}_i^\kappa}$. Note that the aggregated observation $\hat{\boldsymbol{o}}_i$, i.e., the output of the observation GNN, may contain only approximate global information, as $\mathcal{N}_i^\kappa\subseteq\mathcal{V}$ may not encompass all agents.

Similar to DGN [19], we tend to increase the number of the observation GNN layers to get a larger receptive field. As κ increases, the set $\mathcal{N}_i^\kappa$ encompasses more agents, and $\boldsymbol{o}_{\mathcal{N}_i^\kappa}$ is more likely to contain global state information. However, this comes at the cost of an increased input dimension, which may lead to a decline in learning efficiency. When

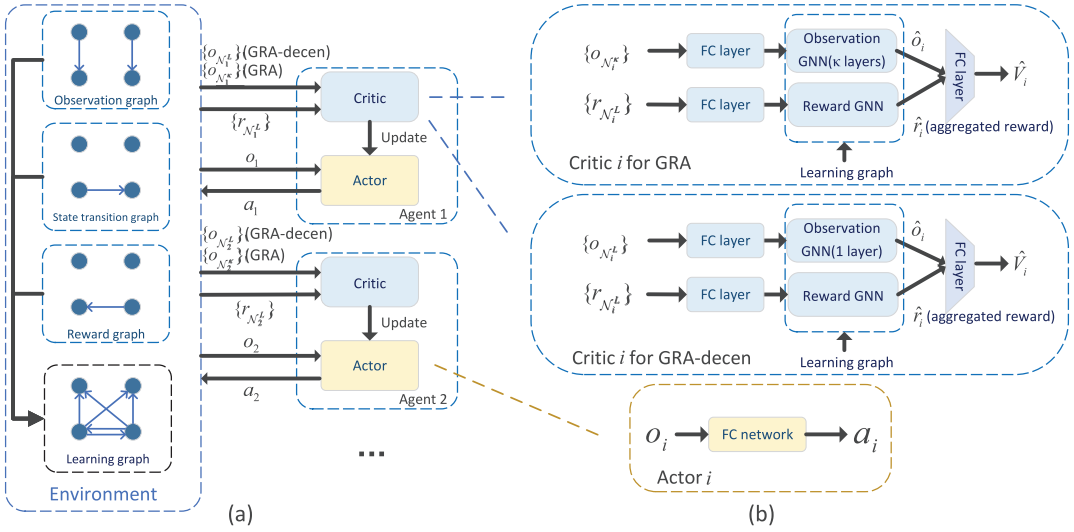


Fig. 1. In (a), information flow among the decentralized actors, the environment and critics in GRA and GRA-decen. In (b), architectures of the critic and actor for agent i, the FC layer is a fully connected layer and the FC network consists of multiple fully connected layers.

the multi-agent network is sparse, the algorithm with small κ may still perform well because the coupling between agents with a long distance is weak. Specifically, in our experiments with GRA, we employ three observation GNN layers.

3.2.2. The GRA algorithm

Algorithm 1 Graph-based reward aggregation multi-agent reinforcement learning

Initialize θ_i for policy π_i and ϕ_i for critic $\hat{V}_i$ of each agent i
Get the learning graph from the environment
while $step \leq step_{\max}$ **do**
  Set data buffer $D = \{\}$
  for $k = 1$ to $batch\_size$ **do**
    $\tau = [\,]$
    for $t = 1$ to T **do**
      for all agents $i = 1$ to n **do**
        $a_i^t = \pi(o_i^t; \theta_i)$
      end for
      Execute actions $\boldsymbol{a}^t$ and observe $\boldsymbol{r}^t, \boldsymbol{o}^{t+1}$
      for all agents $i = 1$ to n **do**
        $v_i^t = \hat{V}_i(\boldsymbol{o}^t, \boldsymbol{r}^t; \phi_i)$
      end for
      $\tau += [\boldsymbol{o}^t, \boldsymbol{a}^t, \boldsymbol{v}^t, \boldsymbol{r}^t, \boldsymbol{o}^{t+1}]$
    end for
    Compute the set of advantage estimate $\hat{A} = \{\hat{A}_1^1, ..., \hat{A}_1^t, ..., \hat{A}_n^1, ..., \hat{A}_n^t\}$ and the set of reward-to-go $\hat{R} = \{\hat{R}_1^1, ..., \hat{R}_1^t, ..., \hat{R}_n^1, ..., \hat{R}_n^t\}$ on τ
    $D = D \bigcup (\tau, \hat{A}, \hat{R})$
  end for
  Each agent i updates θ_i and ϕ_i using D according to (14) and (15)
end while

Now we present our GRA algorithm, whose pseudo-code is shown in Algorithm 1, where $\hat{R}_i^t$ is the reward-to-go of agent i at timestep t. In Algorithm 1, each agent $i \in \mathcal{V}$ is expected to maximize the following objective:

$$\hat{J}_i(\boldsymbol{\theta}) = \mathbb{E}_{s\sim D} \hat{V}_i^{\boldsymbol{\pi}(\boldsymbol{\theta})}(\boldsymbol{o}, \boldsymbol{r}). \tag{12}$$

Inspired by Ref. [32], we use a variant generalized advantage estimation (GAE) based on local value functions:

$$\hat{A}_i^t = \delta_i^t + (\gamma\lambda)\,\delta_i^{t+1} + \cdots + \cdots + (\gamma\lambda)^{T-t-1}\,\delta_i^{T-1}, \tag{13}$$

where $\delta_i^t = \sum_{j\in\mathcal{N}_i^{\mathrm{L}}} r_j^t + \gamma \hat{V}_i\left(\boldsymbol{o}^t, \boldsymbol{r}^t\right) - \hat{V}_i\left(\boldsymbol{o}^{t+1}, \boldsymbol{r}^{t+1}\right)$.

The main novelty of the proposed individual advantage for agent i in (13) is that it only involves the rewards of agents in $\mathcal{N}_i^{\mathrm{L}}$, which is determined by the learning graph.

In our implementation, we adopt the loss functions of PPO [23] to train $\pi_i(\theta_i)$ and $\hat{V}_i(\phi_i)$ for each agent i, where θ_i and ϕ_i are the trainable neural network parameters. The loss of agent i is defined as:

$$L(\theta_i) = \frac{1}{B}\sum_{k=1}^{B} r_{\theta,i}^{(k)} \hat{A}_i^{(k)}, \tag{14}$$

$$L(\phi_i) = \frac{1}{B}\sum_{k=1}^{B} \left(\hat{V}_{\phi_i}(\boldsymbol{o}^{(k)}, \boldsymbol{r}^{(k)}) - \hat{R}_i^{(k)}\right)^2, \tag{15}$$

where $r_{\theta,i}^{(k)} = \frac{\pi_{\theta_i}(a_i^{(k)}|o_i^{(k)})}{\pi_{\theta_{i\mathrm{old}}}(a_i^{(k)}|o_i^{(k)})}$, B is the batch size. The GRA algorithm shows superiority over other CTDE algorithm baselines in our experiments.

3.3. Decentralized learning architecture for GRA

When the number of agents is large, it may be impractical to learn a centralized critic conditioning on the global state and joint action [10]. Additionally, in MASs, some observations may not be readily shareable, which prevents centralized training [33]. Given these two reasons, we replace $\boldsymbol{o}_{\mathcal{N}_i^\kappa}$ with $\boldsymbol{o}_{\mathcal{N}_i^{\mathrm{L}}}$ as the input for the observation GNN.

By simplifying the inputs to the observation GNN and reducing the number of the observation GNN layers to one, we propose a more scalable version of GRA, called GRA-decen, whose $\hat{\boldsymbol{o}}_i$ can be denoted as:

$$\hat{\boldsymbol{o}}_i = \mathrm{OGNN}\left(\boldsymbol{o}_{\mathcal{N}_i^{\mathrm{L}}}, \mathcal{G}_{\mathrm{L}}\right). \tag{16}$$

In our experiments, we find that the GRA-decen algorithm still performs well using the DTDE approach. In addition, we assess the impact of varying the number of observation GNN layers on the performance, which is shown in Appendix C.

3.4. Recognizing the learning graph in MASs with networked couplings

In this subsection, we show how to find the learning graph embedded in a cooperative MAS where different agents have couplings in observation, state transition, and reward functions.

3.4.1. Graphs for direct inter-agent couplings

In Refs. [34,35], each agent is assumed to be able to observe state information of other agents and part of environmental information. Here we consider the same setting and adopt a directed graph $\mathcal{G}_{\mathrm{O}} = (\mathcal{V}, \mathcal{E}_{\mathrm{O}})$ (called the observation graph) to depict the inter-agent observation relationship, where $(i, j) \in \mathcal{E}_{\mathrm{O}}$ implies that agent j can observe the state of agent i. Similarly, we use the state transition graph $\mathcal{G}_{\mathrm{S}} = (\mathcal{V}, \mathcal{E}_{\mathrm{S}})$ and the reward graph $\mathcal{G}_{\mathrm{R}} = (\mathcal{V}, \mathcal{E}_{\mathrm{R}})$ to represent the inter-agent state transition coupling and reward coupling, respectively. More specifically, $(i, j) \in \mathcal{E}_{\mathrm{S}}$ means that the state transition of agent j involves agent i's state or action, and $(i, j) \in \mathcal{E}_{\mathrm{R}}$ means that the local reward of agent j is affected by the state of agent i. The state transition coupling and reward coupling between agents have also been considered in Refs. [16–18], respectively.

The above-mentioned three graphs represent different types of inter-agent couplings, which can usually be directly observed when the MAS has a clear network structure. Next, we will find the learning graph based on the three graphs. For the sake of understanding, we present a distributed resource allocation example.

Consider a network of 4 warehouses consuming resources while transferring resources among each other. The goal is to guarantee adequate supplies for each warehouse. Each warehouse is denoted by a vertex in the graph. The coupling relationships are represented in Fig. 2. At time step t, warehouse $i \in \mathcal{V}$ stores resources of the amount $s_i^t \in \mathbb{R}$, receives a local demand $d_i^t \in \mathbb{R}$, sends part of its resources to and receives resources from agents in the out-neighbor set $\mathcal{N}_i^{\mathrm{S}+} = \{j \in \mathcal{V} : (i, j) \in \mathcal{E}_{\mathrm{S}}\}$ and the in-neighbor set $\mathcal{N}_i^{\mathrm{S}} = \{j \in \mathcal{V} : (j, i) \in \mathcal{E}_{\mathrm{S}}\}$, respectively, in the state transition graph $\mathcal{G}_{\mathrm{S}}$. Besides its neighbors, warehouse i also receives external resources of the amount q_i^t. Let $z_i^t = q_i^t - d_i^t$. Then agent i has the following dynamics

$$s_i^{t+1} = s_i^t - \sum_{j \in \mathcal{N}_i^{\mathrm{S}+}} a_{ij} s_i^t + \sum_{j \in \mathcal{N}_i^{\mathrm{S}}} a_{ji} s_j^t + z_i^t, \tag{17}$$

$$a_{ij} = \begin{cases} a_{ij} & ,\text{if } s_i^t \geq 0 \\ 0 & ,\text{if } s_i^t < 0 \end{cases}, \tag{18}$$

where $a_{ij} \in [0, 1]$ denotes the fraction of resources agent i sends to its neighbor j at time t, and $z_i^t = A_i \sin(t)$ with A_i being a constant represents dynamic disturbances.

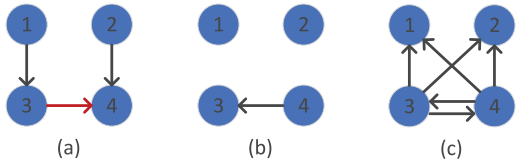


Fig. 2. (a) The state transition graph $\mathcal{G}_{\mathrm{S}} = (\mathcal{V}, \mathcal{E}_{\mathrm{S}})$ and the observation graph $\mathcal{G}_{\mathrm{O}} = (\mathcal{V}, \mathcal{E}_{\mathrm{O}})$. The red and black lines correspond to edges in $\mathcal{E}_{\mathrm{S}}$ and $\mathcal{E}_{\mathrm{O}}$, respectively. (b) The reward graph $\mathcal{G}_{\mathrm{R}} = (\mathcal{V}, \mathcal{E}_{\mathrm{R}})$. (c) The learning graph $\mathcal{G}_{\mathrm{L}} = (\mathcal{V}, \mathcal{E}_{\mathrm{L}})$.

To find the learning graph, we analyze implicit and explicit cooperation among agents. In the following, we analyze how the three graphs produce influential relationships among the agents.

Implicit cooperation of agents in the observation graph and state transition graph: At time t, s_1^t is directly observed by agent 3, then the change of s_1 influences a_3 through $\pi(o_3^t; \theta_3)$, and in turn affects r_3 and s_3^{t+1}. Note that s_3 will affect agent 4 due to the state transition coupling, and therefore passes on the influence of agent 1 to agent 4. Over time, s_1 affects all the agents it can reach in the graph $\mathcal{G}_{\mathrm{SO}} = (\mathcal{V}, \mathcal{E}_{\mathrm{S}} \cup \mathcal{E}_{\mathrm{O}})$. Thus, it is possible to determine from $\mathcal{G}_{\mathrm{SO}}$ which of the other agents each agent will cooperate with. Such a cooperative relationship exhibits temporal characteristics, therefore it is implicit and difficult to exploit in RL algorithms.

Explicit cooperation of agents in the reward graph: The coupling relationship in the reward graph directly represents the mutual influence between different agents, which is explicit.

The implicit and explicit cooperation relationship has also been studied in the causal influence [12] and value function decomposition [36], respectively. However, neither of them considers all types of couplings based on graphs.

3.4.2. Construction of the learning graph

We use $\mathcal{R}_i^{\mathrm{SO}} = \{j \in \mathcal{V} : i \xrightarrow{\mathcal{E}_{\mathrm{SO}}} j\} \cup \{i\}$ to represent the set of agents that have implicit cooperation with agent i, which includes the vertices that are reachable from vertex i and vertex i itself in graph $\mathcal{G}_{\mathrm{SO}}$. We then look for other agents that have explicit cooperation with the agents in the set $\mathcal{R}_i^{\mathrm{SO}}$, and finally get all the sources of information needed for agent i to learn to complete the cooperation task. The set of neighbors for agent i in the learning graph is given by

$$\mathcal{N}_i^{\mathrm{L}} = \left\{ j \in \mathcal{V} : \mathcal{N}_j^{\mathrm{R}} \cap \mathcal{R}_i^{\mathrm{SO}} \neq \varnothing \right\} = \cup_{k \in \mathcal{R}_i^{\mathrm{SO}}} \mathcal{N}_k^{\mathrm{R}+}, \tag{19}$$

where $\mathcal{N}_j^{\mathrm{R}} = \{h \in \mathcal{V} : (h, j) \in \mathcal{E}_{\mathrm{R}}\} \cup \{j\}$; $\mathcal{N}_k^{\mathrm{R}+} = \{j \in \mathcal{V} : (k, j) \in \mathcal{E}_{\mathrm{R}}\} \cup \{k\}$, and $\mathcal{N}_k^{\mathrm{R}+}$ is the set of agents that have explicit cooperation with agent k.

The definition of $\mathcal{N}_i^{\mathrm{L}}$ in (19) implies the following result.

Theorem 1. *By setting the learning graph based on* (19)*, Eq.* (5) *hold. (Proof in Appendix* A.*)*

4. Experiments

In this section, we will provide empirical results in the environment of resource allocation described in Section 3.4, multi-echelon inventory optimization [37] and the Multi-agent Particle-world Environment (MPE) [8], respectively, which support our analysis results and verify the advantages of GRA and GRA-decen. Our code is available at https://github.com/cogithu/GRA.

4.1. Resource allocation

In this example, we compare GRA and GRA-decen with MAPPO [9] and MADDPG [8]. Note that each warehouse may send resources to multiple other warehouses, which causes a high-dimensional action space when discretizing actions. Therefore, we do not use value-based methods as baselines. To show the benefits of using the reward aggregation, we also evaluate GRA without reward aggregation for the ablation study.

4.1.1. A network of 9 warehouses

Based on the resource allocation scenario described in Section 3.4, we consider a network of 9 warehouses with the observation graph, state transition graph, and reward graph shown in Fig. 3. According to (19), the learning graph is obtained in Fig. 4, which is utilized during the implementation of GRA. Note that the warehouses have heterogeneous observation and action spaces since different warehouses may have different neighbors in the observation graph and the state transition graph.

We train the algorithms for 12 000 episodes, and compare GRA and GRA-decen with other algorithms using the CTDE framework. Fig. 5(a) shows the evolution of their global rewards. For all the algorithms, the shadowed area is enclosed by the min and max value of training runs, and the solid line in the middle is the mean value. To strengthen the cooperative relationship among warehouses, we set the dynamic

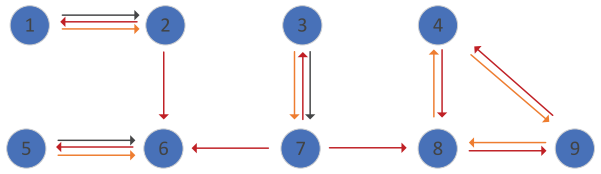


Fig. 3. The state transition graph $\mathcal{G}_S = (\mathcal{V}, \mathcal{E}_S)$, the observation graph $\mathcal{G}_O = (\mathcal{V}, \mathcal{E}_O)$ and the reward graph $\mathcal{G}_R = (\mathcal{V}, \mathcal{E}_R)$. The red, black and yellow lines correspond to edges in $\mathcal{E}_S$, $\mathcal{E}_O$ and $\mathcal{E}_R$, respectively.

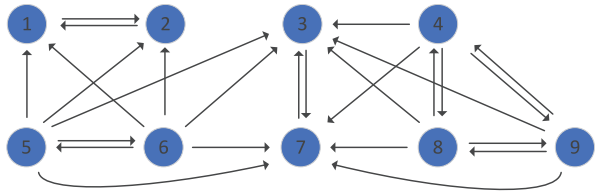


Fig. 4. The learning graph $\mathcal{G}_L = (\mathcal{V}, \mathcal{E}_L)$ for 9 warehouses.

disturbance $A_i = 1$ for $i \in \{2, 6, 7, 8\}$, and $A_i = -1$ for $i \in \{1, 3, 4, 5, 9\}$. Those warehouses with $A_i = -1$ are more likely to have a shortage of resources, which requires other warehouses with more abundant resources to allocate resources to them.

As shown in Fig. 5(a), GRA and GRA-decen outperform MAPPO and MADDPG significantly. This is because the local value function excludes reward information unrelated to itself, which is easier to fit compared with the global value function. The graph attention mechanism filters the information generated within the neighboring set so that the global reward obtained by the policy trained with the local value function increases faster than other algorithms in the early training stage. In this experiment, the coupling relationship among the 9 warehouses is complex, which increases the difficulty of learning the global value function, leading to unsatisfactory results for MAPPO and MADDPG.

4.1.2. A network of 20 warehouses

In this example, we allocate resources to 20 warehouses. The inter-agent coupling relationship diagram can be found in Appendix B. The state transition graph and the observation graph have the same edge set: $\mathcal{E}_S = \mathcal{E}_O = \{(i, j) \in \mathcal{V}^2 : |i - j| = 1, i = 2k, k \in \mathbb{N}\} \cup \{(N, 1)\}$. The edge set of the reward graph is denoted by $\mathcal{E}_R = \{(i, j) \in \mathcal{V}^2 : |i - j| = 1, i = 2k - 1, k \in \mathbb{N}\} \cup \{(1, N)\}$.

Similarly, to strengthen the partnership among warehouses, we set $A_i = 0.5$ if the action of warehouse i affects the state of other warehouses, and $A_i = -0.5$ otherwise. We train the algorithms for 5 000 episodes. Fig. 5(b) shows their learning curves in terms of the mean reward. Although the number of warehouses in the experiments has increased, the coupling relationship among the warehouses is relatively straightforward. GRA and GRA-decen effectively learn a cooperative policy through the utilization of local value functions. In contrast, the networks of MAPPO and MADDPG require the input of global information and face considerable challenges in learning from such extensive data with intricate coupling relationships.

4.1.3. Ablation: effect of the reward aggregation

To validate the importance of reward aggregation, we conduct experiments with the network of 20 warehouses without the reward GNN. From Fig. 5(c), we see that using the reward GNN to achieve reward aggregation among the neighbors significantly improves the performance of the algorithm. Although the reward information is essentially determined by the actions and states of the agents, we find that directly aggregating the rewards of the neighbors of an agent helps fit the local value function. The aggregated reward $\hat{r}_i$ represents what agent i can gain among explicit and implicit cooperative relationships using the current policy.

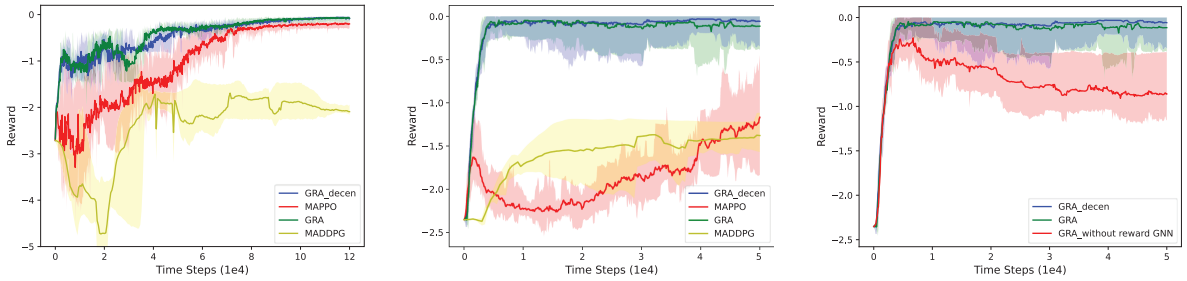
For GRA without reward GNN, the suboptimal performance is possibly caused by the uncertain random noise in the environment, which leads to a high variance in GAE (13) and the agents are unable to handle such uncertainty during the learning. However, the inclusion of the reward GNN can provide this critical reward information, thereby aiding in stabilizing the learning process.

4.2. Multi-echelon inventory optimization

To evaluate the performance of the GRA algorithm in more complex and realistic scenarios, we compare various algorithms in the context of multi-echelon inventory optimization. Unlike the resource allocation environment discussed in Section 4.1, in multi-echelon inventory optimization, the action of each warehouse is to determine the quantity of goods in the order submitted to its upstream suppliers. All warehouses earn profit by selling goods acquired from upstream suppliers to downstream customers. The objective of the entire system is to dynamically coordinate the inventory levels across all warehouses in order to maximize both the total net profit and the fulfillment of downstream orders. A detailed description of the experimental setup can be found in Appendix B.3.

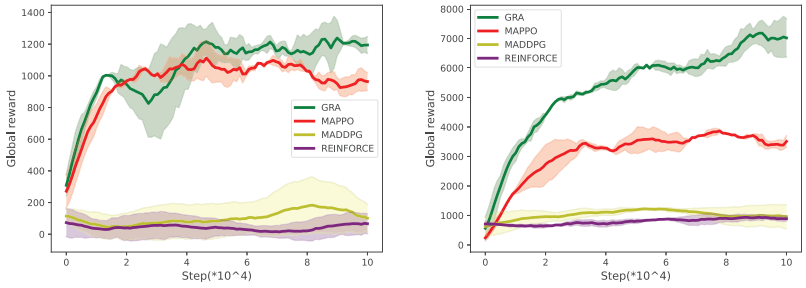
We evaluate GRA, MAPPO, MADDPG, and the REINFORCE algorithm with global rewards in both serial and networked multi-echelon inventory control systems. The variations in global rewards obtained by different algorithms are illustrated in Fig. 6.

In multi-echelon inventory optimization, we obtain conclusions similar to those in Section 4.1. GRA replaces the global value function with the local value function, enabling each warehouse to focus only on collaborating with other warehouses whose long-term returns can be influenced by its own policy. This reduces interference from redundant information and improves training efficiency. As a result, GRA achieves higher global rewards than MAPPO, and its advantage becomes more



(a) 9 warehouses. (b) 20 warehouses. (c) Ablation studies.

Fig. 5. Experimental results in the resource allocation problem.



(a) The serial system. (b) The networked system.

Fig. 6. Experimental results in multi-echelon inventory optimization.

pronounced as the system complexity and scale increase. In contrast, the REINFORCE algorithm struggles to achieve higher rewards due to its inability to address the credit assignment problem.

We use the sales-side demand fulfillment rate, defined as the ratio of the total volume sold by all warehouses connected to a sales terminal to that terminal's total demand, as the service-level metric for the inventory control system. Table 1 and 2 compare this metric among various algorithms in serial and networked inventory control systems, respectively. The results show that GRA consistently outperforms the other methods in both settings, achieving the highest demand fulfillment rates and demonstrating superior overall performance.

Table 1
Demand fulfillment rate in the serial inventory control system.

Algorithms	Mean	Variance
GRA	**79.17**	0.19
MAPPO	63.11	0.21
MADDPG	37.53	0.31
REINFORCE	31.70	0.02

Table 2
Demand fulfillment rate in the networked inventory control system.

Algorithms	Mean	Variance
GRA	**83.15**	0.05
MAPPO	58.61	0.34
MADDPG	43.17	0.01
REINFORCE	44.50	0.15

4.3. Multi-Agent Particle Environment (MPE)

We adapt the MPE environment [8,38] to assess our proposed method. Specifically, the modified environment includes a single landmark along with 20 agents. The objective is to make these agents learn to cooperate, resulting in a distribution of agents in all directions around the target landmark, effectively encircling the target landmark. The coupling relationships among the agents and the experimental setting are presented in Appendix B.4.

From the experimental results in Fig. 7, we find that GRA and GRA-decen perform the best in MPE, while MAPPO exhibits inferior performance. QMIX and MADDPG perform significantly worse than the other algorithms. Our results indicate that GRA and GRA-decen have certain advantages in promoting the learning of cooperative behavior. On the other hand, QMIX and MADDPG struggle to effectively leverage the data with structured constraints in MPE, making it challenging to learn cooperative policies.

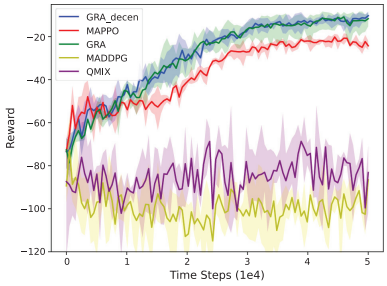


Fig. 7. Learning curve for MPE with 20 agents.

5. Conclusions

We introduced the learning graph derived from the coupling relationships among agents in MASs to design local value functions for MARL. With the help of the learning graph, the proposed GRA algorithm guides the agents to efficiently learn cooperative behaviors based on reward aggregation. Furthermore, we proposed GRA-decen to simplify the inputs of the GRA and enable DTDE. The empirical results show that in comparison to MAPPO and MADDPG, GRA and GRA-decen exhibit significantly better performance. We conclude from our results that appropriately incorporating reward information into the critic during training stabilizes the training process and facilitates the learning of a cooperative policy. For future work, in environments where such graphs are not explicitly predefined, the learning graph can be constructed by analyzing the interaction data to capture the coupling relationships among agents, for example through causal inference or attention mechanisms, which constitutes our subsequent research direction.

CRediT authorship contribution statement

Jingcheng Tang: Writing – original draft, Methodology. **Peng Zhou:** Writing – review & editing, Validation, Methodology. **He Bai:** Writing – review & editing, Supervision. **Gangshan Jing:** Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

Gangshan Jing is an associate editor for *Journal of Automation and Intelligence* and was not involved in the editorial review or the decision to publish this article. All authors declare that there are no competing interests.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (grants 62203073 and 62573068) and the Natural Science Foundation of Chongqing, China (grant CSTB2022NSCQMSX0577).

Data availability

Data will be made available on request.

Appendix A. Proof of Theorem 1

Before proving Theorem 1, we present some definitions. We use $\mathcal{G}_X = (\mathcal{V}, \mathcal{E}_X)$ to denote an unweighted directed graph; define $\mathcal{N}_i^X = \{j \in \mathcal{V} : (j,i) \in \mathcal{E}_X\} \cup \{i\}$, where $X \in \{\mathrm{O}, \mathrm{S}, \mathrm{R}, \mathrm{L}, \mathrm{SO}\}$; the global value function can be written as:

$$\begin{aligned} V^{\boldsymbol{\pi}}(\boldsymbol{s}) &= \mathbb{E}\left[\sum_{j\in\mathcal{V}}\sum_{t=0}^{\infty}\gamma^t r_j\left(s^t_{\mathcal{N}_j^R}, a^t_{\mathcal{N}_j^R}\right)|\boldsymbol{s}^0 = \boldsymbol{s}\right] \\ &= \sum_{t=0}^{\infty}\gamma^t\sum_{j\in\mathcal{V}}\mathbb{E}\Big[r_j\left(s^t_{\mathcal{N}_j^R}, a^t_{\mathcal{N}_j^R}\right)|\boldsymbol{s}^t \sim P(\cdot|\boldsymbol{s}^0 = \boldsymbol{s}, \\ &\quad \boldsymbol{\pi}), \boldsymbol{a} \sim \boldsymbol{\pi}\left(\cdot|\boldsymbol{s}^t\right)\Big], \end{aligned} \tag{20}$$

where $\boldsymbol{\pi}$ is the set of all agents' policies, r_j^t is the reward of agent j at time step t, and $r_j(\boldsymbol{s}, \boldsymbol{a})$ is the reward function of agent j. Similarly, we have

$$\begin{aligned} \hat{V}_i^{\boldsymbol{\pi}}(\boldsymbol{s}) &= \sum_{t=0}^{\infty}\gamma^t\sum_{j\in\mathcal{N}_i^L}\mathbb{E}\Big[r_j\left(s^t_{\mathcal{N}_j^R}, a^t_{\mathcal{N}_j^R}\right)|\boldsymbol{s}^t \sim P(\cdot|\boldsymbol{s}^0 = \boldsymbol{s} \\ &\quad , \boldsymbol{\pi}), \boldsymbol{a} \sim \boldsymbol{\pi}\left(\cdot|\boldsymbol{s}^t\right)\Big], \end{aligned} \tag{21}$$

$$\begin{aligned} \tilde{V}_i^{\boldsymbol{\pi}}(\boldsymbol{s}) &\triangleq V^{\boldsymbol{\pi}}(\boldsymbol{s}) - \hat{V}_i^{\boldsymbol{\pi}}(\boldsymbol{s}) \\ &= \sum_{t=0}^{\infty}\gamma^t\sum_{j\notin\mathcal{N}_i^L}\mathbb{E}\Big[r_j\left(s^t_{\mathcal{N}_j^R}, a^t_{\mathcal{N}_j^R}\right)|\boldsymbol{s}^t \sim P(\cdot| \\ &\quad \boldsymbol{s}^0 = \boldsymbol{s}, \boldsymbol{\pi}), \boldsymbol{a} \sim \boldsymbol{\pi}\left(\cdot|\boldsymbol{s}^t\right)\Big]. \end{aligned} \tag{22}$$

Recall that $\mathcal{R}_i^{\mathrm{SO}} = \{j \in \mathcal{V} : i \xrightarrow{\mathcal{E}_{\mathrm{SO}}} j\} \cup \{i\}$ denotes the set of nodes that are reachable from node i in graph $\mathcal{G}_{\mathrm{SO}}$. We give the following lemma to assist the proof of Theorem 1.

Lemma 1. *For any $i \in \mathcal{V}$, it holds that $\bigcup_{j\in\mathcal{V}\setminus\mathcal{R}_i^{\mathrm{SO}}}\mathcal{N}_j^{\mathrm{SO}} = \mathcal{V}\setminus\mathcal{R}_i^{\mathrm{SO}}$.*

Proof. Assume that $\exists k \in \bigcup_{j\in\mathcal{V}\setminus\mathcal{R}_i^{\mathrm{SO}}}\mathcal{N}_j^{\mathrm{SO}}$ and $k \in \mathcal{R}_i^{\mathrm{SO}}$. Since $k \in \mathcal{N}_j^{\mathrm{SO}}$, we have $j \in \mathcal{R}_k^{\mathrm{SO}}$. It follows that $j \in \mathcal{R}_i^{\mathrm{SO}}$. But our assumption is that $j \in \mathcal{V}\setminus\mathcal{R}_i^{\mathrm{SO}}$, so we arrive at a contradiction. □

Then we prove Theorem 1.

Proof. To show that Eqs. (5) holds, we only need to prove $\bar{V}_i^{\{\pi_i+\zeta,\pi_{i-}\}}(\boldsymbol{s}) - \tilde{V}_i^{\boldsymbol{\pi}}(\boldsymbol{s}) = 0$.

Since $j \notin \mathcal{N}_i^{\mathrm{L}}$ implies that $\mathcal{N}_j^{\mathrm{R}} \cap \mathcal{R}_i^{\mathrm{SO}} = \varnothing$, we have $i \notin \mathcal{N}_j^{\mathrm{R}}$. It follows that a_i and $s^t_{\mathcal{R}_i^{\mathrm{SO}}}$ do not affect $r_j(s^t_{\mathcal{N}_j^{\mathrm{R}}}, a^t_{\mathcal{N}_j^{\mathrm{R}}})$.

According to the definition of the transition probability, there exists a function f such that

$$s_j^t \sim f\left(s^{t-1}_{\mathcal{N}_j^{\mathrm{S}}}, a^{t-1}_{\mathcal{N}_j^{\mathrm{S}}}\right). \tag{23}$$

By Lemma 1 and (23), we use $s_{\mathcal{V}\setminus\mathcal{R}_i^{\mathrm{SO}}}$ to represent the states of agents outside the set $\mathcal{R}_i^{\mathrm{SO}}$, and have another function h:

$$s^t_{\mathcal{V}\setminus\mathcal{R}_i^{\mathrm{SO}}} \sim h\left(s^{t-1}_{\mathcal{V}\setminus\mathcal{R}_i^{\mathrm{SO}}}, a^{t-1}_{\mathcal{V}\setminus\mathcal{R}_i^{\mathrm{SO}}}\right), \tag{24}$$

according to (24), we can get that for $t \geq 0$, $s^t_{\mathcal{V}\setminus\mathcal{R}_i^{\mathrm{SO}}}$ is not affected by $s_{\mathcal{R}_i^{\mathrm{SO}}}$ or $a_{\mathcal{R}_i^{\mathrm{SO}}}$, then we conclude that π_i never influences $s^t_{\mathcal{V}\setminus\mathcal{R}_i^{\mathrm{SO}}}$ for any $t \geq 0$. So we have:

$$\begin{aligned} &\bar{V}_i^{\{\pi_i+\zeta,\pi_{i-}\}}(\boldsymbol{s}) - \bar{V}_i^{\boldsymbol{\pi}}(\boldsymbol{s}) && (25) \\ &= \sum_{t=0}^{\infty}\gamma^t\sum_{j\notin\mathcal{N}_i^{\mathrm{L}}}\mathbb{E}\left[r_j\left(s^t_{\mathcal{N}_j^{\mathrm{R}}}, a^t_{\mathcal{N}_j^{\mathrm{R}}}\right)|\boldsymbol{s}^t \sim P(\cdot|\boldsymbol{s}^0 = \boldsymbol{s}, \right. \\ &\quad \left.\{\pi_i+\zeta, \pi_{i-}\}), a_i \sim \left[\pi_i+\zeta\right]\left(\cdot|\boldsymbol{s}^t\right), a_{i-} \sim \pi_{i-}\left(\cdot|\boldsymbol{s}^t\right)\right] \\ &\quad - \sum_{t=0}^{\infty}\gamma^t\sum_{j\notin\mathcal{N}_i^{\mathrm{L}}}\mathbb{E}\left[r_j\left(s^t_{\mathcal{N}_j^{\mathrm{R}}}, a^t_{\mathcal{N}_j^{\mathrm{R}}}\right)|\boldsymbol{s}^t \sim \right. \\ &\quad \left.P\left(\cdot|\boldsymbol{s}^0 = \boldsymbol{s}, \boldsymbol{\pi}\right), \boldsymbol{a} \sim \boldsymbol{\pi}\left(\cdot|\boldsymbol{s}^t\right)\right] && (26) \\ &= \sum_{t=0}^{\infty}\gamma^t\sum_{j\notin\mathcal{N}_i^{\mathrm{L}}}\mathbb{E}\left[r_j\left(s^t_{\mathcal{N}_j^{\mathrm{R}}}, a^t_{\mathcal{N}_j^{\mathrm{R}}}\right)|\left(s^t_{\mathcal{R}_i^{\mathrm{SO}}}, s^t_{\mathcal{V}\setminus\mathcal{R}_i^{\mathrm{SO}}}\right) \sim \right. \\ &\quad \left.P\left(\cdot|\boldsymbol{s}^0 = \boldsymbol{s}, \{\pi_i+\zeta, \pi_{i-}\}\right), \boldsymbol{a}_{i-} \sim \pi_{i-}\left(\cdot|\boldsymbol{s}^t\right)\right] \\ &\quad - \sum_{t=0}^{\infty}\gamma^t\sum_{j\notin\mathcal{N}_i^{\mathrm{L}}}\mathbb{E}\left[r_j\left(s^t_{\mathcal{N}_j^{\mathrm{R}}}, a^t_{\mathcal{N}_j^{\mathrm{R}}}\right)|\left(s^t_{\mathcal{R}_i^{\mathrm{SO}}}, s^t_{\mathcal{V}\setminus\mathcal{R}_i^{\mathrm{SO}}}\right)\right. \\ &\quad \left.\sim P\left(\cdot|\boldsymbol{s}^0 = \boldsymbol{s}, \boldsymbol{\pi}\right), \boldsymbol{a}_{i-} \sim \pi_{i-}\left(\cdot|\boldsymbol{s}^t\right)\right] && (27) \\ &= \sum_{t=0}^{\infty}\gamma^t\sum_{j\notin\mathcal{N}_i^{\mathrm{L}}}\mathbb{E}\left[r_j\left(s^t_{\mathcal{N}_j^{\mathrm{R}}}, a^t_{\mathcal{N}_j^{\mathrm{R}}}\right)|s^t_{\mathcal{V}\setminus\mathcal{R}_i^{\mathrm{SO}}} \sim P(\cdot|\right. \\ &\quad \left.\boldsymbol{s}^0 = \boldsymbol{s}, \{\pi_i+\zeta, \pi_{i-}\}), \boldsymbol{a}_{i-} \sim \boldsymbol{\pi}_{i-}\left(\cdot|\boldsymbol{s}^t\right)\right] \\ &\quad - \sum_{t=0}^{\infty}\gamma^t\sum_{j\notin\mathcal{N}_i^{\mathrm{L}}}\mathbb{E}\left[r_j\left(s^t_{\mathcal{N}_j^{\mathrm{R}}}, a^t_{\mathcal{N}_j^{\mathrm{R}}}\right)|s^t_{\mathcal{V}\setminus\mathcal{R}_i^{\mathrm{SO}}} \sim \right. \\ &\quad \left.P\left(\cdot|\boldsymbol{s}^0 = \boldsymbol{s}, \boldsymbol{\pi}\right), \boldsymbol{a}_{i-} \sim \boldsymbol{\pi}_{i-}\left(\cdot|\boldsymbol{s}^t\right)\right] && (28) \\ &= 0, && (29) \end{aligned}$$

where (27) holds because $r_j(s^t_{\mathcal{N}_j^{\mathrm{R}}}, a^t_{\mathcal{N}_j^{\mathrm{R}}})$ is independent of a_i, (28) used the fact that $r_j(s^t_{\mathcal{N}_j^{\mathrm{R}}}, a^t_{\mathcal{N}_j^{\mathrm{R}}})$ is independent of $s^t_{\mathcal{R}_i^{\mathrm{SO}}}$, (29) is obtained by noting that π_i never influences $s^t_{\mathcal{V}\setminus\mathcal{R}_i^{\mathrm{SO}}}$. □

Appendix B. Experimental details

B.1. Hyperparameters

See Tables 3–5.

Table 3
Hyperparameters used in GRA, GRA-decen.

Hyperparameters	Value
GNN layer hidden dim	16
Num GNN heads	3
GNN activation	ReLU
PPO epoch	15

Table 4
Common hyperparameters used in GRA, GRA-decen and MAPPO.

Common hyperparameters	Value
γ (discount factor)	0.99
GAE λ	0.95
PPO epoch	15
Critic learning rate	8e−5
Actor learning rate	8e−5
Optimizer	Adam
Hidden size	64
Gradient clip norm	10.0

Table 5
Parameters of the multi-echelon inventory optimization problem.

Parameter	Definition	Value
G_c	Types of goods	5
λ_s	Selling price per unit	Derived from the dataset
λ_i	Procurement cost per unit	Derived from the dataset
λ_d	Penalty coefficient for exceeding capacity	0.5
V_a	Total warehouse capacity	1 000
C_t	Transportation lead time	Derived from the dataset
C_o	Ordering cost	10
C_s	Holding cost per unit	0.01
G_r	Quantity of goods in the order submitted to upstream	\
G_i	Inventory level	\
G_e	Quantity exceeding capacity	\
G_d	Downstream order quantity	\
G_s	Quantity sold downstream	\
G_a	Quantity of goods arrived	\
G_t	Quantity of goods in transit	\
$\mathcal{N}_i^u$	The set of upstream warehouses of warehouse i	\
$\mathcal{N}_i^d$	The set of downstream warehouses of warehouse i	\

B.2. Experimental setting for resource allocation

In resource allocation, we set the individual reward as $r_i^t = \sum_{j\in\mathcal{N}_i^R} \tau_j^t$ where $\tau_j^t = 0$ if $s_j^t \geq 0$, and $\tau_j^t = -(s_j^t)^2$ otherwise. The observation graph and reward graph in resource allocation for 20 warehouses are shown in Figs. 8 and 9, respectively. The state transition graph is set the same as the observation graph.

B.3. Experimental setting for multi-echelon inventory optimization

The parameters of the multi-echelon inventory optimization problem are shown in Table 5. Except for G_c, C_o, and C_s, all other parameters are defined per good type; that is, each such parameter is a vector whose components correspond to individual types of goods.

The action of warehouse i is quantity of goods in the order submitted to upstream suppliers, i.e., $a_i = G_{r,i}$, and the observation o_i can be represented as:

$$o_i = \left\{ \lambda_{s,i},\ \lambda_{i,i},\ G_{i,i},\ G_{t,i},\ \{G_{d,i}(j)\}_{j\in\mathcal{N}_i^d} \right\}, \tag{30}$$

where $\{G_{d,i}(j)\}_{j\in\mathcal{N}_i^d}$ denotes the set of order quantities submitted to warehouse i by each downstream warehouse adjacent to it. The reward function of warehouse i can be expressed as:

$$R_i(\boldsymbol{s}^t, \boldsymbol{a}^t) = I_i^t - C_{b,i}^t, \tag{31}$$

$$I_i^t = (\lambda_{s,i}^t - \lambda_{i,i}^t) * G_{s,i}^t - C_{o,i}^t - G_{i,i}^t C_s - G_{e,i}^t \lambda_{i,i}^t \lambda_d, \tag{32}$$

$$C_{b,i}^t = \lambda_b * (\lambda_{s,i}^t - \lambda_{i,i}^t) * \sum_{j\in\mathcal{N}_i^d} G_{b,i}^t(j), \tag{33}$$

$$G_{b,i}^t(j) = \begin{cases} G_{d,i}^t(j) - G_{s,i}^t(j), & G_{d,i}^t(j) > G_{s,i}^t(j), \\ 0, & \text{otherwise}, \end{cases} \tag{34}$$

where I_i^t is the net profit obtained by warehouse i at time t, $C_{b,i}^t$ is the stockout penalty, λ_b is the stockout penalty coefficient, and λ_d is the penalty coefficient for exceeding capacity.

We employ a serial multi-echelon inventory control system consisting of 4 warehouses. The system's state transition graph and reward graph share the same topology, as illustrated in Fig. 10, and the observation graph has an empty edge set, i.e., $\mathcal{E}_O = \emptyset$. The corresponding learning graph is shown in Fig. 11.

Fig. 8. The observation graph in MPE.

Fig. 9. The reward graph in MPE.

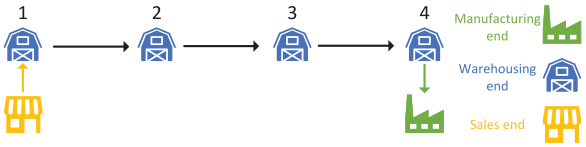


Fig. 10. The state transition graph and the reward graph in the serial system.

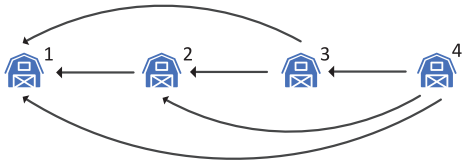


Fig. 11. The learning graph in the serial system.

As with the serial multi-echelon inventory control system, the observation graph of the networked multi-echelon inventory control system has an empty edge set. Its state transition graph and reward graph share the same topology, as illustrated in Fig. 12, and the learning graph is shown in Fig. 13.

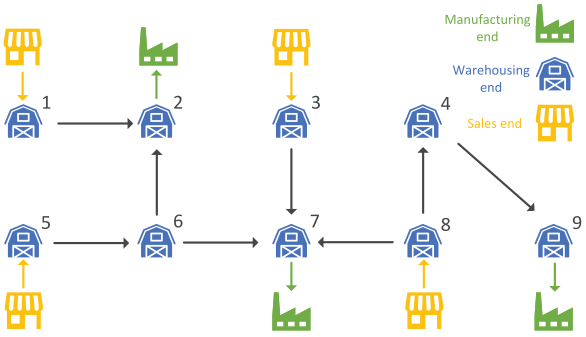


Fig. 12. The state transition graph and the reward graph in the networked system.

The inventory management process at each time step is composed of the following five procedures:

(1) Order Submission: Each warehouse determines its replenishment order quantity based on its current inventory level, historical demand, *etc.*, according to its own policy, and sends this order to its upstream warehouses. This replenishment order directly becomes the demand observed by the upstream warehouse. If the upstream node is a manufacturer, it begins shipping to the warehouse immediately upon receipt of the order; after the necessary lead time, the goods arrive. For warehouse i, let $\mathcal{N}_i^u$ denote the set of its upstream warehouses and $\mathcal{N}_i^d$ the set of its downstream warehouses. The replenishment process

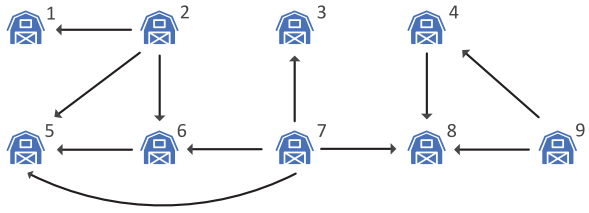


Fig. 13. The learning graph in the networked system.

is expressed by

$$G^t_{d,i}(j) = G^t_{r,j}(i), \tag{35}$$

where $j \in \mathcal{N}^d_i$, $G^t_{r,j}(i)$ is the order quantity submitted at time t by warehouse j to its upstream warehouse i, and $G^t_{d,i}(j)$ is the demand received by warehouse i from warehouse j at time t.

(2) Sales: Each warehouse determines the actual sales quantity based on incoming demand and current inventory, ensuring that sales do not exceed available stock and that demand satisfaction and inventory allocation are balanced. If total demand exceeds available inventory, the allocation to each downstream warehouse is proportional to its share of the total demand:

$$G^t_{s,i}(j) = \begin{cases} G^t_{d,i}(j), & G^t_{i,i} > \sum_{j\in\mathcal{N}^d_i} G^t_{d,i}(j), \\ \dfrac{G^t_{d,i}(j)}{\sum_{j\in\mathcal{N}^d_i} G^t_{d,i}(j)} G^t_{i,i}, & \text{otherwise}, \end{cases} \tag{36}$$

where $G^t_{s,i}(j)$ is the quantity sold by warehouse i to downstream warehouse j at time t.

(3) Goods Arrival: Replenishment shipments arrive after a transportation delay. This arrival process is modeled as

$$G^t_{a,i} = \sum_{j\in\mathcal{N}^u_i} \sum_{k=0}^{t-1} \mathbb{I}_{j,k,i}\, G^k_{s,j}(i), \tag{37}$$

where

$$\mathbb{I}_{j,k,i} = \begin{cases} 1, & \text{if the goods sold by warehouse } j \text{ to} \\ & \text{warehouse } i \text{ at time } k \text{ arrive at time } t, \\ 0, & \text{otherwise.} \end{cases}$$

(4) Goods Reception: Due to capacity constraints, the sum of existing inventory and newly arriving goods cannot exceed the warehouse's storage capacity. If the arriving quantity exceeds available space, the warehouse accepts only up to its maximum capacity and any overflow is liquidated at a lower price. Define

$$\rho^t_i = \min\left(\frac{V_{a,i} - G^t_{i,i}}{G^t_{a,i}}, 1\right), \tag{38}$$

$$B^t_i = G^t_{a,i}\, \rho^t_i, \tag{39}$$

where B^t_i is the actual quantity accepted by warehouse i at time t, and ρ^t_i is the acceptance ratio ($\rho^t_i = 1$ if capacity suffices; otherwise the maximal feasible fraction).

(5) Inventory Update: After sales and arrivals are processed, the system updates each warehouse's inventory level for the next decision epoch:

$$G^{t+1}_{i,i} = G^t_{i,i} - \sum_{j\in\mathcal{N}^d_i} G^t_{s,i}(j) + B^t_i. \tag{40}$$

B.4. Experimental setting for MPE

In MPE, the observation graph and the reward graph are shown in Figs. 8 and 9. We do not consider collisions among agents in the MPE environment, so the state transition graph is completely sparse. We set the individual reward as

$$r^t_i = -0.1(\text{dis}_i - 0.7)^2 - r^t_{\rho i}, \tag{41}$$

where dis_i is the distance between agent i and the target landmark, and we set the distances to make agents distributed within a certain range of the target landmark. We set $r^t_{\rho i} = \sum_{j\in\mathcal{N}^R_i}(\text{ang}_{ij} - 2\pi/n)^2$ as the reward obtained by coordinating the relative positions of the agents, here ang_{ij} denotes the angle between the relative position vector of agent i and the target point and the relative position vector of agent j and the target landmark.

Appendix C. Additional experiment

We investigate the effect of different number of observation GNN layers on the performance in resource allocation for 20 warehouses. As shown in Fig. 14, having a single layer observation GNN appears effective for the considered example. This implies that in this example, it is sufficient for each agent to learn a satisfactory policy based on the observation on its one-hop neighbors in the learning graph.

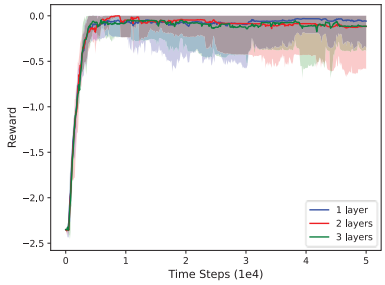


Fig. 14. The algorithm with different number of observation GNN layers in resource allocation.

References

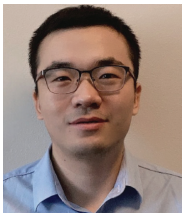
[1] T. Li, G. Peng, Q. Zhu, T. Başar, The confluence of networks, games, and learning a game-theoretic framework for multiagent decision making over networks, IEEE Control Syst. Mag. 42 (4) (2022) 35–67.
[2] M. Mehrabi, W. Masoudimansour, Y. Zhang, J. Chuai, Z. Chen, M. Coates, J. Hao, Y. Geng, Neighbor auto-grouping graph neural networks for handover parameter configuration in cellular network, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, (12) 2023, pp. 14400–14407.
[3] J.E. Flood, Telecommunication Networks, IET, 1997.
[4] V. Popovskij, A. Barkalov, L. Titarenko, Control and Adaptation in Telecommunication Systems: Mathematical Foundations, vol. 94, Springer Science & Business Media, 2011.
[5] H. Sarimveis, P. Patrinos, C.D. Tarantilis, C.T. Kiranoudis, Dynamic modeling and control of supply chain systems: A review, Comput. Oper. Res. 35 (11) (2008) 3530–3561.
[6] M.A. Bellamy, R.C. Basole, Network analysis of supply chain systems: A systematic review and future research, Syst. Eng. 16 (2) (2013) 235–249.
[7] D. Gammelli, J. Harrison, K. Yang, M. Pavone, F. Rodrigues, F.C. Pereira, Graph reinforcement learning for network control via bi-level optimization, 2023, arXiv preprint arXiv:2305.09129.
[8] R. Lowe, Y.I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, I. Mordatch, Multi-agent actor-critic for mixed cooperative-competitive environments, Adv. Neural Inf. Process. Syst. 30 (2017).
[9] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, Y. Wu, The surprising effectiveness of ppo in cooperative multi-agent games, Adv. Neural Inf. Process. Syst. 35 (2022) 24611–24624.
[10] S. Iqbal, F. Sha, Actor-attention-critic for multi-agent reinforcement learning, in: International Conference on Machine Learning, PMLR, 2019, pp. 2961–2970.
[11] S. Sukhbaatar, A. Szlam, R. Fergus, Learning multiagent communication with backpropagation, Adv. Neural Inf. Process. Syst. 29 (2016).

[12] N. Jaques, A. Lazaridou, E. Hughes, C. Gulcehre, P. Ortega, D. Strouse, J.Z. Leibo, N. De Freitas, Social influence as intrinsic motivation for multi-agent deep reinforcement learning, in: International Conference on Machine Learning, PMLR, 2019, pp. 3040–3049.
[13] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, J. Wang, Mean field multi-agent reinforcement learning, in: International Conference on Machine Learning, PMLR, 2018, pp. 5571–5580.
[14] Q. Hao, W. Huang, T. Feng, J. Yuan, Y. Li, GAT-MF: Graph attention mean field for very large scale multi-agent reinforcement learning, in: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '23, Association for Computing Machinery, New York, NY, USA, ISBN: 9798400701030, 2023, pp. 685–697.
[15] C.S. de Witt, T. Gupta, D. Makoviichuk, V. Makoviychuk, P.H. Torr, M. Sun, S. Whiteson, Is independent learning all you need in the starcraft multi-agent challenge?, 2020, arXiv preprint arXiv:2011.09533.
[16] G. Qu, A. Wierman, N. Li, Scalable reinforcement learning of localized policies for multi-agent networked systems, in: Learning for Dynamics and Control, PMLR, 2020, pp. 256–266.
[17] K. Zhang, Z. Yang, H. Liu, T. Zhang, T. Basar, Fully decentralized multi-agent reinforcement learning with networked agents, in: International Conference on Machine Learning, PMLR, 2018, pp. 5872–5881.
[18] X. Liu, H. Wei, L. Ying, Scalable and sample efficient distributed policy gradient algorithms in multi-agent networked systems, 2022, arXiv preprint arXiv:2212.06357.
[19] J. Jiang, C. Dun, T. Huang, Z. Lu, Graph convolutional reinforcement learning, 2018, arXiv preprint arXiv:1810.09202.
[20] S. Nayak, K. Choi, W. Ding, S. Dolan, K. Gopalakrishnan, H. Balakrishnan, Scalable multi-agent reinforcement learning through intelligent information aggregation, in: International Conference on Machine Learning, PMLR, 2023, pp. 25817–25833.
[21] F.A. Oliehoek, C. Amato, A Concise Introduction to Decentralized POMDPs, vol. 1, Springer, 2016.
[22] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, S. Whiteson, Counterfactual multi-agent policy gradients, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, (1) 2018.
[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, 2017, arXiv preprint arXiv:1707.06347.
[24] P. Sunehag, G. Lever, A. Gruslys, W.M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J.Z. Leibo, K. Tuyls, et al., Value-decomposition networks for cooperative multi-agent learning, 2017, arXiv preprint arXiv:1706.05296.
[25] T. Rashid, C. De Witt, G. Farquhar, J. Foerster, S. Whiteson, M. Samvelyan, QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning, in: 35th International Conference on Machine Learning, ICML 2018, 2018, pp. 6846–6859.
[26] K. Son, D. Kim, W.J. Kang, D.E. Hostallero, Y. Yi, Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning, in: International Conference on Machine Learning, PMLR, 2019, pp. 5887–5896.
[27] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, 2017, arXiv preprint arXiv:1710.10903.
[28] S. Shen, Y. Fu, H. Su, H. Pan, P. Qiao, Y. Dou, C. Wang, Graphcomm: A graph neural network based method for multi-agent reinforcement learning, in: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, IEEE, 2021, pp. 3510–3514.
[29] G. Jing, H. Bai, J. George, A. Chakrabortty, P.K. Sharma, Distributed multi-agent reinforcement learning based on graph-induced local value-functions, 2022, arXiv preprint arXiv:2202.13046.
[30] R.J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Mach. Learn. 8 (1992) 229–256.
[31] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, Y. Sun, Masked label prediction: Unified message passing model for semi-supervised classification, 2020, arXiv preprint arXiv:2009.03509.
[32] J. Schulman, P. Moritz, S. Levine, M. Jordan, P. Abbeel, High-dimensional continuous control using generalized advantage estimation, 2015, arXiv preprint arXiv:1506.02438.
[33] K. Zhang, Z. Yang, T. Başar, Decentralized multi-agent reinforcement learning with networked agents: Recent advances, Front. Inf. Technol. Electron. Eng. 22 (6) (2021) 802–814.
[34] M. Samvelyan, T. Rashid, C.S. De Witt, G. Farquhar, N. Nardelli, T.G. Rudner, C.-M. Hung, P.H. Torr, J. Foerster, S. Whiteson, The starcraft multi-agent challenge, 2019, arXiv preprint arXiv:1902.04043.
[35] J. Suarez, Y. Du, P. Isola, I. Mordatch, Neural MMO: A massively multiagent game environment for training and evaluating intelligent agents, 2019, arXiv preprint arXiv:1903.00784.
[36] C. Guestrin, M. Lagoudakis, R. Parr, Coordinated reinforcement learning, in: ICML, vol. 2, 2002, pp. 227–234.
[37] X. Yang, Z. Liu, W. Jiang, C. Zhang, L. Zhao, L. Song, J. Bian, A versatile multi-agent reinforcement learning benchmark for inventory management, 2023, arXiv preprint arXiv:2306.07542.
[38] A. Agarwal, S. Kumar, K. Sycara, Learning transferable cooperative behavior in multi-agent teams, 2019, arXiv preprint arXiv:1906.01202.

Jingcheng Tang received the B.E. degree in vehicle engineering from Xihua University, China, in 2022. He is currently a graduate student in Electronic Information at School of Automation, Chongqing University. His research interests include multi-agent reinforcement learning and networked systems.

Peng Zhou received the B.E. degree in Geological Engineering from Southwest Jiaotong University, China, in 2022. He is currently a graduate student in Control Science and Engineering at School of Automation, Chongqing University. His research interests include reinforcement learning, multi-agent coordination, and causal inference.

He Bai received his Ph.D. degree in Electrical Engineering from Rensselaer Polytechnic Institute, Troy, NY, in 2009. From 2009 to 2010, he was a postdoctoral researcher at Northwestern University, Evanston, IL. From 2010 to 2015, he was a Senior Research and Development Scientist at UtopiaCompression Corporation, Los Angeles, CA. In 2015, he joined the School of Mechanical and Aerospace Engineering at Oklahoma State University, Stillwater, OK, where he is currently an associate professor. His research interests include distributed estimation, control and learning, reinforcement learning, nonlinear control, and robotics.



Gangshan Jing received the Ph.D. degree in Control Theory and Control Engineering from Xidian University, Xi'an, China, in 2018. From 2016 to 2017, he was a research assistant at Hong Kong Polytechnic University, Hong Kong. From 2018–2021, he was a postdoctoral researcher at Ohio State University, USA and North Carolina State University, USA, respectively. Since Dec. 2021, he has been a faculty member at Chongqing University, China. His research interests include control, optimization, and machine learning for network systems.