NVIF: Neighboring Variational Information Flow for Cooperative Large-Scale Multiagent Reinforcement Learning

Jiajun Chai, *Graduate Student Member, IEEE*, Yuanheng Zhu, *Senior Member, IEEE*, and Dongbin Zhao, *Fellow, IEEE*

***Abstract*— Communication-based multiagent reinforcement learning (MARL) has shown promising results in promoting cooperation by enabling agents to exchange information. However, the existing methods have limitations in large-scale multiagent systems due to high information redundancy, and they tend to overlook the unstable training process caused by the online-trained communication protocol. In this work, we propose a novel method called neighboring variational information flow (NVIF), which enhances communication among neighboring agents by providing them with the maximum information set (MIS) containing more information than the existing methods. NVIF compresses the MIS into a compact latent state while adopting neighboring communication. To stabilize the overall training process, we introduce a two-stage training mechanism. We first pretrain the NVIF module using a randomly sampled offline dataset to create a task-agnostic and stable communication protocol, and then use the pretrained protocol to perform online policy training with RL algorithms. Our theoretical analysis indicates that NVIF-proximal policy optimization (PPO), which combines NVIF with PPO, has the potential to promote cooperation with agent-specific rewards. Experiment results demonstrate the superiority of our method in both heterogeneous and homogeneous settings. Additional experiment results also demonstrate the potential of our method for multitask learning.**

***Index Terms*— Large-scale multiagent, neighboring communication, reinforcement learning (RL), variational information flow.**

I. Introduction

MULTIAGENT reinforcement learning (MARL) utilizes RL to tackle problems in multiagent systems. Previous works have addressed cooperative scenarios, such as controlling robot swarms with limited sensing capabilities [1], [2], mastering multiagent coordination [3], [4], and micromanagement tasks in real-time strategy (RTS) games [5], [6], [7]. To promote cooperation in multiagent systems, several methods adopt the *centralized training with decentralized execution* (CTDE) framework [8]. However, CTDE-based methods face scalability and credit assignment issues, as they still require centralized learning, which can be a challenge, especially in large-scale multiagent systems.

In contrast, *communication-based* MARL is a promising solution for promoting cooperation in multiagent systems. This approach involves exchanging local information between agents using a communication protocol to aid decision-making. However, providing communication for multiagent systems gives rise to the problem of *information redundancy*, particularly in large-scale systems [9]. This makes it challenging for agents to discern the important information amidst a vast amount of available information. Some methods try to mitigate the redundancy using event-triggered communication [10], [11], [12] or gated communication [13], [14] to reduce the communication frequency. However, these studies do not pay attention to the communication protocol and only share the agents' local information, resulting in their poor performance.

Recent works focus on training their communication protocol based on specific graph structures to mitigate the information redundancy. Some use a learned graph structure [15], [16], [17], while others use a specific rule-based structure, such as neighborhood [18], [19], hierarchical [20], and central [21]. Methods that utilize the mean-field approximation can also be seen as providing communication based on the virtual central structure [22], [23]. However, these communication protocols are trained using online policy training process, which can result in the policy breakdown issue. Specifically, earlier trained policies may not match the newly updated protocols.

Another focus of this work is to promote cooperation. The existing works can utilize both team rewards and agent-specific rewards to optimize the policies. In this work, we primarily focus on the agent-specific reward setting, where the reward provided by the environment may differ among agents. These agent-specific rewards must satisfy some conditions, such as monotonicity [8], to avoid the potential adversarial interactions among agents [24], [25], [26]. While the existing methods

Manuscript received 18 April 2022; revised 13 March 2023 and 26 May 2023; accepted 23 August 2023. Date of publication 6 September 2023; date of current version 3 December 2024. This work was supported in part by the Strategic Priority Research Program of Chinese Academy of Sciences (CAS) under Grant XDA27030400, in part by the National Natural Science Foundation of China under Grant 62293541 and Grant 62136008, in part by the National Key Research and Development Program of China under Grant 2018AAA0102404, and in part by the Youth Innovation Promotion Association of CAS. *(Corresponding author: Yuanheng Zhu.)*

The authors are with the State Key Laboratory of Multimodal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, and also with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: yuanheng.zhu@ia.ac.cn).

Digital Object Identifier 10.1109/TNNLS.2023.3309608

using agent-specific rewards have made some progress, they do not provide any theoretical analysis to ensure cooperation [27], [28], [29].

Motivated by the aforementioned weaknesses, this article proposes a new method called neighboring variational information flow (NVIF) to provide information exchange among neighboring agents. The main contribution is twofold.

1) We propose a new approach for enhancing communication among neighboring agents by introducing the concept of maximum information set (MIS). MIS allows agents to gather more information than the existing methods by collecting the most information possible. The proposed NVIF method compresses MIS into a compact latent state, which serves as an auxiliary feature to help agents improve their performance.
2) We present a two-stage training mechanism to stabilize the overall training process. We first pretrain the communication protocol using a randomly sampled offline dataset and combine NVIF with RL algorithms for online policy learning. We provide a theoretical analysis to demonstrate the potential of NVIF-proximal policy optimization (PPO), which combines NVIF with PPO, to promote cooperation in the agent-specific reward setting.

To evaluate the effectiveness of the proposed method, we conduct experiments on two large-scale multiagent tasks: a heterogeneous environment with continuous action spaces and a homogeneous environment with discrete action spaces. These two tasks entail scenarios of different scales, distinguished by the number of agents, where greater numbers of agents increase the training difficulty. We compare our method with state-of-the-art methods and find that the NVIF-PPO outperforms other methods in most tasks, particularly in those at larger scales. Furthermore, we conduct additional experiments to demonstrate its scalability to unseen tasks.

A. Related Work

Communication-based MARL methods are designed to facilitate information exchange among agents, allowing them to access more information beyond their local observations. Early methods establish direct communication channels between all pairs of agents, such as DIAL [30] and CommNet [31]. These methods may suffer from severe *information redundancy*, making it challenging for agents to determine the importance of the received information, especially in large-scale multiagent systems. To address this issue, event-triggered communication is employed to provide communication with specific events, such as ETCNet [12] and event-based adaptive dynamic programming (ADP) [32]. SchedNet [13] and IC3Net [14] use a gated module to reduce communication by shielding some agents. However, these studies only focus on reducing communication frequency and overlook the design of the communication protocol. They only share the agents' local information among agents, resulting in their poor performance. In this work, we introduce the concept of MIS, which contains the most information can be obtained in the system, to further enrich the agent observation. The proposed NVIF method utilizes the compression capacity of variational autoencoder (VAE) to compress the MIS into a compact latent state to better mitigate the redundancy.

One of the major topics in recently published communication-based MARL methods is learning a communication protocol based on specific graph structures. Some works use the graph structure learned during the MARL training process. Many works dynamically generate an online-learned graph for communication, such as DGN [33], MAGIC [16], and MARGIN [34]. I2C [9] pretrains a prior network to help agents determine whom to communicate with, while ATOC [35] is proposed to identify the message sender or receiver. Other works adopt a rule-based graph for communication. LSC [20] designs a hierarchical mechanism to provide a more effective graph. HAMMER [21] and CCOMA [36] adopt a centralized topology that allows a powerful central agent to communicate with the others. GraphComm [37] and NCC [19] adopt neighboring communication for communication, which is a more reasonable graph. Mean-field approximation-based methods, such as MFQ [22] and ACM [23], have also been developed to provide communication on a virtual central structure. MASIA [38] proposes two self-supervised loss functions to generate common information-aggregated features for agents' decision-making. However, these methods train their communication protocol during the MARL training process, leading to the unstable protocol to reduce the overall training efficiency. To address this, we present a two-stage training mechanism to pretrain the protocol using offline data before the online policy training process.

Recently, actor-critic methods have shown promising results in MARL, particularly those using PPO [39], a popular approach in single-agent RL. Independent PPO (IPPO) [40] employs PPO to enable agents to be trained independently with a team reward. To further investigate the potential of PPO in multiagent scenarios, MAPPO [41] summarizes several techniques to improve IPPO's performance. CoPPO [42] is proposed to promote cooperation based on MAPPO and includes some theoretical analysis. However, these approaches fail in large-scale multiagent systems, because their value factorization technique is not appropriate for large-scale systems. Another thread of methods utilizes the agent-specific rewards for each agent to optimize policies. Zhang et al. [27] propose two fully decentralized actor-critic methods with agent-specific rewards and provide convergence results under linear approximation. cA2C [28] uses a centralized value network and decentralized policy to handle large-scale fleet management. G2ANet [29] employs graph neural networks to learn adaptive and dynamic attention values without team rewards. However, optimizing policies using agent-specific rewards may not promote cooperation due to the nonstationary problem. In this work, the proposed method is compatible with two reward settings, and we provide a theoretical analysis to show the potential of our method to promote cooperation in the agent-specific reward setting.

B. Organization

This article is organized as follows. Section II introduces the problem formulation of MARL with communication and the

background knowledge on PPO. Section III proposes NVIF with its implementation and training algorithm and provides a theoretical analysis for its potential to promote cooperation. Sections IV and V present the experiments and results. Finally, Section VI concludes this article.

II. Background

A. Problem Formulation

In this section, we describe the fully cooperative multiagent task with partially observable environment in which agents communicate with each other to exchange information. We define the task as a tuple $\mathscr{U} = \{n, \mathbb{S}, \mathbb{O}, \mathbb{A}, \mathbb{T}, \mathbb{R}, \gamma\}$, where n is the number of agents, $\mathbb{S}$ is the global state space, and $\mathbb{O} = \{O_i\}_{i=1}^n$ is the joint observation space, with O_i being the observation space of agent i. Similarly, $\mathbb{A} = \{A_i\}_{i=1}^n$ is the joint action space, with A_i being the action space of agent i. The observation and action spaces of different agents are not necessarily the same. $\mathbb{T}$ is the transition function, and $\mathbb{R} = \{R_i\}_{i=1}^n$ is the reward function, where R_i calculates the reward for agent i at timestep t, denoted as $r_{i,t}$, based on changes in the global state. Finally, γ denotes the discount factor.

In the interaction process between the multiagent system and environment, the system takes the joint action $\boldsymbol{a}_t = \{a_{1,t}, \ldots, a_{n,t}\} \in \mathbb{A}$ and receives the immediate reward $\boldsymbol{r}_t = \{r_{1,t}, \ldots, r_{n,t}\}$ from the environment. If the environment provides a team reward R_t, then $r_{1,t} = \cdots = r_{n,t} = R_t$; otherwise, the rewards of each agent are not necessarily the same. The introduction of a team reward may lead to a credit assignment problem, where the contribution of each agent needs to be evaluated, potentially reducing the training speed of the method. After executing the joint action, the next global state s_{t+1} is produced according to the probability specified by $\mathbb{T}(s_{t+1}|s_t, \boldsymbol{a}_t)$.

In a large-scale multiagent system, the huge gap between the local observation of agents and the global state of the system creates significant obstacles to cooperation. To address this, efficient communication between agents is necessary to promote cooperation. By communicating, agents can obtain information about the global state $s_t \in \mathbb{S}$ of the system at timestep t, which is referred to as the *latent state* $\hat{s}_{i,t}$ obtained by each agent $i \in [1, n]$. This latent state can serve as an auxiliary feature for agent decision-making. Each agent maintains a policy $\pi_i(a_{i,t}|o_{i,t}, \hat{s}_{i,t})$ that considers both its local observation $o_{i,t}$ and the obtained latent state $\hat{s}_{i,t}$ to make decisions, where $a_{i,t}$ is the action.

B. Independent Proximal Policy Optimization

IPPO is an MARL algorithm that uses PPO to independently optimize the discounted cumulative reward $\xi_{i,t} = \sum_{j=0}^{\infty} \gamma^j r_{i,t+j}$ for each agent i in the multiagent system. We first provide the definitions of the state value function and advantage estimator. The state value function $V_i^{\boldsymbol{\pi}}(s_t)$ of agent i can be defined as follows:

$$V_i^{\boldsymbol{\pi}}(s_t) = \mathbb{E}_{\boldsymbol{\pi}}\left[r_{i,t+1} + \gamma r_{i,t+2} + \cdots | s_t\right] \tag{1}$$

where the formula on the right indicates the expected value of the future discounted cumulative rewards under joint policy $\boldsymbol{\pi} = \{\pi_i\}_{i=1}^n$. Then, we define the advantage estimator based on generalized advantage estimator (GAE) [43]

$$\begin{aligned} A_i^{\boldsymbol{\pi}} &= \delta_{i,t} + (\gamma\lambda)\delta_{i,t+1} + \cdots + (\gamma\lambda)^{T-t-1}\delta_{i,T-1} \\ \text{where } \delta_{i,t} &= r_{i,t+1} + \gamma V_i^{\boldsymbol{\pi}}(s_{t+1}) - V_i^{\boldsymbol{\pi}}(s_t) \end{aligned} \tag{2}$$

where s_{t+1} is the next state, $\delta_{i,t}$ is the temporal difference, λ is the hyperparameter of GAE, and T is the timestep at the end of an episode. GAE enables more efficient and stable estimation of a policy's advantage function, while utilizing λ to balance the trade-off between bias and variance in the estimation process. They are abbreviated as $V_{i,t}$ and $A_{i,t}$.

PPO algorithm trains the agent's actor network by maximizing the following objective:

$$\begin{aligned} \mathcal{L}_{\text{ar}}(\boldsymbol{\theta}) &= \hat{\mathbb{E}}_t\left[\sum_{i=1}^{n} \min\left(\rho_{i,t}A_{i,t}, \text{clip}\left(\rho_{i,t}, 1-\epsilon, 1+\epsilon\right)A_{i,t}\right)\right] \\ &= \hat{\mathbb{E}}_t\left[\sum_{i=1}^{n} \alpha_{i,t}A_{i,t}\right] \end{aligned} \tag{3}$$

where $\boldsymbol{\theta}$ is the network parameter of the joint policy. If the algorithm is applied to heterogeneous environment, each agent has a unique network parameter $\boldsymbol{\theta} = \{\theta_i\}_{i=1}^n$; otherwise, all agents share the same network parameter $\boldsymbol{\theta} = \theta_i, \forall i$. $\rho_{i,t} = (\pi_i(a_{i,t}|\hat{s}_{i,t}, o_{i,t}; \theta_i))/(\pi_i(a_{i,t}|\hat{s}_{i,t}, o_{i,t}; \theta_i^{\text{old}}))$ is the probability ratio that measures the difference between $\pi_i(\theta_i)$ and $\pi_i(\theta_i^{\text{old}})$, ϵ is the clip coefficient, and $\alpha_{i,t}$ is the simplified clipped ratio. The critic network is updated by minimizing the following loss function:

$$\mathcal{L}(\boldsymbol{\phi}) = \mathbb{E}_t\left[\sum_{i=1}^{n}\left(V_i\left(\hat{s}_{i,t}, o_{i,t}; \phi_i\right) - \xi_{i,t}\right)^2\right] \tag{4}$$

where $\boldsymbol{\phi}$ is the parameter of the agents' critic network. $\boldsymbol{\phi}$ takes the similar setting as $\boldsymbol{\theta}$.

III. Method

In this section, we focus on promoting cooperation in large-scale multiagent systems and propose a novel method called NVIF. To enrich local observations, NVIF enables agents to exchange information by combining graph convolutional network (GCN) and VAE to compress the MIS into a compact latent state. As illustrated in Fig. 1, we introduce a two-stage training mechanism to enhance the stability of the overall training process. In Stage 1, we randomly sample an offline dataset and pretrain the NVIF module as the communication protocol for information exchange among agents. In Stage 2, we use the latent state output by the pretrained protocol as an auxiliary feature for agent decision-making and train their policies using MARL methods.

In Section III-A, we begin by formalizing the information exchange process in multiagent systems. Next, we present our proposed NVIF method and its corresponding training algorithm. Finally, we combine NVIF with the PPO algorithm and provide a theoretical analysis to demonstrate the potential of our method in promoting cooperation.

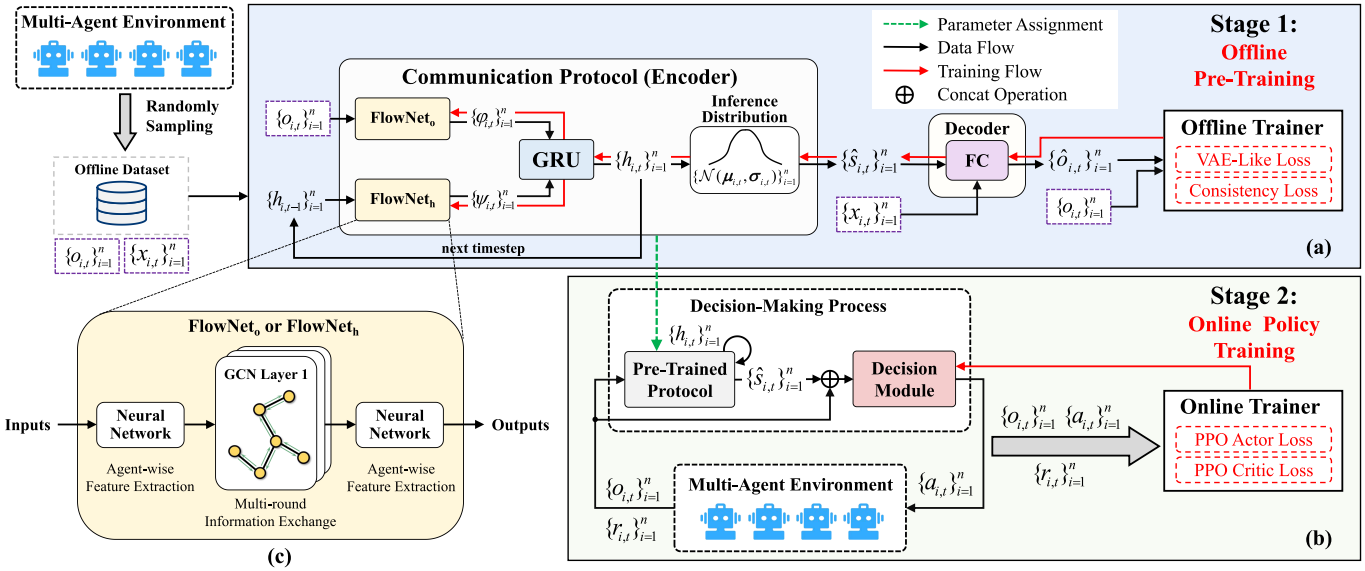

Fig. 1. Overall framework of our method. (a) and (b) Two-stage training mechanism. In the first stage, we pretrain a stable communication protocol using an offline dataset randomly sampled from the multiagent environment. This protocol serves as the foundation for the second stage, where we use an online trainer to optimize the decision module based on data obtained from interactions with the environment. (c) FlowNet in more detail.

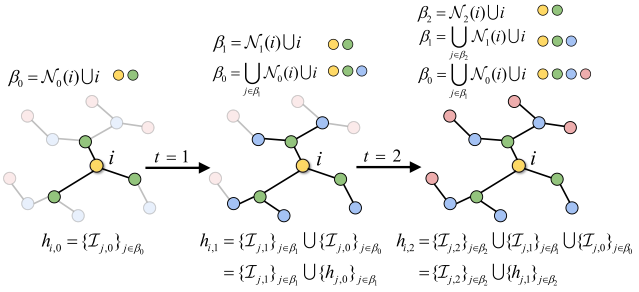

Fig. 2. Communication process to collect MIS for agent i. MIS contains the most information that can be obtained through communication, including the current information and historical information.

A. Neighboring Variational Information Flow

In this work, we represent the relationship between agents using the neighboring structure and enable information exchange among them. Our method does not impose any constraints on the design of neighboring relations, allowing flexibility to adapt to task requirements. Communication in a large-scale multiagent system can result in an overwhelming amount of information for each agent, leading to the issue of information redundancy. To address this, we utilize the compression capacity of VAE to compress the MIS into a compact latent state to better mitigate the redundancy.

Neighboring communication is a trade-off between fully communication and noncommunication. As shown in Fig. 2, each agent can receive information from its neighbors as well as from its second-order neighbors at the last timestep and so on. Therefore, the MIS that can be obtained by agent i at timestep t can be expressed as follows:

$$h_{i,t} = \bigcup_{k=t:0} \left\{ \{\mathcal{I}_{j,k}\}_{j\in\beta_k}, \beta_k = \bigcup_{j\in\beta_{k+1}} \mathcal{N}_k(j) \cup j \right\} \tag{5}$$

where $h_{i,t}$ denotes the MIS, $\mathcal{I}_{j,k}$ is the information shared by agent j at timestep k, and $\mathcal{N}_k(j)$ are its neighbors. β_k is the set of agents whose information shared at timestep k can be accessed by agent i now, where β_{t+1} is initialized as $\{i\}$. Take $k = t$ as an example, and $\beta_t = \{\mathcal{N}_t(i) \cup i\}$ represents the first-order neighbors, which contains the neighbors of agent i at timestep t and the agent itself. Furthermore, the second-order neighbors $\beta_{t-1} = \{\cup_{j\in\beta_t} \mathcal{N}_{t-1}(j) \cup j\}$ include the neighbors of all agents in β_t at the last timestep and so on. We refer to this process as the *neighboring information flow*, which denotes the flow of information in a multiagent system.

Furthermore, the notation of $h_{i,t}$ can be simplified in a recurrent way as follows:

$$\begin{aligned} h_{i,t} &= \bigcup_{k=t-1:0} \left\{ \{\mathcal{I}_{j,k}\}_{j\in\beta_k}, \beta_k = \bigcup_{j\in\beta_{k+1}} \mathcal{N}_k(j) \cup j \right\} \\ &\quad \cup \{\mathcal{I}_{j,t}\}_{j\in\mathcal{N}_t(i)\cup i}, \quad \beta_t = \mathcal{N}_t(i) \cup i \\ &= \{h_{j,t-1}\}_{j\in\beta_t} \cup \{\mathcal{I}_{j,t}\}_{j\in\mathcal{N}_t(i)\cup i} \\ &= \{h_{j,t-1}\}_{j\in\mathcal{N}_t(i)\cup i} \cup \{\mathcal{I}_{j,t}\}_{j\in\mathcal{N}_t(i)\cup i} \end{aligned} \tag{6}$$

where $h_{i,t}$ is decomposed into two parts.

1) The first part is the set that contains the last MIS of the neighbors of agent i and itself, and the initial value is $h_{i,-1} = \varnothing$. We define it as $\psi_{i,t} = \{h_{j,t-1}\}_{j\in\mathcal{N}_t(i)\cup i}$.
2) The second part is the set aggregated from the information currently shared by the neighbors of agent i and itself. We define it as $\varphi_{i,t} = \{\mathcal{I}_{j,t}\}_{j\in\mathcal{N}_t(i)\cup i}$.

The information contained in $h_{i,t}$ may be redundant for the needs of the agent. To address this, we propose a new method called NVIF, which employs a VAE module to improve communication efficiency. The encoder part of NVIF compresses the information in $h_{i,t}$ into a latent state that serves as an auxiliary feature for agent's decision-making. The encoder part can be represented as follows:

$$\hat{s}_{i,t} \sim q\left(s|\psi_{i,t}, \varphi_{i,t}\right) \tag{7}$$

where $q(s|\psi_{i,t}, \varphi_{i,t})$ is the inference distribution with the MIS as its input and $\hat{s}_{i,t}$ represents the latent state output from the encoder part.

In the decoder part, we evaluate the compressed latent state by reconstructing the joint observation of agents, instead of learning communication protocols through MARL training. As a result, the communication protocol trained by NVIF has the potential to be applied to multitask learning. The loss function for NVIF is adapted from the loss function used in VAE. It can be written as follows:

$$\mathcal{L}_v = \frac{1}{n}\sum_{i=1}^{n}\Big[\mathbb{E}_{\hat{s}_{i,t}\sim q(s|\psi_{i,t},\varphi_{i,t})}\mathrm{BL}\big[o_{i,t}, \hat{o}_{i,t}\big] + \mathrm{KL}\big[q\big(s|\psi_{i,t}, \varphi_{i,t}\big)|p(s)\big]\Big] \tag{8}$$

where $\hat{o}_{i,t}$ represents the reconstructed observation output by the decoder $\mathcal{D}(\hat{s}_{i,t}, x_{i,t})$, whose inputs are the latent state $\hat{s}_{i,t}$ and the unique information $x_{i,t}$, such as the position of agent i. $\mathrm{BL}[\cdot,\cdot]$ is the binary cross-entropy loss to measure the difference between two input vectors. $p(s)$ is the prior distribution of the latent state, which is set to the standard normal distribution $\boldsymbol{N}(\boldsymbol{0}, \boldsymbol{I})$. The first part of this loss function is the reconstruction loss of the joint observation, and the second part is the KL divergence between the inference distribution and the prior distribution.

However, directly training with (8) may cause the latent state to degenerate into the current local observation. To prevent this, we introduce the following additional consistency loss based on (8):

$$\mathcal{L}_c = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{s}_{i,t} - \frac{1}{n}\sum_{j=1}^{n}\hat{s}_{j,t}\right)^2. \tag{9}$$

This loss function encourages agents to preserve the same latent state. When combined with the reconstructing ability provided by (9), the same latent state for all agents means that each agent can reconstruct its local observation based on it. Consequently, the latent state can serve as a representation of the multiagent system's global state, encompassing the information required by all agents. The overall loss function can be summarized as follows:

$$\mathcal{L} = \mathcal{L}_v + \alpha\mathcal{L}_c \tag{10}$$

where α balances the relative effect of the main loss and the consistency loss. Minimizing this loss function enables each agent to obtain an efficient latent state.

B. Training Algorithm of NVIF

This section introduces a novel network architecture for implementing NVIF and the offline pretraining stage. As described in (6), the MIS can be recursively decomposed. To enable information exchange within the neighboring communication graph structure, we introduce the GCN [44] into the encoder part of NVIF and present a network architecture FlowNet.

As shown in Fig. 1(c), FlowNet integrates a multilayer GCN with two neural networks, which can be of any type, such as fully connected (FC) or convolutional layers. The first neural network performs agentwise operations to extract features locally from the original inputs. Subsequently, the GCN layer encodes both the graph structure and the extracted features H_t^0 using the adjacency matrix G_t. The multilayer GCN propagation rule is summarized as follows [44]:

$$H_t^{l+1} = \mathrm{ReLU}\Big(\tilde{B}_t^{-\frac{1}{2}}\tilde{G}_t\tilde{B}_t^{-\frac{1}{2}}H_t^l W^l\Big) \tag{11}$$

where H_t^l is the feature extracted by the lth layer and H_t^0 can be $\{o_{i,t}\}_{i=1}^n$ or $\{h_{i,t-1}\}_{i=1}^n$. If there exist L layers of GCN, then H_t^{L+1} is $\boldsymbol{\varphi}_t = \{\varphi_{i,t}\}_{i=1}^n$ or $\boldsymbol{\psi}_t = \{\psi_{i,t}\}_{i=1}^n$. $\tilde{G}_t = G_t + I$ is the adjacency matrix G_t with self-loop I. $\tilde{B}_t = \mathrm{diag}(\sum_j \tilde{G}_t[:, j])$ is a diagonal degree matrix whose elements are the sum of each row of $\tilde{G}_t$. W^l is a trainable parameter.

The multiple GCN layers can be perceived as an instance of *multiround communication*, where the number of rounds is equal to the number of layers. Augmenting the number of GCN layers can enhance the quantity of information exchange per timestep, but can also increase the training difficulty. As shown in Fig. 1(a), $\mathrm{FlowNet}_o$ and $\mathrm{FlowNet}_h$ take observations and hidden states as inputs, respectively, and output the corresponding extracted features

$$\begin{aligned}\{\varphi_{i,t}\}_{i=1}^n &= \mathrm{FlowNet}_o\big(\{o_{i,t}\}_{i=1}^n\big)\\ \{\psi_{i,t}\}_{i=1}^n &= \mathrm{FlowNet}_h\big(\{h_{i,t}\}_{i=1}^n\big)\end{aligned} \tag{12}$$

where $\mathrm{FlowNet}_o$ and $\mathrm{FlowNet}_h$ process their inputs independently.

Besides, we introduce a gated recurrent unit (GRU) layer [45], which is a kind of recurrent network to process sequential inputs, to realize the recurrent property of MIS

$$h_{i,t}, \big(\boldsymbol{\mu}_{i,t}, \boldsymbol{\sigma}_{i,t}\big) = \mathbf{GRU}\big(\varphi_{i,t}, \psi_{i,t}\big). \tag{13}$$

For each agent, it takes $\varphi_{i,t}$ and $\psi_{i,t}$ as inputs and outputs the next hidden state. $\boldsymbol{\mu}_{i,t}$ and $\boldsymbol{\sigma}_{i,t}$ are the mean and standard deviation of the inference distribution $\boldsymbol{N}(\boldsymbol{\mu}_{i,t}, \boldsymbol{\sigma}_{i,t})$, which is a normal distribution. By sampling from the inference distribution for each agent, the MIS is compressed into a latent state

$$\hat{s}_{i,t} \sim \boldsymbol{N}\big(\boldsymbol{\mu}_{i,t}, \boldsymbol{\sigma}_{i,t}\big). \tag{14}$$

As shown in Fig. 1(a), for the decoder part, we concatenate the unique information $x_{i,t}$ with the latent state and use an FC network as the decoder part to get the reconstructed observation $\hat{o}_{i,t}$

$$\hat{o}_{i,t} = \mathcal{D}\big(\hat{s}_{i,t}, x_{i,t}\big) \tag{15}$$

where $\mathcal{D}(\cdot)$ is the decoder network.

To collect the randomly sampled offline dataset depicted in Fig. 1, we first execute random policies to interact with the environment, storing observations $\boldsymbol{o}_t$, positions $\boldsymbol{x}_t$, and the adjacency matrix G in the memory buffer $\mathcal{M}_{\mathrm{nvif}}$. Then, we use the forward rules described above to compute the reconstruction $\hat{\boldsymbol{o}}_t = \{\hat{o}_{i,t}\}_{i=1}^n$ and optimize the parameters of NVIF with the loss function shown in (10). By iterating through all episodes in $\mathcal{M}_{\mathrm{nvif}}$, one training epoch is completed. The process is repeated for multiple epochs until convergence. After the pretraining of the NVIF module, we use its encoder

Algorithm 1 PPO With NVIF in Large-Scale Multiagent Systems

Initialize the parameters of the actor network $\boldsymbol{\theta}$ and critic network $\boldsymbol{\phi}$;
Initialize the replay buffer $\mathcal{M}_{\text{ppo}}$;
for $epoch = 1$ *to* m **do**
  for $episode = 1$ *to* b **do**
    Initialize the hidden state $\boldsymbol{h}_0$ as zero vectors;
    for $t = 1$ *to* T **do**
      Collect joint observation $\boldsymbol{o}_t$ and last hidden state $\boldsymbol{h}_t$;
      Get latent state $\hat{\boldsymbol{s}}_t$ by NVIF;
      Choose actions for agents by actor network;
      Get state values by the critic network;
      Execute joint action $\boldsymbol{a}_t$ and collect rewards $\boldsymbol{r}_t$ for all agents;
      Store $\{\boldsymbol{o}_t, \hat{\boldsymbol{s}}_t, \boldsymbol{a}_t, \boldsymbol{p}_t, \boldsymbol{r}_t, \boldsymbol{V}_t\}$ into $\mathcal{M}_{\text{ppo}}$;
    end
    Compute $\{A_{i,t}\}_{t=1}^{T}$ and $\{\xi_{i,t}\}_{t=1}^{T}$ for all agents;
    Store $\{\boldsymbol{A}_t, \boldsymbol{G}_t\}_{t=1}^{T}$ into the replay buffer;
  end
  for *Update times from* 1 *to* k **do**
    Update the parameters of actor network $\boldsymbol{\theta}$ by maximizing Eq. (3);
    Update the parameters of critic network $\boldsymbol{\phi}$ by minimizing Eq. (4);
  end
end

part to serve as the communication protocol for multiagent systems. This pretrained protocol ensures a stable information exchange process for the MARL training and has the potential for application in multitask learning scenarios.

C. MARL With NVIF

This section presents the online policy training stage. We utilize the communication protocol pretrained in Section III-B to enrich the agents' observations and promote cooperation. We combine NVIF and PPO to train agents in the large-scale multiagent system and refer this algorithm as NVIF-PPO. If the environment provides team rewards for the agents, NVIF-PPO can be seen as a communication-based version of MAPPO [41], which has been shown to promote cooperation among agents. In cases where the environment only provides agent-specific rewards, we provide a theoretical analysis for the potential of NVIF-PPO to promote cooperation in large-scale multiagent systems.

In Algorithm 1, we realize the online policy training stage in Fig. 1(b) by adopting the framework of PPO to train each agent. At each timestep t, we collect the local observations and hidden states from the neighbors of each agent i and itself and use the communication protocol to compute the latent state $\hat{s}_{i,t}$ and the next hidden state $h_{i,t}$. Each agent chooses action according to $o_{i,t}$ and $\hat{s}_{i,t}$ as follows:

$$a_{i,t} \sim \pi_i\left(\cdot|\hat{s}_{i,t}, o_{i,t}; \theta_i\right) \tag{16}$$

where θ_i is the parameters of the agent i's actor network. Similarly, the critic network is presented as $V_i(\hat{s}_{i,t}, o_{i,t}; \phi_i)$, where ϕ_i is its parameter. Then, the environment executes these actions and feeds back the reward $r_{i,t+1}$ for each agent.

For each episode, we store $\{\boldsymbol{o}_t, \hat{\boldsymbol{s}}_t, \boldsymbol{a}_t, \boldsymbol{p}_t, \boldsymbol{r}_t, \boldsymbol{V}_t\}$ into the replay buffer $\mathcal{M}_{\text{ppo}}$ in a chronological order, where these data include all agents, such as $\boldsymbol{o}_t = \{o_{i,t}\}_{i=1}^{n}$ and $\hat{\boldsymbol{s}}_t = \{\hat{s}_{i,t}\}_{i=1}^{n}$. $\boldsymbol{p}_t = \{p_{i,t}\}_{i=1}^{n}$ is the probability that action $a_{i,t}$ is selected by policy $\pi_i(a_{i,t}|\hat{s}_{i,t}, o_{i,t}; \theta_i)$. $\boldsymbol{V}_t$ is used for the advantage estimator $\boldsymbol{A}_t = \{A_{i,t}\}_{i=1}^{n}$, which is calculated and stored into the replay buffer at the end of an episode. Furthermore, $\boldsymbol{r}_t = \{r_{i,t}\}_{i=1}^{n}$ is used to calculate $\xi_{i,t}$, which is also stored into the replay buffer for future calculation. We train the actor network and the critic network by optimizing the loss functions presented by (3) and (4).

In the case of team reward setting, the agents share the same team reward and aim to maximize it. We begin by defining the variables based on the team reward R_t. Given the team reward R_t of the system, the objective used to train the policy by team reward can be defined as follows:

$$\begin{aligned}\mathcal{L}_{\text{tr}} &= \hat{\mathbb{E}}\left[\sum_{i=1}^{n} \min\left(\rho_{i,t}\tilde{A}_t, \text{clip}\left(\rho_{i,t}, 1-\epsilon, 1+\epsilon\right)\tilde{A}_t\right)\right] \\ &= \hat{\mathbb{E}}\left[\sum_{i=1}^{n} \alpha_{i,t}\tilde{A}_t\right]\end{aligned} \tag{17}$$

where $\tilde{A}_t$ is the abbreviation of $\tilde{A}_t^{\boldsymbol{\pi}}$, which is the advantage estimator calculated by the team reward. Its definition is similar to (2)

$$\begin{aligned}\hat{A}_t^{\boldsymbol{\pi}} &= \hat{\delta}_t + (\gamma\lambda)\hat{\delta}_{t+1} + \cdots + (\gamma\lambda)^{T-t-1}\hat{\delta}_{T-1} \\ \text{where } \hat{\delta}_t &= R_{t+1} + \gamma\tilde{V}^{\boldsymbol{\pi}}(s_{t+1}) - \tilde{V}^{\boldsymbol{\pi}}(s_t)\end{aligned} \tag{18}$$

where the value function $\tilde{V}^{\boldsymbol{\pi}}(s_t)$ represents the cumulative discount expectation team reward under the joint policy $\boldsymbol{\pi}$

$$\tilde{V}^{\boldsymbol{\pi}}(s_t) = \hat{\mathbb{E}}_{\boldsymbol{\pi}}\left[R_{t+1} + \gamma R_{t+2} + \cdots | s_t\right]. \tag{19}$$

It is abbreviated as $\tilde{V}_t$.

If the environment only provides the team reward, then optimizing (3) is equivalent to optimizing (17) by setting $r_{i,t} = R_t, \forall i$. Although MAPPO has achieved promising results in several widely used multiagent environments based on the simplified version of (17), optimizing policies based on team rewards can encounter significant credit assignment issue in large-scale multiagent systems. Algorithms that optimize (3) based on agent-specific rewards naturally avoid the credit assignment problem. However, they still require theoretical support to establish the equivalence with optimizing (17). In this article, we provide a theoretical analysis that demonstrates the equivalence of these two optimization approaches. We conclude that algorithms utilizing agent-specific rewards can still promote cooperation if the policy can make decisions based on the global state under an *additive task* defined as follows.

Definition 1: In a cooperative multiagent task, if the decision of the overall system can be evaluated by the following

team reward:

$$R_t = \sum_{i=1}^{n} r_{i,t+1} \tag{20}$$

then the task can be called an additive task.

The additive task is very common in multiagent environments, such as StarCraft II micromanagement and predator prey. In the following, we use the two lemmas and a theorem to establish the equivalence.

Lemma 1: Given two functions $f(x)$ and $g(x)$, if the inner product of their gradients $(\partial f)/(\partial x)(x_0)$ and $(\partial g)/(\partial x)(x_0)$ at point x_0 is greater than 0, then two functions can be optimized by either of these two gradients under sufficiently small learning rates.

Proof: The condition mentioned in the lemma can be formalized as follows:

$$\frac{\partial f}{\partial x}(x_0)^T \cdot \frac{\partial g}{\partial x}(x_0) > 0. \tag{21}$$

Let $x_f = x_0 + \alpha_f(\partial f)/(\partial x)(x_0)$ and $x_g = x_0 + \alpha_g(\partial g)/(\partial x)(x_0)$, where $0 < \alpha_f$ and $\alpha_g \ll 1$. For the function $f(x)$, we can get the following derivation according to the Taylor expansion:

$$\begin{aligned} f(x_g) &= f(x_0) + \frac{\partial f}{\partial x}(x_0)^T (x_g - x_0) + \sum_{j=2}^{+\infty} o_j(x_g - x_0) \\ &= f(x_0) + \alpha_g \frac{\partial f}{\partial x}(x_0)^T \frac{\partial g}{\partial x}(x_0) + \sum_{j=2}^{+\infty} \alpha_g^j o_j\left(\frac{\partial g}{\partial x}(x_0)\right) \end{aligned} \tag{22}$$

where $o_j(\cdot)$ is the jth-order term in the Taylor expansion. To get the conclusion, we get the following derivation:

$$\begin{aligned} f(x_g) &= f(x_0) + \alpha_g \left[\frac{\partial f}{\partial x}(x_0)^T \frac{\partial g}{\partial x}(x_0) \right. \\ &\qquad \left. + \sum_{j=2}^{+\infty} \alpha_g^{j-1} o_j\left(\frac{\partial g}{\partial x}(x_0)\right) \right] \\ &= f(x_0) + \alpha_g \eta(x_0, \alpha_g) \end{aligned} \tag{23}$$

where $\eta(x_0, \alpha_g)$ is a simplified form and $\lim_{\alpha_g \to 0} \eta(x_0, \alpha_g) = (\partial f)/(\partial x)(x_0)^T (\partial g)/(\partial x)(x_0) > 0$. Therefore, for any point x_0 and threshold $\epsilon = (1/2) \cdot (\partial f)/(\partial x)(x_0)^T (\partial g)/(\partial x)(x_0)$, there exists a bound δ, such that for any $\alpha_g < \delta$, we have $|\eta(x_0, \alpha_g) - (\partial f)/(\partial x)(x_0)^T (\partial g)/(\partial x)(x_0)| \le \epsilon$. Then, we have $\eta(x_0, \alpha_g) \ge (1/2) \cdot (\partial f)/(\partial x)(x_0)^T (\partial g)/(\partial x)(x_0) > 0$, supporting the result $f(x_g) > f(x_0)$. $g(x_f) > g(x_0)$ can also be derived by a similar process. Therefore, we can get the conclusion that the two functions can be optimized by either of these two gradients under sufficiently small learning rates. □

Lemma 2: Given the same training data and the same initial network parameters, the inner product of the policy gradient obtained by optimizing (3) and (17) is greater than or equal to 0 as follows:

$$\left(\frac{\partial \mathcal{L}_{\text{tr}}}{\partial \theta}\right)^T \cdot \frac{\partial \mathcal{L}_{\text{ar}}}{\partial \theta} \ge 0. \tag{24}$$

Proof: In an additive task, the relationship between the value function for agent-specific rewards and the value function for team rewards is formulated as follows:

$$\tilde{V}_t = \sum_{i=1}^{n} V_{i,t}. \tag{25}$$

Similarly, the relationship of the advantage functions is formulated as follows:

$$\tilde{A}_t = \sum_{i=1}^{n} A_{i,t}. \tag{26}$$

Therefore, the result that the inner product of the policy gradient obtained by (3) and (17) is greater than or equal to 0 is equivalent to the following formula:

$$\begin{aligned} &\left(\frac{\partial \sum_{i=1}^{n} \alpha_{i,t} \tilde{A}_t}{\partial \theta}\right)^T \cdot \frac{\partial \sum_{i=1}^{n} \alpha_{i,t} A_{i,t}}{\partial \theta} \ge 0 \\ &\left(\frac{\partial \sum_{i=1}^{n} \alpha_{i,t} \sum_{j=1}^{n} A_{j,t}}{\partial \theta}\right)^T \cdot \frac{\partial \sum_{i=1}^{n} \alpha_{i,t} A_{i,t}}{\partial \theta} \ge 0. \end{aligned} \tag{27}$$

We take one layer of the actor network as an example and derive the second gradient as follows:

$$\frac{\partial \sum_{i=1}^{n} \alpha_{i,t} A_{i,t}}{\partial \theta} = \sum_{i=1}^{n} A_{i,t} \frac{\partial \alpha_{i,t}}{\partial \theta} = \sum_{i=1}^{n} A_{i,t} X_{i,t} \tag{28}$$

where $X_{i,t} = (\partial \alpha_{i,t})/(\partial \theta)$ is a gradient matrix used to simplify the notation. Similarly, the first gradient for the last layer can be derived as follows:

$$\frac{\partial \sum_{i=1}^{n} \alpha_{i,t} \sum_{j=1}^{n} A_{j,t}}{\partial \theta} = \sum_{i=1}^{n} X_{i,t} \sum_{j=1}^{n} A_{j,t}. \tag{29}$$

To help the subsequent derivation, we flatten the $X_{i,t}$ into a 1-D vector with the shape of $d \times 1$. Consequently, (28) can be rewritten as $X^T \vec{A}$, and (29) can be rewritten as $(\vec{e}^T \vec{A}) X^T \vec{e}$. $\vec{e}$ is an all-ones vector with the shape of $n \times 1$, $\vec{A}$ is the vector formed by $A_{i,t}$ with the shape of $n \times 1$, and X is the matrix formed by $X_{i,t}$ with the shape of $n \times d$, where d is the dimension of $X_{i,t}$. Referring back to the derivation of (27), the inner product can be expressed as follows:

$$\left(\left(\vec{e}^T \vec{A}\right) X^T \vec{e}\right)^T \cdot X^T \vec{A} = \vec{e}^T X X^T \vec{A} \vec{A}^T \vec{e} = \vec{e}^T M \vec{e} \tag{30}$$

where $M = X X^T \vec{A} \vec{A}^T$ is the matrix of a quadratic form with the shape of $n \times n$. Consequently, $\vec{e}^T M \vec{e} \ge 0$ is equivalent to that M is a positive semidefinite matrix. Next, we prove the positive semidefinite property of M through its eigenvalues as follows:

$$M\vec{v} = X X^T \vec{A} \vec{A}^T v = \lambda \vec{v} \tag{31}$$

where λ is the eigenvalue and $\vec{v}$ is the eigenvector. Since $\vec{A}^T \vec{v}$ is a scalar, the eigenvector should be $k X X^T \vec{A}$, where k is an arbitrary constant. Then, we can get the following derivations:

$$\begin{aligned} M\vec{v} &= k X X^T \vec{A}\left(\vec{A}^T X X^T \vec{A}\right) = \lambda \vec{v} = \lambda k X X^T \vec{A} \\ \lambda &= \vec{A}^T X X^T \vec{A}. \end{aligned} \tag{32}$$

Since the matrix $X X^T$ is always a positive semidefinite matrix, the conclusion $\lambda \ge 0$ is always true, meaning

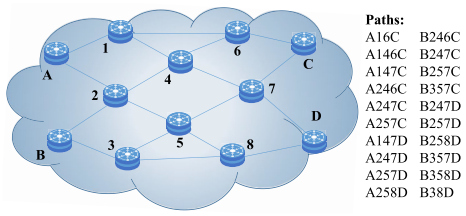

Fig. 3. Medium-scale packet routing task. It has 12 routers and 20 available paths [19]. Take router 2 as an example; it receives packets with different destinations (C or D) from Link-A2 and Link-B2. The router should send the packets to routers 4 and 5 in different proportions.

TABLE I
AVERAGE RETURNS OF PACKET ROUTING EXPERIMENTS

Method \ Task	Small	Medium	Large
MAAC† [46]	13.57	0.02	1.33
ATT-MA† [47]	13.72	15.55	17.36
NCC-AC† [19]	**13.86**	18.82	43.33
IPPO [40]	13.76	22.81	58.69
NVIF-PPO	13.79	**22.95**	**58.74**

† Results provided by the code repository implemented in [19].

that M is also a positive semidefinite matrix. Consequently, the inner product of the policy gradient obtained by optimizing (3) and (17) is greater than or equal to 0. □

Theorem 1: Given the same training data and the same initial network parameters, when the learning rate α is sufficiently small, the policy gradient obtained by (3) can optimize the objective denoted by (17) as the following denotation:

$$\mathcal{L}_{\text{tr}}\left(\theta + \alpha \frac{\partial \mathcal{L}_{\text{ar}}}{\partial \theta}\right) \geq \mathcal{L}_{\text{tr}}(\theta). \tag{33}$$

Proof: According to Lemma 2, the inner product of the policy gradient obtained by optimizing (3) and (17) is greater than or equal to 0.

1) In the case of the inner product is greater than 0, it is easy to conclude that the policy gradient obtained by (3) can also optimize the objective denoted by (17) according to the result of Lemma 1.
2) In the case of the inner product is equal to 0, the eigenvalue λ in the proof of Lemma 2 is equal to 0, meaning that $\mathcal{L}_{\text{ar}}(\theta) = X^T \vec{A}$ is a zero vector. Therefore, we can get $\mathcal{L}_{\text{tr}}(\theta + \alpha(\partial \mathcal{L}_{\text{ar}})/(\partial\theta)) = \mathcal{L}_{\text{tr}}(\theta)$. □

Our method replaces the ground-truth global state s_t by the combination of $\hat{s}_{i,t}$ and $o_{i,t}$ as an approximation in practice. Based on the above theorem, we can conclude that when working with large-scale multiagent systems, NVIF-PPO with agent-specific rewards can optimize the team reward and has the potential to promote cooperation.

IV. EXPERIMENTS ON PACKET ROUTING

We first evaluate our proposed method in the heterogeneous environment introduced in [19], which features a continuous action space. As shown in Fig. 3, in this environment, multiple routers transmit Internet packets through available paths. The path is made up of several links, each with a maximum capacity to transmit packets. Routers should determine how to split received packets to next routers, with each router having a unique action space. The routers' objectives are to cooperate and minimize the maximum link utilization in the entire network. We provide subtasks of three different scales: 4R-6P, 12R-20P, and 24R-120P, where "R" represents the controllable routers and "P" represents the paths. The neighborhood relationships of routers are predefined and fixed. More details are available in [19].

We compare NVIF-PPO with IPPO [40], MAAC [46], ATT-MADDPG [47], and NCC-AC [19]. IPPO uses local observation to make decisions and employs PPO to optimize its policy. MAAC and ATT-MADDPG utilize attention mechanism to improve learning efficiency, and they both adopt decentralized policies with centralized critics. NCC-AC has a similar structure to our proposed method, but with some differences. The authors assume that there exists a cognition consistency in the neighborhood of an agent and use a GCN module and a VAE module to extract auxiliary features for decision-making. However, they do not utilize historical information, and their communication protocol is trained online, which limits its performance in complex tasks.

Table I shows the experiment results of packet routing task with three different scales. The results of MAAC, ATT-MADDPG (simplified as ATT-MA in the table), and NCC-AC are provided by the code repository[1] implemented in [19]. The evaluation metric is the average return, which equals to the average reward summation of an episode. The results of IPPO and NVIF-PPO are averaged by repeating three experiments.

The experimental results show that the NCC-AC achieves the best performance in the small-scale task, while NVIF-PPO performs best in the medium task and the large-scale task. Since the packet routing is a simple task and it does not heavily rely on cooperation, IPPO and NVIF-PPO can obtain the best performance. The returns achieved by these two algorithms are close, because they are both approaching the optimal values. Furthermore, our approach is validated by its superior performance in comparison with other methods.

V. EXPERIMENTS ON GATHERING FOODS

A. Comparative Experiments

To evaluate our method in a more complex task, we utilize MAgent,[2] an open-source RL platform featuring a large population of agents in a grid world, to create a cooperative task called *gathering foods*. The task features controllable agents (blue blocks) working together to eliminate food units (red blocks) as quickly as possible. The observation of an agent, which contains the position and presence information of the agents and foods, is a local spatial view with the channel numbers of 7 and the observation radius of 6, and we reshape it into a 1-D tensor. The action space of an agent is a discrete space with 33 actions, which consists two types of actions: *move* and *attack*. The agent can only perform one action among these 33 actions at one timestep. The reward for attacking depends on the target: attacking food yields a

[1]https://github.com/maohangyu/marl_demo
[2]https://github.com/geek-ai/MAgent

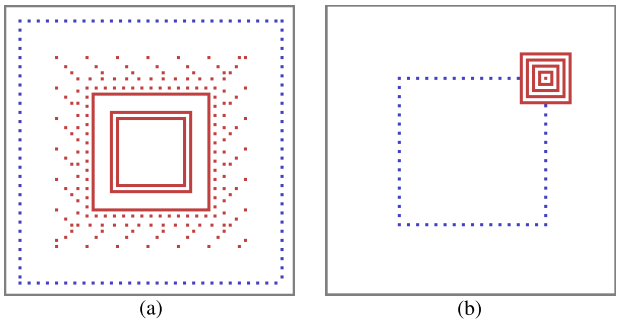

Fig. 4. Gathering foods task. (a) Fixed scenario, in which the initial position of agents and food is fixed. (b) Initial position of food is random, which requires efficient information exchange between agents.

TABLE II
DETAILED DESCRIPTIONS OF EXPERIMENTAL MAPS

Type	Scale	Map Size	Agent	Food
Normal	Small	24	27	87
	Medium	48	56	237
	Large	96	115	521
Random	Small	24	15	17
	Medium	48	29	49
	Large	96	49	161

positive reward $+1.0$, while attacking empty space incurs a small penalty -0.1. Attacking other agents earns no reward and imposes a greater penalty on the attacked agent -0.5, potentially leading to its death. A minor penalty -0.05 is also provided to encourage faster task completion.

We create two types of subtasks: *normal task* and *random task*, with identical unit attributes (e.g., hit point, observation range, attack range, and movement range) and fixed initial agent positions. As shown in Fig. 4, the normal task has fixed initial food positions, while the random task assigns initial food positions at the beginning of each episode, making it more challenging and better suitable for evaluating communication efficiency. Specifically, in the random task, the food units are organized into a square shape, with center points randomly generated in the map's four corners. We use maps of different sizes (24, 48, and 96) to evaluate the performance of our method under varying population sizes of the multiagent system. More details about the maps can be found in Table II. Besides, we design a rule-based communication graph structure for this task to ensure full connectivity, which establishes a bidirectional neighborhood relationship between an agent and its neighbors with the smallest distance in the up, down, left, and right directions.

We also provide NVIF-deep-Q network (DQN), an extension of our approach that incorporates DQN with NVIF to enable integration with other RL algorithms. We compare NVIF-DQN and NVIF-PPO with MFQ [22], DGN [33], IPPO [40], and MASIA [38]. To address the challenge of enormous interactions in large-scale multiagent systems, MFQ uses the mean-field approximation to learn the best response of each agent to the mean effect of its neighbors. DGN utilizes the GCN module to enable information exchange between agents in a multiagent system. The agents collect the features of all agents with relational kernel to alleviate information redundancy. MASIA [38] introduces two self-supervised loss functions aimed at producing information-aggregated features for agents to leverage in their decision-making processes. The codes of these algorithms are all open source.

We choose the average return under the same number of training timesteps as the evaluation metric to ensure the fairness of comparison. To help performance comparisons across maps with different sizes, we normalize the returns by the number of foods present in each map. Furthermore, we optimize the hyperparameters of the experimental algorithms separately to attain their optimal performance. Fig. 5 shows the learning curves of experimental algorithms, and Table III shows the normalized average returns when the algorithms reach convergence.

1) *Normal Tasks:* In normal tasks, the performance of agents does not heavily rely on the communication. They should pay more attention to learn how to avoid attacking or being attacked by other agents. The results presented in Fig. 5(a)–(c) demonstrate that MFQ, NVIF-DQN, and IPPO exhibit superior performance and faster convergence on the small-scale map. However, as the size of the map and the number of agents increase, their performance decreases significantly. DGN and MASIA fail to converge within the specified number of training timesteps. NVIF-PPO can achieve the best overall performance on maps across all sizes, owing to the auxiliary features provided by the pretrained communication protocol.

   We analyze the replays of NVIF-PPO to find out two typical strategies help the multiagent system achieve better performance. The first strategy *concentrating attack* is a common strategy shown in the bottom right of Fig. 6(a). To help the agents complete the task as quickly as possible, several agents move around a food unit and attack it together. The second strategy *crossing obstacles* is an advanced strategy shown in the top left of Fig. 6(a). The agent crosses the obstacle from the position denoted by the green block, representing the agent's location at the last timestep, toward the endpoint indicated by the black arrow. This enables the agent to preemptively attack internal food units, enhancing attacking efficiency and mitigating future congestion caused by the aggregation of agents.
2) *Random Tasks:* The random tasks are more challenging than the normal tasks. In these tasks, the initial positions of food units are randomized, resulting in only a subset of agents being able to observe the food units at the beginning of each episode. Therefore, they need to exchange information efficiently with each other. The results presented in Fig. 5(d)–(f) demonstrate that MFQ and NVIF-DQN can still perform well in small-scale maps, where the gap between observation range and map size is relatively small. However, as this gap expands, MFQ can hardly learn an effective agent policy. DGN, MASIA, and IPPO exhibit poor performance and slow convergence rates. Due to the efficient communication, agents trained by NVIF-PPO can achieve the best performance in larger maps.

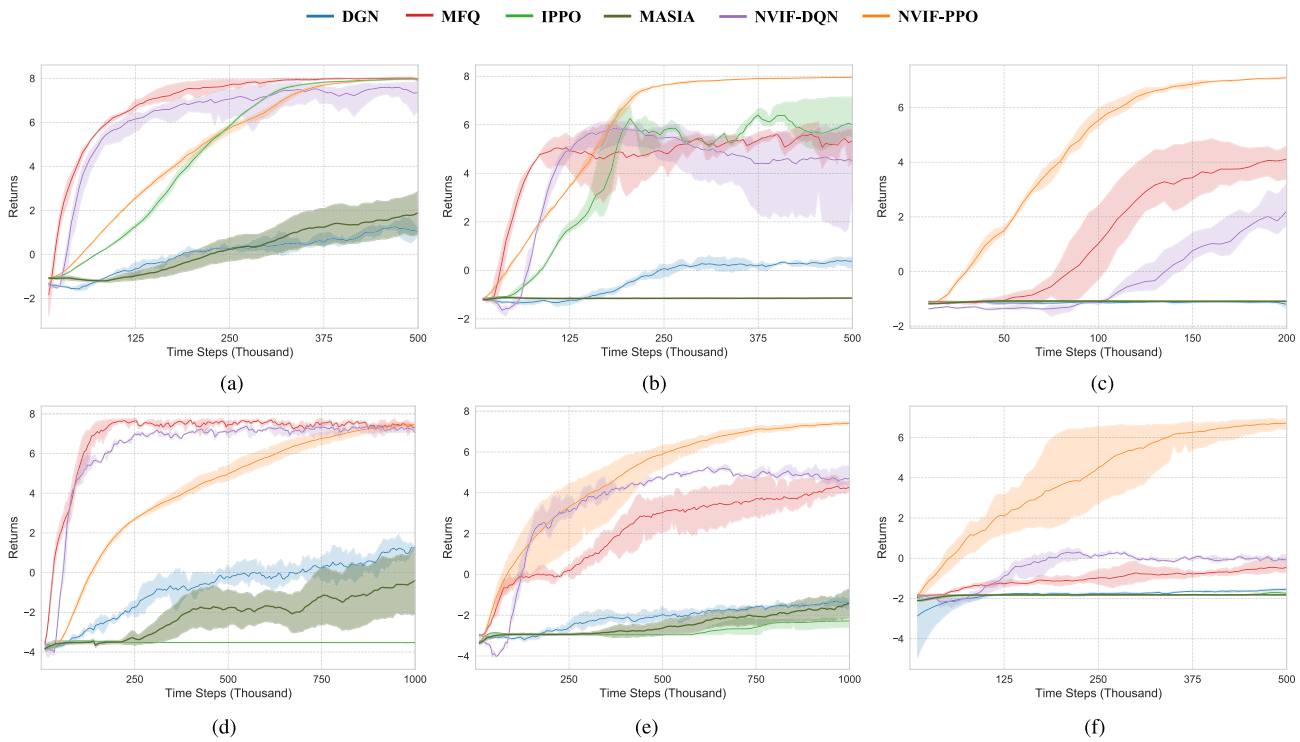

Fig. 5. Average return for DGN, MFQ, IPPO, MASIA, NVIF-DQN, and NVIF-PPO. (a)–(c) Results in small-, medium-, and large-scale normal tasks. (d)–(f) Results in small-, medium-, and large-scale random tasks. The evaluation metric is the return, which is the sum of the rewards of all agents in an episode. NVIF completes all tasks and gets the best performance. The shadow area is the confidence interval when the degree of confidence is set to 95%.

TABLE III
DETAILED RESULTS OF GATHERING FOODS EXPERIMENTS

Method	Normal Task			Random Task		
	Small	Medium	Large	Small	Medium	Large
DGN [33]	2.77 ± 0.80	0.46 ± 0.37	-0.38 ± 0.29	2.72 ± 0.92	0.03 ± 0.53	-1.36 ± 0.27
MFQ [22]	7.98 ± 0.14	6.03 ± 0.93	4.76 ± 0.95	7.26 ± 0.84	4.68 ± 0.48	-0.03 ± 0.42
IPPO [40]	**8.11 ± 0.00**	7.98 ± 0.02	0.73 ± 0.05	7.77 ± 0.01	-2.09 ± 0.04	-1.79 ± 0.00
MASIA [38]	2.07 ± 0.82	-1.15 ± 0.01	-1.41 ± 0.18	-0.26 ± 1.22	-1.77 ± 0.59	-1.63 ± 0.19
NVIF-DQN	7.30 ± 0.43	4.31 ± 0.69	3.12 ± 1.73	7.45 ± 0.25	5.08 ± 0.58	-0.14 ± 0.59
NVIF-PPO	**8.11 ± 0.00**	**8.05 ± 0.00**	**7.89 ± 0.02**	**7.85 ± 0.00**	**7.74 ± 0.02**	**7.36 ± 0.09**

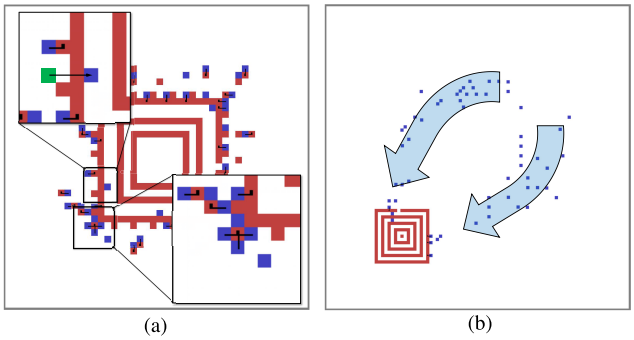

Fig. 6. Strategies learned by the agents using NVIF-PPO. There are two basic strategies *concentrating attack* and *crossing obstacles*, and one special strategy *gathering*. (a) Bottom right shows the concentrating attack and top left shows the crossing obstacles. (b) Agents gather directly to the correct position of food units.

We analyze the replays of NVIF-PPO to find out the *gathering* strategy, which is particularly vital when undertaking random tasks. Given the absence of predetermined locations for food units, agents must determine their movement based on the information exchanged with other agents. As denoted in Fig. 6(b), all agents converge on the location of the food units. Once an agent receives the information shared by other agents capable of observing the food units, it moves directly to the true location, enabling the agents to reach the correct positions efficiently to accomplish the task.

B. Ablation Experiments

In this section, we conduct ablation experiments to investigate the effect of the following: 1) the neighboring communication mechanism; 2) the latent state provided by NVIF; 3) the communication bandwidth; and 4) number of training epochs in the pretraining stage. The first two experiments are performed on the medium-scale map of the normal and random task, and the last two experiments are performed on the medium-scale map of the normal task.

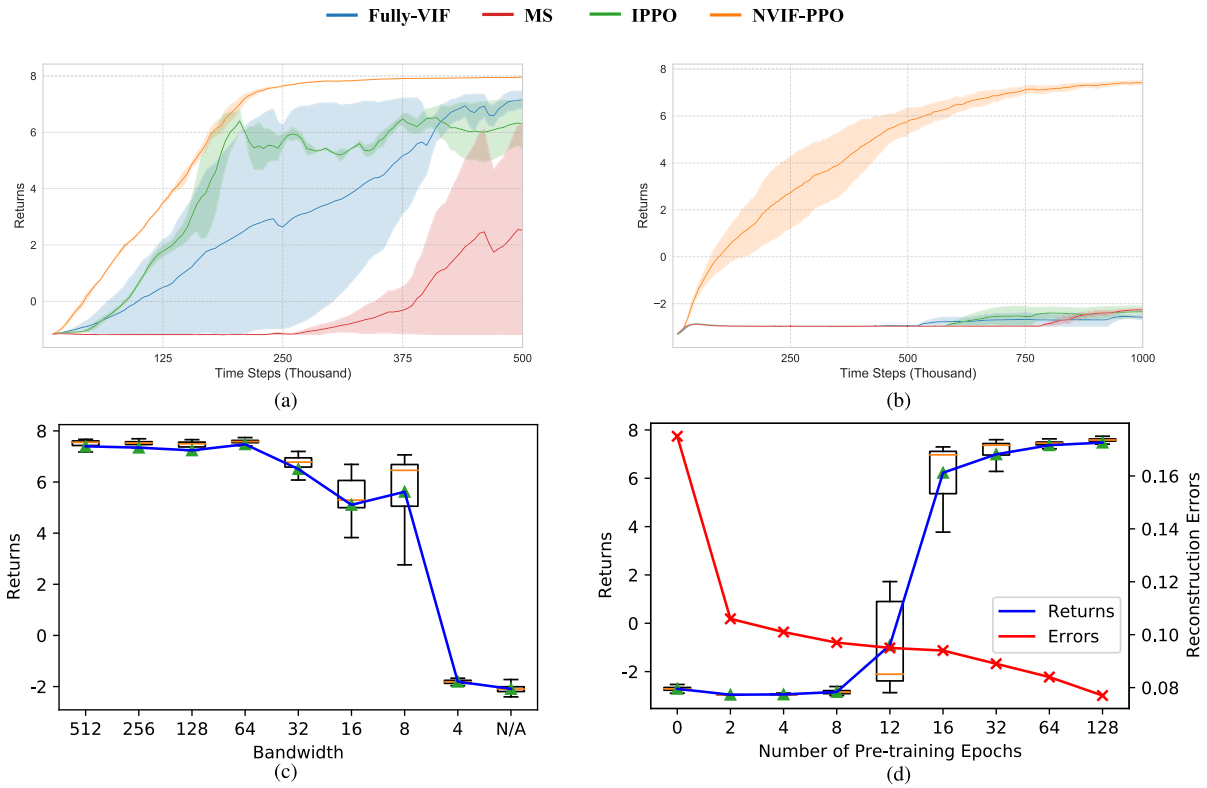

Fig. 7. Results of ablation experiments in the medium-scale map of (a) normal task and (b) random task. MS takes the mean observation as the auxiliary feature for agent decision-making to investigate the efficiency of NVIF. Fully VIF uses the fully communication instead of neighboring communication to demonstrate the impact of information redundancy. (c) Returns at varying communication bandwidths, with the IPPO algorithm denoted by N/A. (d) Relationship between pretraining epochs and both returns and reconstruction errors. The box diagrams in (c) and (d) are the result of 100 replicated experiments.

1) *Effect of Neighboring Communication:* We introduce an ablation method, called fully VIF, which provides information exchange between each pair of agents, potentially leading to increased information redundancy. As shown in Fig. 7(a) and (b), the blue curves represent the performance of fully VIF, which lies between NVIF-PPO and IPPO, highlighting the importance of neighboring communication.
2) *Effect of Latent State:* We propose an ablation method, called mean-state (MS), which uses the mean observation of all alive agents as the auxiliary feature. It is the simplest way to obtain the information of the whole system. However, the mean observation is not as efficient and effective as the latent state provided by NVIF. As shown in Fig. 7(a) and (b), the green curves represent the performance of MS. Given that the mean observation fails to provide comprehensive information, it adversely impacts the training process and leads to a slower convergence rate.
3) *Effect of Communication Bandwidth:* In the FlowNet's architecture depicted in Fig. 1(c), the first neural network extracts local features from the inputs and shares them with neighboring agents. Consequently, the communication bandwidth of our method is defined as the hidden state dimension of the first neural network in FlowNet, rather than its original definition related to communication frequency. We reduce the bandwidth from 512 to 0 (N/A, i.e., IPPO), as illustrated in Fig. 7(c). The results show that our method can achieve satisfactory performance when the bandwidth is greater than 4, indicating that NVIF-PPO can handle scenarios with limited communication bandwidth.
4) *Effect of Pretraining Epochs:* The pretraining stage provides the communication protocol the capability to efficiently represent MIS. Inadequate pretraining can impair this capability. As shown in Fig. 7(d), we increase the pretraining epochs from 0 to 128 and evaluate the communication protocol by conducting the online policy training process. The results show that satisfactory policy performance is achieved only after 32 pretraining epochs, despite rapid decline in reconstruction error during the early pretraining phase.

C. Scalability Experiments

The pretrained NVIF module is task-agnostic, enabling the communication protocol to facilitate scaling of agent policies to unseen scenarios and achieve satisfactory performance. We evaluate the effectiveness of our approach by testing the trained policies on previously unseen maps. The scalability scores presented in Fig. 8 show the average returns normalized by the maximum return for each testing task. The score at row i and column j indicates that the policy trained on map i is tested on map j. Diagonal elements represent the performance on seen scenarios, while off-diagonal elements represent the performance on unseen scenarios. We compare NVIF-PPO with MFQ, which outperforms other experimental algorithms.

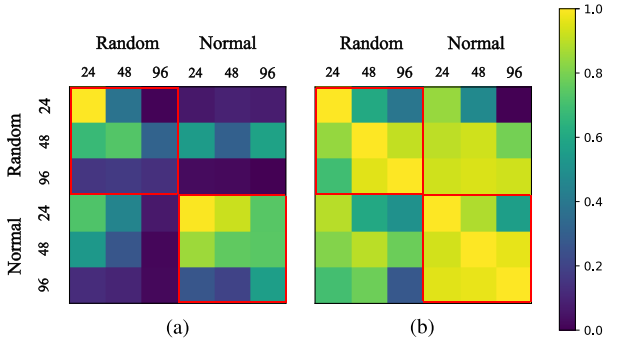

Fig. 8. Scalability score matrix of (a) MFQ and (b) NVIF. Scores represent the scalability performance of agents trained in one map when running in another map. The red rectangles indicate the results within the same type of task.

As shown in Fig. 8, NVIF-PPO is capable of achieving good performance not only in the same task with different map sizes but also in different tasks. In contrast, MFQ only exhibits scalability in normal tasks. In small-scale maps, agents tend to rely less on communication and cooperation to complete tasks, resulting in policies that are less scalable to larger maps. In addition, policies trained in larger maps are more prone to mistakenly attacking other agents due to the increased crowding on smaller maps. The scalability of the policy is also dependent on the type of task. Policies trained in normal tasks are less likely to use communication to promote cooperation, resulting in difficulty scaling to random tasks.

VI. Conclusion

This article proposes a novel communication-based MARL method to improve communication efficiency in large-scale multiagent systems. The proposed method utilizes neighboring communication and provides agents with more information by providing MIS, which is compressed into a compact latent state to further mitigate the information redundancy by the proposed NVIF module. To stabilize the overall training process, a two-stage training mechanism is presented. In the first stage, the NVIF module is pretrained offline to serve as a stable communication protocol that enables agents to enrich their local observations. In the second stage, the MARL algorithm and the pretrained protocol are used for online policy training to obtain agent policies and promote cooperation. Moreover, we provide a theoretical analysis for the potential of our proposed NVIF-PPO method in promoting cooperation with agent-specific rewards. We compare our method with state-of-the-art methods in two tasks that cover diverse multiagent scenarios. Experiment results demonstrate that the NVIF outperforms other methods in the communication efficiency and cooperative performance. We also conduct ablation experiment and scalability experiment to demonstrate the effect of our proposed method. To further enhance the communication efficiency, future work can investigate the development of an efficient communication protocol and graph structure from the offline dataset. Furthermore, the improved training efficiency achieved in this work can potentially support the practical deployment of multiagent systems in real-world settings.

References

[1] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 427–438, Feb. 2013.

[2] W. Jiang, G. Feng, S. Qin, T. S. P. Yum, and G. Cao, "Multi-agent reinforcement learning for efficient content caching in mobile D2D networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 3, pp. 1610–1622, Mar. 2019.

[3] Y. Zhu, H. He, and D. Zhao, "LMI-based synthesis of string-stable controller for cooperative adaptive cruise control," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 11, pp. 4516–4525, Nov. 2020.

[4] Y. Zhu, D. Zhao, X. Li, and D. Wang, "Control-limited adaptive dynamic programming for multi-battery energy storage systems," *IEEE Trans. Smart Grid*, vol. 10, no. 4, pp. 4235–4244, Jul. 2019.

[5] K. Shao, Y. Zhu, and D. Zhao, "StarCraft micromanagement with reinforcement learning and curriculum transfer learning," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 3, no. 1, pp. 73–84, Feb. 2019.

[6] J. Chai et al., "UNMAS: Multiagent reinforcement learning for unshaped cooperative scenarios," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 4, pp. 2093–2104, Apr. 2023.

[7] Z. Tang, K. Shao, Y. Zhu, D. Li, D. Zhao, and T. Huang, "A review of computational intelligence for StarCraft AI," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Nov. 2018, pp. 1167–1173.

[8] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4295–4304.

[9] Z. Ding, T. Huang, and Z. Lu, "Learning individually inferred communication for multi-agent cooperation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 22069–22079.

[10] D. V. Dimarogonas, E. Frazzoli, and K. H. Johansson, "Distributed event-triggered control for multi-agent systems," *IEEE Trans. Autom. Control*, vol. 57, no. 5, pp. 1291–1297, May 2012.

[11] Y. Zhu, D. Zhao, H. He, and J. Ji, "Event-triggered optimal control for partially unknown constrained-input systems via adaptive dynamic programming," *IEEE Trans. Ind. Electron.*, vol. 64, no. 5, pp. 4101–4109, May 2017.

[12] G. Hu, Y. Zhu, D. Zhao, M. Zhao, and J. Hao, "Event-triggered communication network with limited-bandwidth constraint for multi-agent reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 8, pp. 3966–3978, Aug. 2023.

[13] D. Kim et al., "Learning to schedule communication in multi-agent reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–17.

[14] A. Singh, T. Jain, and S. Sukhbaatar, "Learning when to communicate at scale in multiagent cooperative and competitive tasks," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–16.

[15] Z. Chen, Z. Zhu, G. Yang, and Y. Gao, "HiSA: Facilitating efficient multi-agent coordination and cooperation by hierarchical policy with shared attention," in *Proc. Pacific Rim Int. Conf. Artif. Intell.* Cham, Switzerland: Springer, 2022, pp. 77–90.

[16] Y. Niu, R. Paleja, and M. Gombolay, "Multi-agent graph-attention communication and teaming," in *Proc. 20th Int. Conf. Auto. Agents MultiAgent Syst.*, 2021, pp. 964–973.

[17] C. Zhao, J. Zhao, Z. Du, and K. Lu, "ACUTE: Attentional communication framework for multi-agent reinforcement learning in partially communicable scenarios," *Electronics*, vol. 11, no. 24, p. 4204, Dec. 2022.

[18] T. Luo, B. Subagdja, D. Wang, and A.-H. Tan, "Multi-agent collaborative exploration through graph-based deep reinforcement learning," in *Proc. IEEE Int. Conf. Agents (ICA)*, Oct. 2019, pp. 2–7.

[19] H. Mao et al., "Neighborhood cognition consistent multi-agent reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 5, pp. 7219–7226.

[20] J. Sheng et al., "Learning structured communication for multi-agent reinforcement learning," *Auto. Agents Multi-Agent Syst.*, vol. 36, no. 2, p. 50, Oct. 2022.

[21] N. Gupta, G. Srinivasaraghavan, S. K. Mohalik, N. Kumar, and M. E. Taylor, "HAMMER: Multi-level coordination of reinforcement learning agents via learned messaging," 2021, *arXiv:2102.00824*.

[22] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5571–5580.

[23] Z. Zhou and H. Xu, "Large-scale multiagent system tracking control using mean field games," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 10, pp. 5602–5610, Oct. 2022.

[24] Y. Zhu and D. Zhao, "Online minimax Q network learning for two-player zero-sum Markov games," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 3, pp. 1228–1241, Mar. 2022.
[25] J. Chai, W. Chen, Y. Zhu, Z.-X. Yao, and D. Zhao, "A hierarchical deep reinforcement learning framework for 6-DOF UCAV air-to-air combat," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 53, no. 9, pp. 5417–5429, Sep. 2023.
[26] Z. Tang, Y. Zhu, D. Zhao, and S. M. Lucas, "Enhanced rolling horizon evolution algorithm with opponent model learning: Results for the fighting game AI competition," *IEEE Trans. Games*, vol. 15, no. 1, pp. 5–15, Mar. 2023.
[27] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Basar, "Fully decentralized multi-agent reinforcement learning with networked agents," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5872–5881.
[28] K. Lin, R. Zhao, Z. Xu, and J. Zhou, "Efficient large-scale fleet management via multi-agent deep reinforcement learning," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 1774–1783.
[29] Y. Liu, W. Wang, Y. Hu, J. Hao, X. Chen, and Y. Gao, "Multi-agent game abstraction via graph attention neural network," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 7211–7218.
[30] J. Foerster, Y. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2137–2145.
[31] S. Sukhbaatar et al., "Learning multiagent communication with backpropagation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 2244–2252.
[32] Q. Zhang, D. Zhao, and D. Wang, "Event-based robust control for uncertain nonlinear systems using adaptive dynamic programming," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 1, pp. 37–50, Jan. 2018.
[33] J. Jiang, C. Dun, T. Huang, and Z. Lu, "Graph convolutional reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–13.
[34] S. Ding, W. Du, L. Ding, J. Zhang, L. Guo, and B. An, "Multiagent reinforcement learning with graphical mutual information maximization," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Feb. 16, 2023, doi: 10.1109/TNNLS.2023.3243557.
[35] J. Jiang and Z. Lu, "Learning attentional communication for multi-agent cooperation," in *Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–11.
[36] J. Su, S. Adams, and P. A. Beling, "Counterfactual multi-agent reinforcement learning with graph convolution communication," 2020, *arXiv:2004.00470*.
[37] Q. Yuan, X. Fu, Z. Li, G. Luo, J. Li, and F. Yang, "GraphComm: Efficient graph convolutional communication for multiagent cooperation," *IEEE Internet Things J.*, vol. 8, no. 22, pp. 16359–16369, Nov. 2021.
[38] C. Guan et al., "Efficient multi-agent communication via self-supervised information aggregation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 1020–1033.
[39] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[40] C. S. de Witt et al., "Is independent learning all you need in the StarCraft multi-agent challenge?" 2020, *arXiv:2011.09533*.
[41] C. Yu et al., "The surprising effectiveness of PPO in cooperative multi-agent games," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 24611–24624.
[42] Z. Wu, C. Yu, D. Ye, J. Zhang, and H. H. Zhuo, "Coordinated proximal policy optimization," in *Proc. 35th Conf. Neural Inf. Process. Syst.*, 2021, pp. 26437–26448.
[43] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2015, *arXiv:1506.02438*.
[44] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–14.
[45] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *Proc. NIPS Workshop Deep Learn.*, 2014, pp. 1–9.
[46] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2961–2970.
[47] H. Mao, Z. Zhang, Z. Xiao, and Z. Gong, "Modelling the dynamic joint policy of teammates with attention multi-agent DDPG," in *Proc. 18th Int. Conf. Auto. Agents MultiAgent Syst.*, 2019, pp. 1108–1116.

Jiajun Chai (Graduate Student Member, IEEE) received the B.S. degree from the Faculty of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, China, in 2020. He is currently pursuing the Ph.D. degree with the State Key Laboratory of Multimodal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China.

His current research interests include multiagent reinforcement learning, deep learning, and game artificial intelligence (AI).

Yuanheng Zhu (Senior Member, IEEE) received the B.S. degree in automation from Nanjing University, Nanjing, China, in 2010, and the Ph.D. degree in control theory and control engineering from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2015.

From 2015 to 2017, he was an Assistant Professor with the Institute of Automation, Chinese Academy of Sciences, where he is currently an Associate Professor. From 2017 to 2018, he was a Visiting Scholar with the Department of Electrical, Computer, and Biomedical Engineering, University of Rhode Island, Kingston, RI, USA. His current research interests include deep reinforcement learning, game theory, game intelligence, and multiagent learning.

Dr. Zhu serves as an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS.

Dongbin Zhao (Fellow, IEEE) received the Ph.D. degree from the Harbin Institute of Technology, Harbin, China, in 2000.

He is currently a Professor with the Institute of Automation, Chinese Academy of Sciences, Beijing, China, and also a Professor with the University of Chinese Academy of Sciences, Beijing. He has published seven books and over 100 international journal articles. His current research interests include deep reinforcement learning, computational intelligence, autonomous driving, game artificial intelligence, and robotics.

Dr. Zhao serves as an Associate Editor for the IEEE TRANSACTIONS ON CYBERNETICS, the IEEE TRANSACTIONS ON ARTIFICIAL INTELLIGENCE, and *IEEE Computation Intelligence Magazine*.