Multiagent Reinforcement Learning With Heterogeneous Graph Attention Network

Wei Du, Shifei Ding, Chenglong Zhang, and Zhongzhi Shi, *Life Senior Member, IEEE*

***Abstract*— Most recent research on multiagent reinforcement learning (MARL) has explored how to deploy cooperative policies for homogeneous agents. However, realistic multiagent environments may contain heterogeneous agents that have different attributes or tasks. The heterogeneity of the agents and the diversity of relationships cause the learning of policy excessively tough. To tackle this difficulty, we present a novel method that employs a heterogeneous graph attention network to model the relationships between heterogeneous agents. The proposed method can generate an integrated feature representation for each agent by hierarchically aggregating latent feature information of neighbor agents, with the importance of the agent level and the relationship level being entirely considered. The method is agnostic to specific MARL methods and can be flexibly integrated with diverse value decomposition methods. We conduct experiments in predator–prey and StarCraft Multiagent Challenge (SMAC) environments, and the empirical results demonstrate that the performance of our method is superior to existing methods in several heterogeneous scenarios.**

***Index Terms*— Graph attention network, heterogeneous agents, multiagent reinforcement learning (MARL), relationship-level attention.**

NOMENCLATURE

$\Re$	Relationship.
$o_i \in O$	Observation of agent i.
h	Feature representation of an agent.
δ	Type of agent.
M_δ	Transformation matrix.
$Z^\Re$	Relationship-specific feature representation.
$\beta^\Re$	Weight of relationship $\Re$.
$\alpha_{ij}^\Re$	Weight of relationship-based agent pair (i, j).
$N_i^\Re$	Relationship-based neighbors of agent i.
Z_i	Final feature representation of agent i.

Manuscript received 25 October 2021; revised 6 July 2022; accepted 14 October 2022. Date of publication 4 November 2022; date of current version 6 October 2023. This work was supported by the National Natural Science Foundation of China under Grant 61976216 and Grant 62276265. *(Corresponding author: Shifei Ding.)*

Wei Du, Shifei Ding, and Chenglong Zhang are with the School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, China, and also with the Mine Digitization Engineering Research Center of Ministry of Education (China University of Mining and Technology), Xuzhou 221116, China (e-mail: 1394471165@qq.com; dingsf@cumt.edu.cn; zhangchl1992@qq.com).

Zhongzhi Shi is with the Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China (e-mail: shizz@ict.ac.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3215774.

Digital Object Identifier 10.1109/TNNLS.2022.3215774

I. INTRODUCTION

MULTIAGENT reinforcement learning (MARL) has emerged as an extremely active research field and has shown excellent success in multiagent systems [1], [2]. In recent years, with the progress of deep neural networks, deep reinforcement learning has been comprehensively studied and accomplished beyond human-level performances in challenging scenarios [3]. Meanwhile, the integration of deep learning and MARL to tackle complex and large-scale problems has also aroused considerable interest. Most existing deep MARL methods leverage the centralized training with decentralized execution (CTDE) paradigm and concentrated on deploying decentralized policies for multiple agents by a centralized critic network. MARL methods leveraging the CTDE paradigm can be categorized as the "learning for consensus method" and the "learning to communicate method" based on whether the agent is able to obtain information in the execution stage. In the former method, each agent selects decentralized actions only based on its local observation [4], [5], [6], [7], while the latter method utilizes communication schemes in the execution stage [8], [9], [10], [11], [12].

Despite the abundance of deep MARL methods, there still remain some unsolved challenges. The first is the scalability issue of the MARL method. Because of the curse of dimensionality that can exist in modeling centralized critic networks, the MARL method has constraints when simulating agents of large scale. Besides, most methods concentrate on deploying cooperative policies for homogeneity agents and pay insufficient attention to large-scale heterogeneous agents. The graph neural network (GNN) is a graph representation approach that has attracted extensive attention recently. The integration of MARL and GNN forms a new paradigm to tackle the scalability challenge of the large-scale multiagent system, in which agents leverage the graph convolution layer to learn to communicate [13], [14]. Nevertheless, most of the work in this direction has been on the study of homogeneous agents [15], [16], [17], and little effort has been made to tackle the heterogeneous agent environments by constructing GNNs. In fact, the heterogeneity of the agents and the diversity of relationships causes the learning of policy excessively difficult in large-scale heterogeneous multiagent scenarios.

In this article, to tackle these challenges, we present a method that constructs a heterogeneous graph attention network for the heterogeneous multiagent environment. The proposed method leverages attention mechanisms, including the agent-level attention module and the relationship-level

attention module, to make the agent learn better strategies and coordination in heterogeneous scenarios. Besides, in many multiagent systems, taking all other agents into consideration may be costly and less helpful. Therefore, the proposed method only considers neighboring agents, which is efficient and effective. In the heterogeneous setting, each agent has neighbors of different types and, therefore, has different relationships with the neighbors. Given a specific relationship, each agent has various relationship-based neighbors. Our method attempts to distinguish the differences between these neighboring agents and focuses on the more important neighboring agents at the current state. For each agent, the agent-level attention module explores to obtain the importance of relationship-based neighbor agents and distributes them with diverse attention values. Besides, different types of neighbor agents are of different importance to an agent, and the relationship-level attention module concentrates on determining the importance of diverse relationships and deploying their appropriate weights. The primary contributions and novelties of our method are summarized as follows.

1) The proposed method is the first attempt to investigate the heterogeneous multiagent issues leveraging MARL with the heterogeneous graph attention network.
2) We present a novel heterogeneous graph attention network for MARL, which can aggregate the feature information of neighbor agents for policy learning.
3) The method is agnostic to specific MARL methods and can be flexibly integrated with diverse value decomposition methods.

The remainder of this article is constructed as follows. The related work is presented in Section II. In Section III, we present the proposed multiagent reinforcement learning with the heterogeneous graph attention network (MA-HA), including the method structure and learning procedures. In Section IV, the results of experiments on several heterogeneous settings demonstrate the superiority and feasibility of the MA-HA. Finally, the conclusion and future directions are given in Section V.

II. Related Work

A. Multiagent Reinforcement Learning With Graph Neural Network

In recent years, as a solution to the scalability problem of MARL, the GNN has been leveraged to model large-scale multiagent settings. Iqbal et al. [6] leverages the GNN to construct the centralized critic and deploy decentralized actors by employing the soft actor–critic method. Malysheva et al. [15] presents a graph generation layer to construct an adjacency matrix among agents. Jiang et al. [16] leverages a graph convolutional network to construct the multiagent system for inter-agent communication. Wang et al. [18] considers the issue of multiple types of agents, but it can only deal with fixed-size graphs. Sheng et al. [19] introduces an effective communication mechanism by leveraging the hierarchical GNN to propagate information among agents and groups.

More recently, Niu et al. [20] designs an attentional and scalable communication mechanism to decide how to process messages and when to communicate by graph attention networks. Bohmer et al. [21] leverages coordination graphs to coordinate the actions of agents by passing messages on graphs. Ryu et al. [22] applies a hierarchical attention mechanism to GNNs, which adequately represents the relationships among groups of agents. Liu et al. [23] models the relationship between agents by a novel two-stage attention graph network, which can learn the importance of the interaction between two agents. Shen et al. [24] uses GNNs to model the relationship among agents to assist the communication learning of agents.

Although previous "MARL with GNN" work has successfully modeled the interactions or relationships between multiple agents, the work mentioned above is not designed to tackle heterogeneous scenarios, where agents have different tasks or attributes. Besides, the proposed method can deal with unfixed-size graphs. Our proposed method leverages the heterogeneous graph attention network to model the heterogeneous agents and diverse relationships, which shows feasibility and superiority in heterogeneous agent scenarios.

B. Dec-POMDP

The MARL problems can be modeled as the decentralized partially observable Markov decision process (Dec-POMDP) [25], [26]. Concretely, a Dec-POMDP is represented by a tuple $G = \langle S, A, P, R, O, n, \gamma \rangle$, where the state of the partially observable environment is denoted as $s \in S$, and the local observation that agent i can obtain is represented as $o_i \in O$. The agent i determines its action $a_i \in A$ according to its local observation o_i. The joint action $\vec{a}$=$(a_1, \ldots, a_n)$ produces the next state based on the state transition function $P : S \times A \to S$. The target of the agent i is to maximize its cumulative discounted reward $R_i = \Sigma_{t=0}^{T} \gamma^t r_i^t$, where $\gamma \in [0, 1]$ denotes a discount factor. The objective is to learn a joint policy $\vec{\pi}(\vec{\tau}, \vec{a})$ to maximize the global value $Q_{\text{tot}}^{\vec{\pi}}(\vec{\tau}, \vec{a}) = \mathbb{E}_{s,\vec{a}}[\sum_{t=0}^{\infty} \gamma^t R(s, \vec{a}) | s_0 = s, \vec{a}_0 = \vec{a}]$. Here, τ_i denotes the observation history of agent i.

C. Graph Attention Network

The graph attention network is a kind of GNN that is able to efficiently cope with graph structured data. The graph attention network learns to generate the graph-embedding vector $h_i' = \sigma(\Sigma_{j \in N_i} \alpha_{ij} W h_j)$ of node i by integrating embedding h_j from neighboring nodes $\{j \in N_i\}$ that are connected to the node i. Here, $\alpha_{ij} = \text{softmax}_j(e_{ij})$ represents the attention weight, which determines the importance of node j to node i, where $e_{ij} = a(Wh_i, Wh_j)$ determines the importance of node j to node i by measuring node-embedding value h_i'.

III. Proposed Method

In this section, we model the heterogeneous multiagent setting as a heterogeneous graph, in which different types of agents are represented by the different types of nodes of the heterogeneous graph.

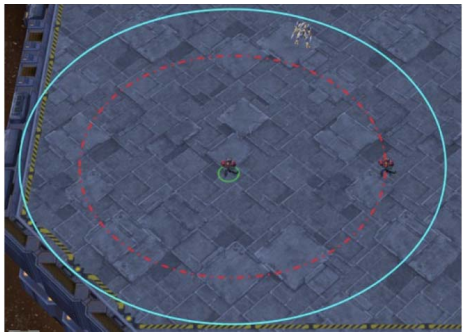

Fig. 1. Illustrative example of the field of view of the agent. The cyan circles border the view range of the agent from [32].

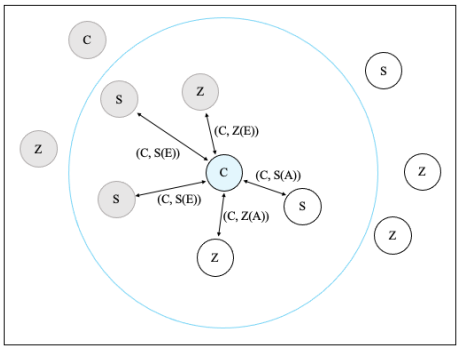

Fig. 2. Illustrative example of a heterogeneous graph based on neighbors of the agent in the 1c3s5z scenario. C, S, and Z represent Colossus, Stalkers, and Zealots.

A. Problem Formulation

We represent a heterogeneous agent graph as (V, E), which consists of the set of nodes represented as V and the set of links represented as E. The heterogeneous graph (V, E) contains a node type mapping function $\varphi : V \rightarrow A$ and a link type mapping function $\psi : E \rightarrow \Re$. A and $\Re$ represent the sets of agent types and link types, where $|A| + |\Re| \geq 3$. Each node represents an agent, and the link type represents the relationship of pairs of agents. The different link type between any two classes of agents is regarded as a different relationship.

In large-scale heterogeneous multiagent scenarios, taking all other agents into consideration may be costly and less helpful. This is because receiving the information of all agents requires high bandwidth and causes high computational complexity, and agents cannot distinguish valuable information from a large amount of globally shared information. Since convolution operation can gradually expand the receptive domain of the agent, the scope of collaboration is not limited. Therefore, it is efficient and effective to only consider neighboring agents, which can share information with each other. The logic and intuition behind this are that neighboring agents are more likely to interact and influence each other.

In our proposed method, neighbors are determined by the field of view of the agent or other metrics, which depends on the scenario. Note that different heterogeneous scenarios can have different definitions of neighbors. Take the StarCraft Multiagent Challenge (SMAC) environment as an example, as shown in Fig. 1; at each timestep, the agent obtains local observation within its field of view, and the neighbors are defined as the allied units and enemy units within the view. The agent can only exchange information with the allied neighbor units because the enemy agent is controlled by the built-in AI.

To be more intuitive, as shown in Fig. 2, we model the 1c3s5z scenario of SMAC as a heterogeneous graph. It contains six types of agents, including Colossus, Stalkers, Zealots of Ally unit, and Enemy. The blue node C in the center represents one Colossus, and the cyan circles border the sight range of the Colossus. Nodes S and Z in the cyan circle represent Stalkers and Zealots, which are the neighbor agents of node C. Gray nodes represent enemy agents, while white nodes represent allied agents.

For example, agent C has different relations with different kinds of neighbor agents. Agent C has a relationship [C, Z(E)] with enemy agent Z and a relationship [C, S(A)] with allied agent S. Meanwhile, there may also be different neighbor agents in the same relationship, such as the relationship [C, S(E)]. Concretely, given a node or agent i, it has a set of neighbors $N_i^{\Re}$. Given the relationship type $\Re$ in the modeled heterogeneous graph, the relationship-based neighbors $N_i^{\Re}$ of agent i are defined as the set of neighbor agents that connect with agent i through relationship $\Re$. It is worth noting that agent C can only communicate with allied agents; however, it can pass the relationship information about enemy agents to the neighbor allied agents.

B. MA-HA Network Structure

In this section, we present the structure of our proposed method MA-HA and further provide a comprehensive introduction. MA-HA can be integrated with any value decomposition method. MA-HA mainly consists of six modules: observation encoder, project layer, agent-level attention module, relationship-level attention module, the Q network, and the mixing network, as shown in Fig. 3.

First, we leverage the encoder module and project layer to preprocess the feature. At each timestep t, agent i acquires a local observation o_i^t. Then, the local observation o_i^t is input to the encoder module, which can consist of the MLP/LSTM layers and output a feature vector h_i^t. h_i^t is a low-level vector and contains only the local observation of agent o_i^t. Next, we leverage the project layer to transform diverse kinds of feature vectors into uniform feature space and generate the latent feature h_i^t.

Second, the heterogeneous graph attention network is used to aggregate the feature of neighbor agent. The latent features go through two attention modules: agent-level attention and relationship-level attention. Note agent i is restricted to only communicating with its neighbors. That is, by agent-level attention module, agent i is able to acquire the importance of its neighbor agents based on a specific relationship and aggregates representations of these meaningful neighbors to form feature representation $Z_i^{\Re}$. Here, we assume that there is a specific relationship between any two types of agents. We will further explain the rationality of this hypothesis in the experimental section. The relationship-level attention

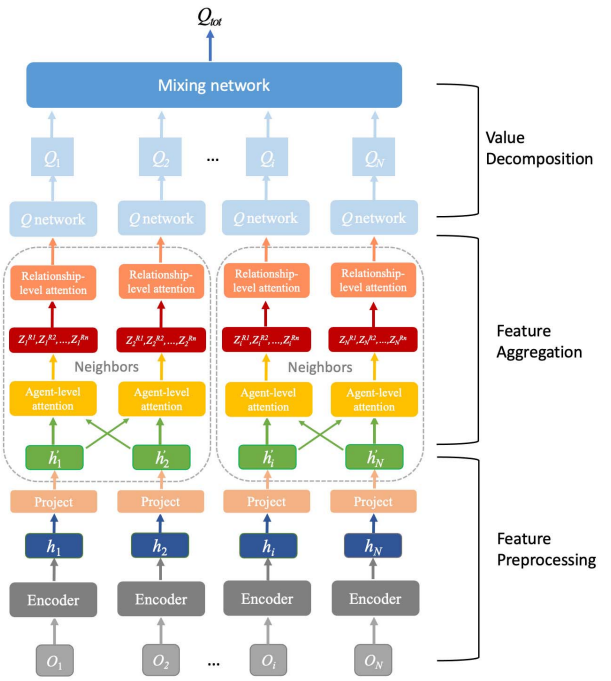

Fig. 3. MA-HA network structure that consists of feature preprocessing, feature aggregation, and value decomposition components.

module determines the importance of diverse relationships between agents and integrates them to new feature representation. At each timestep, a heterogeneous graph attention layer corresponds to one round of feature aggregation of neighbor agents. By stacking several layers, we can obtain the final feature representation Z_i.

Third, we input the final feature representation Z_i to individual decentralized Q network and output individual Q values $Q_i(o_i, a_i, Z_i)$; then, we leverage the mixing network to obtain the global Q values Q_{tot}. As the MA-HA framework leverages the CTDE paradigm, all parameters of networks can be updated by the standard TD loss of the global Q values, Q_{tot}, which are the output of any mixing network such as Q mixing network (QMIX)

$$\mathcal{L}_{\text{TD}}(\theta) E_{(\vec{\tau},\vec{a},r,\vec{\tau}')\mathcal{D}}\left[(y - Q_{\text{tot}}(\vec{\tau},\vec{a};\theta))^2\right] \tag{1}$$

where $y = r + \max_{\vec{a}'} Q_{\text{tot}}(\vec{\tau}', \vec{a}'; \theta^-)$ is the target and θ^- denotes the parameters of the target network. Under the CTDE paradigm, the mixing network will be removed during the execution phase. Furthermore, details of the attention module will be described in next.

C. Agent-Level Attention Module

It is noted that different relationship-based neighbors of each agent influence the learning feature representation of a certain task differently. We propose the agent-level attention module, which is able to acquire the importance of each agent's neighbor agents based on a specific relationship and integrates representations of these meaningful neighbors to form feature representation. Different types of agents have different feature spaces due to the heterogeneity of agents. Consequently, for each type of agents (such as agent of type δ_i), we construct a transformation matrix of a specific type M_{δ_i} to transform the features of agents of diverse types into a uniform feature space. The projection procedure is given as follows:

$$h'_i = M_{\delta i}.h_i. \tag{2}$$

The original and projected features of agent i are represented by h_i and h'_i, respectively. The agent-level attention can deal with different types of agents through a type-specific projection function. Next, we employ the self-attention mechanism to learn the weight between agents of different kinds. Given an agent pair (i, j) that is connected through relationship $\Re$, the agent-level attention module is able to obtain the importance $e_{ij}^{\Re}$ that represents the importance of agent j to agent i. We define the importance of relationship-based agent pair (i, j) leveraging attention equation as follows:

$$e_{ij}^{\Re} = \text{attention}_{\text{agent}}(h'_i; h'_j; \Re). \tag{3}$$

The DNN layer that executes the attention mechanism of the agent level is represented as $\text{attention}_{\text{agent}}$. Given the relationship $\Re$, all relationship-based agent pairs share $\text{attention}_{\text{agent}}$. The above function illustrates that, given the relationship $\Re$, the weight of the relationship-based agent pair (i, j) is determined by their features. It is worth noting that $e_{ij}^{\Re}$ is asymmetric, which illustrates the importance of agent i to agent j, and that of agent j to agent i could be different. After receiving the importance $e_{ij}^{\Re}$ between relationship-based agents, it is normalized to obtain the weight coefficient $\alpha_{ij}^{\Re}$ through the softmax operation

$$\alpha_{ij}^{\Re} = \text{softmax}_j\left(e_{ij}^{\Re}\right) = \frac{\exp\left(\sigma\left(a_{\Re}^T \cdot \left[h'_i \| h'_j\right]\right)\right)}{\sum_{k \in N_i^{\Re}} \exp\left(\sigma\left(a_{\Re}^T \cdot \left[h'_i \| h'_k\right]\right)\right)}. \tag{4}$$

The activation function is represented as σ, and the concatenate operation is denoted as $\|$. Consequently, the relationship-based feature representation of agent i can be concentrated by the latent features of neighbor agents according to the corresponding coefficients as follows:

$$z_i^{\Re} = \sigma\left(\sum\nolimits_{j \in N_i^{\Re}} \alpha_{ij}^{\Re} \cdot h'_j\right). \tag{5}$$

The feature representation of agent i for the relationship $\Re$ is denoted as $z_i^{\Re}$. To explain the aggregating procedure of agent-level attention more intuitively, we present a concise explanation in Fig. 4. Every feature representation of agent is integrated by its neighbor agents. The attention weight $\alpha_{ij}^{\Re}$ is generated for a single relationship $\Re$ for agent i; it is relationship-specific and can maintain one type of relationship information. The agent-level attention module is extended to multihead attention to perform the learning procedure more robustly. Concretely, the agent-level attention is repeated for K times, and the feature representation is concatenated as the relationship-specific feature

$$z_i^{\Re} = \Big\|_{k=1}^{K} \sigma\left(\sum\nolimits_{j \in N_i^{\Re}} \alpha_{ij}^{\Re} \cdot h'_j\right). \tag{6}$$

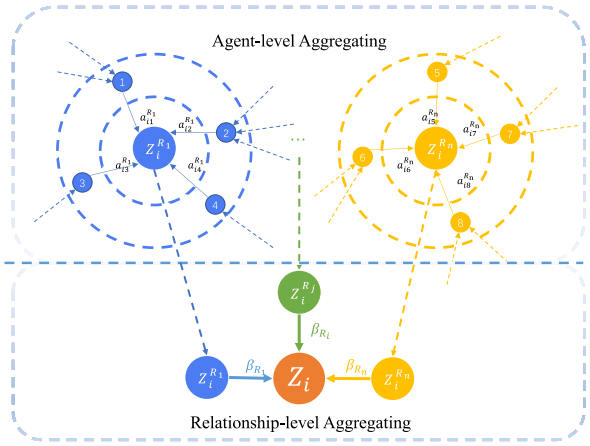

Fig. 4. Illustration of the aggregating procedure in the agent-level attention module and the relationship-level module.

Given the relationship set $\{\Re_1, \ldots, \Re_p\}$ for agent i between neighbors, after inputting latent features into the agent-level attention module, it can generate relationship-specific features of P groups, which is represented as $\{Z_i^{\Re_0}, Z_i^{\Re_1}, \ldots, Z_i^{\Re_p}\}$.

D. Relationship-Level Attention Module

In fact, each agent in the heterogeneous setting involves various kinds of relationships, and one relationship-specific feature representation is able to express features from one aspect merely. To generate a more comprehensive feature representation of agent, the method is required to integrate multiple relationships. To address this problem, we leverage an efficient relationship-level attention module to obtain the importance of diverse relationships automatically and integrate these relationship-specific features to generate the final feature representation. Concretely, for agent i, inputting P groups of relationship-specific feature representation generated from the agent-level attention module to the relationship-level module, the learned weights of each relationship $(\beta_i^{\Re_0}, \beta_i^{\Re_1}, \ldots, \beta_i^{\Re_p})$ can be represented as follows:

$$\left(\beta_i^{\Re_0}, \beta_i^{\Re_1}, \ldots, \beta_i^{\Re_p}\right) = \text{attention}_{\text{rela}}\left(Z_i^{\Re_0}, Z_i^{\Upsilon_1}, \ldots, Z_i^{\Re_p}\right). \tag{7}$$

The DNN module that executes the relationship-level attention is represented as $\text{attention}_{\text{rela}}$. It illustrates that the relationship-level attention module is able to apprehend diverse kinds of relationship information of the heterogeneous agents. To obtain the importance of certain relationships, the relationship-specific feature representation is transformed through one-layer MLP. Next, the importance of the relationship-specific feature representation is measured by leveraging an attention vector q. Besides, the importance of the relationship-specific features is averaged. We represent the importance of each relationship $p \in P$ as $w_i^{\Re_p}$

$$w_i^{\Re_p} = \frac{1}{|P|}\sum_{p\in P} q^T \cdot \tanh\left(W \cdot z_i^{\Re_p} + b\right). \tag{8}$$

Here, the weight matrix is represented as W, and the bias vector is denoted as b. All relationships and relationship-specific features share all the above parameters. After the weight of all relationships is received, they are normalized by leveraging the softmax operation. The importance of relationship $\Re_p$ for agent i can be calculated via the softmax equation as follows:

$$\beta_i^{\Re_p} = \frac{\exp\left(w_i^{\Re_p}\right)}{\sum_{i=1}^{P}\exp\left(w_i^{\Re_p}\right)}. \tag{9}$$

Apparently, the higher $\beta_i^{\Re_p}$, the more important $\Re_p$ is. Consequently, $\beta_i^{\Re_p}$ is considered as the contribution of the relationship $\Re_p$ for a certain assignment. These relationship-specific features can be integrated to general the final feature representation Z_i leveraging the learned importance as coefficients as follows:

$$Z_i = \sum_{p=1}^{P} \beta_i^{\Re_p} Z_i^{\Re_p}. \tag{10}$$

To explain the feature aggregation procedure of the relationship-level attention module more intuitively, we present a concise illustration in Fig. 4. The final integrated feature representation is aggregated by all relationship-specific features of neighbor agents. Then, the ultimate feature representation is input into the individual Q network to get the Q value, and then, the global Q value is obtained through the value decomposition module. The notations and explanations that we use throughout the method are illustrated in Nomenclature.

E. Algorithmic Summary

The details of MA-HA are illustrated in Algorithm 1. As shown in Lines 1–9, we initialize the parameters of networks in each episode, and each agent i leverages its policy or ε-greedy exploration to select the action a_i and execute it. Then, the agents obtain the reward r_t and new observation o_{t+1}. We store the obtained information of all agents $(o_t, \vec{a}_t, r_t, o_{t+1})$ at the replay buffer. While the experience stored at the replay buffer is enough, we randomly sample a batch of samples for training. First, during the feature preprocessing stage, we use the encoder module and project layer to preprocess the feature, as shown in Lines 10–14. The obtained feature is a low-level feature vector and contains solely local observation of agent i.

Subsequently, during the feature aggregation stage, we aggregate the feature of neighbors for the agent by heterogeneous graph attention layers, which contains agent- and relationship-level attention modules, as shown in Lines 16–24. The relationship-level attention module obtains the importance of diverse relationships automatically and integrates these relationship-specific features to generate a new synthetic feature representation. At each timestep, a heterogeneous graph attention layer corresponds to one round of feature aggregation of neighbor agents. By stacking several layers, we can obtain the final feature representation Z_i for agent i.

Finally, during the value decomposition stage, we compute the joint value Q_{tot} leveraging the mixing network. The mixing network is a feedforward neural network that takes the

Algorithm 1 Pseudocode of MA-HA

Input: The local observation $o_i \in O$ of agent i, environment state s, the action-observation history τ_i of agent i, the relationship set $\{\Re_0, \Re_1, \ldots, \Re_p\}$, and the attention head number K.
Output: The global action-value function Q_{tot}.
Initial the parameters of networks, the frequency of network updating, the maximum size of replay buffer, the maximum length of an episode, the learning rate, the random process $\mathcal{N}$ for action exploration.
for episode = 1 to M **do**
  for time step $t = 1$ to max length of episode **do**
    for each agent $i \in N$ **do**
    $a_i = a(o_i) + \mathcal{N}_t$
    Execute actions $\vec{a}$ and obtain r_t and o_{t+1}
    Store $(o_t, \vec{a}_t, r_t, o_{t+1})$ in replay buffer
    for each time step t in the episode of batch b **do**
      random batch b from replay buffer
      Stage 1: Feature preprocessing stage.
      Receive local observation o_i
      Obtain feature h_i through encoder module
      Input feature h_i to project layer
      Obtain feature h_i' by Eq.(2)
      Stage 2: Feature aggregation stage.
      Receive the relationship-based neighbors $N_i^{\Re}$
      for k = 1 to K **do**
        for $j \in N_i^{\Re}$
          Calculate importance coefficient $\alpha_{ij}^{\Re}$ by Eq. (4)
        end for
        Obtain relationship-specific feature $z_i^{\Re}$ by Eq. (5)
        Concatenate the learned feature by Eq. (6)
      end for
      Fuse relationship-specific embedding by Eq. (10)
      Obtain the final feature embedding Z_i
      Input the Z_i into individual Q network
      Obtain the individual Q value $Q_i(o_i, a_i, Z_i)$
      Stage 3: Value decomposition stage.
      Calculate the target Q_{tot} using mixing network
    end for
    Minimize loss function by Eq. (1)
    Update parameters of networks
    Output global action-value Q_{tot}
  end for
end for

individual values as input and mixes the values monotonically, thereby generating the joint value Q_{tot}. Under the CTDE paradigm, the mixing network will be removed during the execution phase. Next, we leverage the global value Q_{tot} to compute the gradient of the expected reward of the agent by backpropagation and leverage the Adam optimizer to update the policy. We update the parameters of networks for agent i at the end of each episode; all parameters can be updated by the standard TD loss of the global Q values according to (1). The initialization process and training period are repeated subsequently until the agent wins the game.

The MA-HA method is able to process various types of agents and relationships in the heterogeneous settings. Besides, the method can be parallelized easily because the agent-level attention and relationship-level attention can be parallelized across agent pairs and relationships, respectively. The attention module is highly effective, in which the computation of the attention module can be calculated across all agents and relationships independently. Given a relationship $\Re$, the time complexity of the proposed attention module is calculated as $O(V_{\Re} F_1 F_2 K + E_{\Re} F_1 K)$, where the number of attention heads is denoted as K, the number of agents in the heterogeneous environment is represented as $V_{\Re}$, $E_{\Re}$ denotes the number of relationship-based agent pairs, F_1 denotes the number of rows, and F_2 denotes the number of columns of M. Consequently, the time complexity is linearly related to the amounts of agents and relationship-based agent pairs.

IV. Experiments

In this section, the performance of MA-HA is evaluated in several heterogeneous environments. We first conduct experiments on the mixed cooperative-competitive scenarios extending from predator–prey [22]. To access the effectiveness of MA-HA, we employ multiagent deep deterministic policy gradient (MADDPG) [5], multiagent reinforcement learning with actor-attention-critic (MAAC) [6], multiagent actor-critic with hierarchical graph attention network (HAMA) [22], and actor-critic with graph attention network (GA-AC) [23] as the baseline methods. The last three methods all use the GNNs abovementioned and have achieved advanced performance in several tasks of the predator–prey environment.

Next, we test MA-HA on the SMAC platform, which is constructed on the StarCraft II game.

We employ QMIX [28], Q-learning with transformation (QTRAN) [29], duplex dueling multiagent Q-learning (QPLEX) [30], nearly decomposable Q-learning (NDQ) [12], temporal message control (TMC) [17], and graph neural network for multiagent comunication (GRC) [24] as baselines in SMAC environment [32]. These methods use the value decomposition or GNN and have state-of-the-art performance on SMAC. As MA-HA can be fused with any value decomposition method, we select the simple value decomposition network (VDN) [27] mixing network on predator–prey and the more complicated QMIX mixing network on the SMAC benchmark. Baselines are introduced as follows.

1) MADDPG [5] employs the framework of CTDE, which favors the agents to exploit additional information during the training stage.
2) MAAC [6] applies a soft-attention mechanism to learn a centralized critic network. The method can decide which agents to concentrate on dynamically at each timestep.
3) HAMA [22] applies a hierarchical attention mechanism to GNNs, which adequately represents the relationships among groups of agents.
4) GA-AC [23] leverages a two-stage attention mechanism to model the relationship between agents.
5) QMIX [28] proposes a network in which the joint action values are estimated by the nonlinear combination of individual action values.

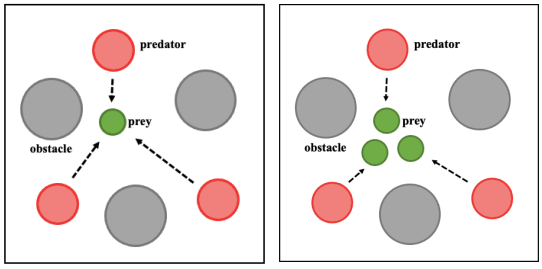

Fig. 5. Illustrations of the experimental environments: three predators versus one prey (left) and three predators versus three preys (right).

6) QTRAN [29] presents a more general decomposition by factorizing the value function into a VDN-factorized function with linear additive constraints.
7) QPLEX [30] designs a novel dueling network to decompose the value function.
8) NDQ [12] leverages communication minimization to effectively learn the value function decomposition, in which agents send messages to others occasionally.
9) TMC [17] stores messages of agents in the buffer and sends new messages out only when the messages contain relatively new information compared to old ones.
10) GRC [24] uses GNNs to model the relationship among agents to assist the communication learning of agents.

A. Predator–Prey

1) Three Predators Versus One Prey: As shown in Fig. 5, three predators versus one prey environment contains two competing sets of agents, along with some obstacles (the color is gray). The target of three homogeneous predators (the color is red) is to catch one heterogeneous prey (the color is green). At the same time, the prey's target is to dodge from the three predators. However, the three predators have slower speeds and accelerations than the prey, so they necessitate cooperation to hunt the prey. Once the prey is captured by the three predators, the prey gets a negative reward of -10, while the predator gets a positive reward of $+10$. The prey will obtain a negative reward if it flees a specific domain. This setup is to prevent it from escaping further. It is worth noting that both predators and prey attempt to maximize their cumulative returns, leading to competition between predators and prey.

To compare the performance of MA-HA, we adopt the methods mentioned above as baselines: MADDPG, MAAC, HAMA, and GA-AC. Each method is first trained in self-play, meaning that both predators and prey are learned in the uniform method. Then, we use the learned agent to play with other agents learned by other methods to verify the learned policies. Table I summarizes the average scores and the standard deviation that a predator is able to receive per step. Table I shows that predators trained with MA-HA scored higher or similar while playing with prey learned with other methods. At the same time, the predators trained by each method had the lowest points when capturing the prey trained by MA-HA.

In the predator–prey scenarios, we define other agents and obstacles within an agent's field of vision as neighbors. For a predator agent, it has three types of relationships with the allied predator, enemy prey, and obstacle, respectively. For a prey agent, it only has two types of relationships with the enemy predator and obstacle. It should be noted that MA-HA still achieves the best performance when training single prey that did not require cooperation, illustrating that it uses a hierarchical attention mechanism to effectively weigh the attention to the relationship with predators and obstacles.

TABLE I
AVERAGE POINTS OF PREDATORS IN THREE PREDATORS VERSUS ONE PREY ENVIRONMENT

Predator (number=3)	Prey (number=1)				
	MADDPG	MAAC	HAMA	GA-AC	MA-HA
MADDPG	0.30±0.02	0.23±0.02	0.07±0.02	0.21±0.03	0.05±0.02
MAAC	0.35±0.02	0.39±0.03	0.07±0.01	0.22±0.02	0.05±0.02
HAMA	0.45±0.03	0.39±0.03	0.16±0.01	0.24±0.03	0.12±0.03
GA-AC	0.38±0.03	0.42±0.02	0.09±0.02	0.34±0.02	0.08±0.02
MA-HA	**0.52±0.02**	**0.45±0.03**	**0.32±0.01**	**0.48±0.02**	**0.21±0.01**

TABLE II
AVERAGE POINTS OF PREDATORS IN THREE PREDATORS VERSUS THREE PREYS ENVIRONMENT

Predator (number=3)	Prey (number=3)				
	MADDPG	MAAC	HAMA	GA-AC	MA-HA
MADDPG	1.18±0.13	1.05±0.22	0.02±0.01	0.12±0.02	0.02±0.01
MAAC	0.65±0.20	0.33±0.13	0.07±0.04	0.25±0.03	0.03±0.01
HAMA	6.33±0.10	3.36±0.34	1.19±0.09	1.47±0.06	0.82±0.03
GA-AC	4.26±0.15	2.74±0.26	0.78±0.10	0.65±0.03	0.52±0.02
MA-HA	**8.25±0.12**	**5.23±0.28**	**3.62±0.15**	**3.68±0.12**	**2.12±0.08**

2) Three Predators Versus Three Preys: To further verify the validity of the method in a more complex heterogeneous environment, we employ a variation of the initial three predators versus one prey environment, which is three predators versus three preys setting. In this variation, there is no reward if the predator recaptures an already captured prey. When the predators have captured all preys, the game is over, and the predator obtains an additional return $+10 * t_r$. The number of remaining timesteps in the episode is represented as t_r.

Table II summarizes the experimental results of the three predators versus three preys setting; each agent is learned by the self-play way and plays against agents trained with other methods. As illustrated in Table II, the predators trained with MA-HA scored the highest, while prey trained with MA-HA provided the best defense. The performance of MA-HA agents is significantly better than agents learned by other methods. In the three predators versus one predator setting, the predator's sole policy is to capture a particular prey in a cooperation relationship. Consequently, the predator trained with MA-HA does not improve its performance significantly compared with other algorithms. Nevertheless, in the three predators versus three preys setting, each predator is able to select different policies, such as chasing prey alone or capturing preys cooperating with other homogeneous predators. The superior performance of MA-HA is possible due to the fact that it can generate better feature representation with the relationship information among neighboring agents by the agent-level attention module and the relationship-level attention module.

Fig. 6. Screenshots of several SMAC scenarios from [32].

TABLE III
MEDIAN WIN PERCENTAGE OF DIFFERENT MAPS

Scenario	QMIX	QPLEX	NDQ	TMC	GRC	Ours
2s3z	**97**	**97**	93	96	94	96
3s5z	**94**	91	90	92	**94**	94
1c3s5z	89	92	95	**96**	92	95
3s5z_3s6z	0	0	3	0	0	**16**
2c3s5z	65	51	75	81	71	**91**
3c5s7z	56	39	65	67	72	**89**
1o2r_vs_4r	38	43	71	57	61	**84**
1o10b_vs_1r	27	42	73	55	52	**93**
MMM2	51	32	64	70	83	**92**
MMM3	45	51	55	51	59	**82**
MMM4	42	23	41	52	56	**78**

B. SMAC

1) Setting: SMAC is built on the prevalent real-time strategy game: StarCraft II. The full game of StarCraft II has also been used as a reinforcement learning environment, as there are many interesting challenges in the game. DeepMind's AlphaStar [33] has performed the human level of play in the StarCraft II matchup leveraging a centralized controller. SMAC, by contrast, is not an environment for agents to learn to play the full gameplay of StarCraft II. Instead of addressing the full game of StarCraft II with macromanagement and tactics, SMAC focuses on tackling decentralized micromanagement challenges.

SMAC contains several StarCraft II microscenarios, as shown in Fig. 6. These microscenarios are well-designed, and the agent requires acquiring no less than one micromanagement skill to beat the enemy agent. To verify the effectiveness and superiority of MA-HA, we evaluate it in 11 heterogeneous scenarios of SMAC. All scenarios have diverse agent types or diverse agent numbers.

2) Performance: To demonstrate the performance of the proposed method and baselines, Table III shows the results for all 11 maps. We used the median win rate of the episodes during the training procedure as the primary evaluation metric [32]. MA-HA has the best performance in up to nine scenarios among all 11 scenarios while almost tying with QMIX and other baselines in the rest two easy scenarios. In the easy scenarios, 2s3z, 3s5z, and 1c3s5z, both MA-HA and baselines showed near-perfect win rates. However, in the super-hard map, 3s5z_3s6z, MA-HA beats all the other methods, almost all of which completely failed. Therefore, we do not further show the learning curve for these four scenarios.

For other hard and super-hard scenarios, we use Figs. 7 and 8 to show the learning curve of some representative baselines and MA-HA. The results contain the

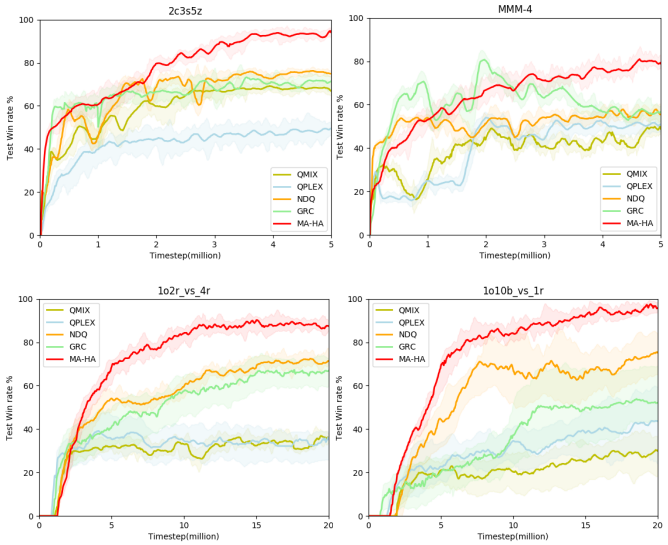

Fig. 7. Median test win rates on several SMAC maps.

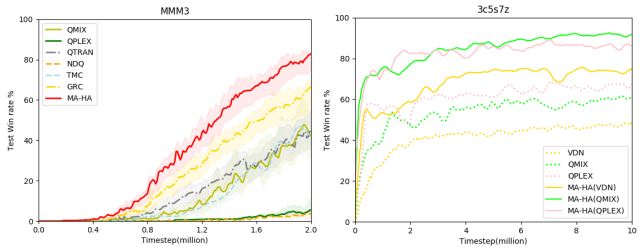

Fig. 8. Performance of MA-HA and all baselines in MMM3 map scenario (left). Average test win rates of MA-HA integrated with different value decomposition methods, including VDN, QMIX, and QPLEX (right).

performance of the median and 25%–75% percentile, which is recommended in [32] to avoid any outliers. As shown in Fig. 7, the MA-HA significantly outperforms QMIX and NDQ on these hard and super-hard maps, showing its high coordination ability even in complicated heterogeneous settings.

We analyze the acquired actions of agents to explain the basic principles behind these results. In map 2c3s5z, MA-HA has successfully learned the trick of intercepting enemy military Zealots with allied military Zealots, therefore protecting the allied military Stalkers from serious destruction. This illustrates that allied Zealots pay more attention to the relationship with the enemy Zealots and the allied Zealots than to the relationship with Stalkers and Colossus in some states, indicating that the attentional mechanism at the relationship level has a great influence on them.

In map 3s5z, the policy of the MA-HA agent is considerably different from map 2c3s5z. Allied Zealots learn the skill of bypassing enemy military Zealots and attacking the enemy military Stalkers and then deploy allied Stalkers to flank enemy Zealots on either side. This means that the allied Zealots give priority to the relationship with the enemy Stalkers, rather than the relationship with the enemy Zealots, indicating that the relationship-level attention module exerts

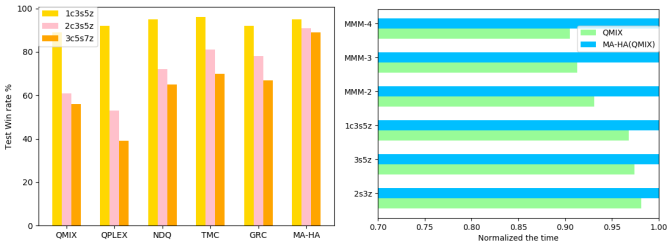

Fig. 9. Average win rates for MA-HA and baselines in 1c3s5z, 2c3s5z, and 3c5s7z scenarios (left). Time of QMIX compared with MA-HA (right).

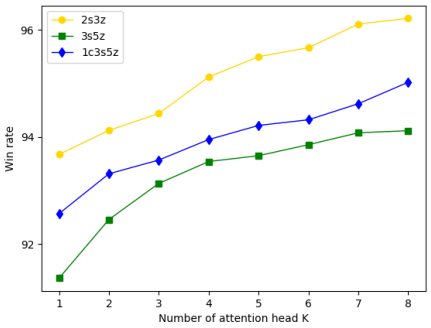

Fig. 10. Parameter sensitivity of MA-HA w.r.t. the number of attention head K on several map scenarios.

an enormous function on it. Such a highly complicated and coordinated strategy cannot be obtained by other methods.

In map 3s5z versus 3s6z, the allied Zealots learn the technology of holding enemy Zealots to protect allied Stalkers while attacking enemy Stalkers. In map MMM-3, the allied Medivacs learn to be close to the enemies, absorb their fire, and then withdraw to treat the appropriate allied agent. MA-HA employs the hierarchical attention mechanism to achieve such complicated skills that MA-HA starts to win, while other methods generally failed. MA-HA derives admirable decentralized policies for different types of agents compared with the existing MARL methods in heterogeneous settings.

To more clearly illustrate the learning efficiency and performance of MA-HA compared to all other baselines, Fig. 8 (left) shows the learning curve of all methods for the first two million timesteps of the total ten million timesteps in the MMM3 scenario. Notably, both QMIX and NDQ fail in MMM3 as it is difficult to coordinate the actions of diverse agents with abundant relationships. GRC outperforms the other baselines because it models the relationship between agents. However, it is not designed for heterogeneous settings and, therefore, does not perform MA-HA.

MA-HA can be integrated with any MARL value function decomposition method. For example, we apply it to existing value decomposition methods: VDN, QMIX, and QPLEX. The combined methods, called MA-HA(VDN), MA-HA(QMIX), and MA-HA(QPLEX), respectively, are tested on map 3c5s7z. As shown in Fig. 8 (right), all integrated methods perform better than the original value decomposition methods, which indicates that the heterogeneous graph attention network module of MA-HA can significantly enhance policy learning and coordination.

In general, as the type of agents increases, policy learning becomes increasingly tough, while the interactions and relationships between diverse types of agents also become increasingly complicated. To intuitively verify and illustrate that the proposed method can easily scale the number of agents, we compare the performance of MA-HA and baselines in scenarios 1c3s5z, 2c3s5z, and 3c5s7z. As shown in Fig. 9 (left), as the number of agents increases, the difficulty of the agent learning strategy increases, and the win rate of baselines dropped significantly. Nevertheless, the MA-HA algorithm still maintains a very high win rate and obtains accurate policy for agents as the number of agents increases.

It is worth noting that, compared with the baseline value decomposition methods, such as QMIX, the network structure of MA-HA has an extra feature aggregation component, which requires more computing time for our method. In the same scenario and parameter settings, we compared the training time required for MA-HA(QMIX) and QMIX in several scenarios. For comparison facilitation, we normalize the training time based on MA-HA. As shown in Fig. 9 (right), MA-HA requires more training time than QMIX as the complexity of the scenario increases. Nevertheless, even in the most complicated scenario (MMM-4), MA-HA requires only about 10% more training time than QMIX. Compared to the significant improvement in win rate (42%–78%), this growth in consumption is tolerable.

To examine the effect of multihead attention, we explore the performance of MA-HA in an SMAC environment with different numbers of attention heads. As shown in Fig. 10, the more attention head generally improves the performance of the MA-HA method. The number of attention heads set to 1 in Fig. 10 indicates that the multihead is removed. However, the performance improved only slightly with the change in the number of attention heads. At the same time, we find that multihead attention can enhance the stability of the training process.

C. Hyperparameters of MA-HA Method for SMAC

All experiments are constructed on CPU Intel Xeon Silver 4210 and GPU Nvidia RTX 2080 using five random seeds on SMAC2, the latest version 4.10. In the scenarios of SMAC, the training time of two million timesteps is approximately 12–20 h, which is ranged according to the agent number and scenario features of each map. Depending on the difficulty level of the scenario, the number of the total training timesteps ranges from two to 20 million until the learning curve stabilizes. The hyperparameters are listed in Table IV.

In SMAC, the agent obtains the local observation within its field of view, with the view range set to 9. The feature obtained by the agent includes the following attributes of allied units and enemy units within its field of view: *unit type, relative x, relative y, distance, health,* and *shield.* The action space of the agent is of dimension four, which contains actions *move, no-op, attack,* and *stop*. The range of fire is set to 6. The agent can obtain an additional bonus of 10 for each enemy unit that it kills and 200 for each battle that it wins.

TABLE IV
HYPERPARAMETERS OF MA-HA FOR SMAC

Hyper-parameter	Value
Optimizer	Adam
Batch size	64
Replay buffer size	5000
Target update	1000
Discount factor	0.99
Learning rate	1e-5
Training step	(2e+7, 2e+8)
Attention head	8

V. CONCLUSION

In this article, we concentrate on obtaining policies for heterogeneous agents with different attributes in the multiagent environment. We learn the relationship between agents by defining a novel hierarchical graph attention module. The method can be flexibly integrated with diverse MARL methods. Experimental results in the SMAC environment and predator–prey environment illustrate that the MA-HA method with a heterogeneous graph attention network can get better performance in comparison with existing MARL algorithms. In future work, we intend to conduct the proposed method on realistic heterogeneous multiagent environments.

REFERENCES

[1] Z. Zhang, D. Wang, and J. Gao, "Learning automata-based multiagent reinforcement learning for optimization of cooperative tasks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 10, pp. 4639–4652, Oct. 2021.

[2] C. Sun, W. Liu, and L. Dong, "Reinforcement learning with task decomposition for cooperative multiagent systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 5, pp. 2054–2065, May 2021.

[3] D. Silver et al., "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[4] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–9.

[5] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6379–6390.

[6] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2961–2970.

[7] 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.

[8] S. Sukhbaatar et al., "Learning multiagent communication with backpropagation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2244–2252.

[9] P. Peng et al., "Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games," 2017, *arXiv:1703.10069*.

[10] J. Jiang and Z. Lu, "Learning attentional communication for multi-agent cooperation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 7254–7264.

[11] A. Das et al., "TarMAC: Targeted multi-agent communication," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 1538–1546.

[12] T. Wang, J. Wang, C. Zheng, and C. Zhang, "Learning nearly decomposable value functions via communication minimization," 2019, *arXiv:1910.05366*.

[13] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1263–1272.

[14] A. Agarwal, S. Kumar, and K. Sycara, "Learning transferable cooperative behavior in multi-agent teams," 2019, *arXiv:1906.01202*.

[15] A. Malysheva, D. Kudenko, and A. Shpilman, "MAGNet: Multi-agent graph network for deep multi-agent reinforcement learning," in *Proc. 16th Int. Symp. Problems Redundancy Inf. Control Syst. (Redundancy)*, Oct. 2019, pp. 171–176.

[16] J. Jiang, C. Dun, T. Huang, and Z. Lu, "Graph convolutional reinforcement learning," 2018, *arXiv:1810.09202*.

[17] S. Q. Zhang, Q. Zhang, and J. Lin, "Succinct and robust multi-agent communication with temporal message control," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 1–12.

[18] T. Wang, R. Liao, J. Ba, and S. Fidler, "NerveNet: Learning structured policy with graph neural networks," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–26.

[19] J. Sheng et al., "Learning structured communication for multi-agent reinforcement learning," 2020, *arXiv:2002.04235*.

[20] Y. Niu, R. Paleja, and M. Gombolay, "Multi-agent graph-attention communication and teaming," in *Proc. Int. Conf. Auton. Agents Multi Agent Syst.*, 2021, pp. 1–12.

[21] W. Böhmer, V. Kurin, and S. Whiteson, "Deep coordination graphs," in *Proc. 37th Int. Conf. Mach. Learn.*, vol. 119, 2020, pp. 980–991.

[22] H. Ryu, H. Shin, and J. Park, "Multi-agent actor-critic with hierarchical graph attention network," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 7236–7243.

[23] 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.*, 2020, pp. 7211–7218.

[24] S. Shen, Y. Fu, H. Su, and C. Wang, "Graphcomm: A graph neural network based method for multi-agent reinforcement learning," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Jun. 2021, pp. 3510–3514.

[25] F. A. Oliehoek, M. T. J. Spaan, and N. Vlassis, "Optimal and approximate Q-value functions for decentralized POMDPs," *J. Artif. Intell. Res.*, vol. 32, pp. 289–353, May 2008.

[26] W. Du and S. Ding, "A survey on multi-agent deep reinforcement learning: From the perspective of challenges and applications," *Artif. Intell. Rev.*, vol. 54, pp. 1–24, Nov. 2020.

[27] P. Sunehag et al., "Value-decomposition networks for cooperative multiagent learning," in *Proc. AAMAS*, 2018, pp. 1–17.

[28] 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.

[29] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, "QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *Proc. 31st Int. Conf. Mach. Learn.*, 2019, pp. 5887–5896.

[30] J. H. Wang, Z. Z. Ren, T. Liu, Y. Yu, and C. J. Zhang, "QPLEX: Duplex dueling multi-agent Q-learning," in *Proc. 9th Int. Conf. Learn. Represent.*, 2021, pp. 1–27.

[31] Y. Yang et al., "Q-value path decomposition for deep multiagent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 10706–10715.

[32] M. Samvelyan et al., "The starcraft multi-agent challenge," in *Proc. 18th Int. Conf. Auto. Agents MultiAgent Syst.*, 2019, pp. 2186–2188.

[33] O. Vinyals et al., "Grandmaster level in starcraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.