Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

Partially Observable Mean Field Multi-Agent Reinforcement Learning Based on Graph–Attention

Min Yang | Guanjun Liu* | Ziyuan Zhou

[1]Department of Computer Science, Tongji University, Shanghai, China

Summary

Traditional multi-agent reinforcement learning algorithms are difficultly applied in a large-scale multi-agent environment. The introduction of mean field theory has enhanced the scalability of multi-agent reinforcement learning in recent years. This paper considers partially observable multi-agent reinforcement learning (MARL), where each agent can only observe other agents within a fixed range. This partial observability affects the agent's ability to assess the quality of the actions of surrounding agents. This paper focuses on developing a method to capture more effective information from local observations in order to select more effective actions. Previous work in this field employs probability distributions or weighted mean field to update the average actions of neighborhood agents, but it does not fully consider the feature information of surrounding neighbors and leads to a local optimum. In this paper, we propose a novel multi-agent reinforcement learning algorithm, Partially Observable Mean Field Multi-Agent Reinforcement Learning based on Graph–Attention (GAMFQ) to remedy this flaw. GAMFQ uses a graph attention module and a mean field module to describe how an agent is influenced by the actions of other agents at each time step. This graph attention module consists of a graph attention encoder and a differentiable attention mechanism, and this mechanism outputs a dynamic graph to represent the effectiveness of neighborhood agents against central agents. The mean–field module approximates the effect of a neighborhood agent on a central agent as the average effect of effective neighborhood agents. We evaluate GAMFQ on three challenging tasks in the MAgents framework. Experiments show that GAMFQ outperforms baselines including the state-of-the-art partially observable mean-field reinforcement learning algorithms. The code for this paper is here https://github.com/yangmin32/GPMF.

KEYWORDS:

Graph–Attention, Multi-agent reinforcement learning, Mean field theory, Partial observation

1 | INTRODUCTION

Reinforcement learning has been widely used in video games [26] and recently in education [7]. For multi-agent reinforcement learning (MARL) [33], it involves multiple autonomous agents that make autonomous decisions to accomplish some specific competitive or cooperative tasks by maximizing global reward, it has been applied in some real-world scenarios such as autonomous mobile [21] drone swarm confrontation[1] and multi-UAV collaboratively delivering goods [22]. For example, in some of the drone swarm adversarial tasks, drones need to make actions based on autonomous decisions. Due to the inevitable death of some drones in the confrontation environment [38], the surviving drones must constantly evolve their strategies in real-time during the interaction with the environment to obtain the overall maximum reward. In order to make better interaction among agents, it is required that each agent in the multi-agent system can effectively perceive environmental information and fully acquire the information of surrounding agents.

However, the global communication cost among multiple agents is high, and in many practical tasks, each agent only observes part of the environmental information. Take the task of Autonomous Driving as an example, each vehicle makes decision in the limited observation space which is a typical local observation scene. Each agent can only rely on limited observation information in the local observation environment, therefore the agent needs to learn a decentralized strategy. There are two common decentralization strategies. One is Centralized Training and Decentralized Execution (CTDE), which requires agents to communicate with each other during training and to independently make decisions based on their own observations during testing in order to adapt to large-scale multi-agent environments. Some classic algorithms using the CTDE framework such as MADDPG [15], QMIX [19] and MAVEN [16]. Another one takes the policy of decentralized training and decentralized execution, in which each agent can only observe part of the information during the training and testing phases, which is closer to the real environment with limited communication. Especially large-scale multi-agent environments are complex and non-stationary [10], it is difficult for agents to observe the entire environment globally, limiting their ability to find the best actions. Furthermore, as the number of agents increases, joint optimization of all information in a multi-agent environment may result in a huge joint state-action space, which also brings scalability challenges. This paper focuses on the second strategy.

Traditional multi-agent reinforcement learning algorithms are difficult to be applied in large-scale multi-agent environments, especially when the number of agents is exponential. Recent studies address the scalability issues of multi-agent reinforcement learning [31, 30, 12] by introducing mean-field theory, i.e., the multi-agent problem is reduced to a simple two-agent problem. However, Yang et al. [31] assumes that each agent can observe global information, which is difficult to apply in some real tasks. Therefore, it is necessary to study large-scale multi-agent reinforcement learning algorithms in partially observable cases [3]. In addition, researchers have intensively studied mean-field-based multi-agent reinforcement learning algorithms to improve performance in partially observable cases. One way is to further decompose the Q-function of the mean field-based multi-agent reinforcement learning algorithm [34, 6]. Another way uses probability distribution or weighted mean field to update the mean action of neighborhood agents [5, 37, 23, 28]. Hao [8] combined the graph attention with the mean field to calculate the interaction strength between agents when agents interact, but only considered the scene where the agent has a fixed relative position, and the agents can observe the global information. The difference is that when the agent is partially observable, we consider the dynamic change of the agent's position and the death scene of the agent, and construct a more flexible partial observable graph attention network based on the mean field.

However, for partially observable multi-agent mean field reinforcement learning, the existing methods do not fully consider the feature information of the surrounding neighbors, which will lead to falling into local optimum. This paper focuses on identifying the neighborhood agents that may have the greater influence on the central agent in a limited observation space, in order to avoid the local optimum issue. Since the graph neural network [29] can fully aggregate the relationship between the central agent and its surrounding neighbors, we propose a graph attention-based mechanism to calculate the importance of neighbor agents to estimate the average action more efficiently.

The main contributions of this paper are as follows:

- We propose a partially observable mean–field reinforcement learning based on the graph–attention (GAMFQ), which can learn a decentralized agent policy from an environment without requiring global information of an environment. In the case of partially observable large-scale agents, the judgment of the importance of neighbor agents is insufficient in our GAMFQ.

- We theoretically demonstrate that the settings of the GAMFQ algorithm are close to Nash equilibrium.

- Experiments on three challenging tasks in the MAgents framework show that GAMFQ outperforms two baseline algorithms as well as the state-of-the-art partially observable mean-field reinforcement learning algorithms.

2 | RELATED WORK

Most of the recent MARL algorithms for partial observability research are model-free reinforcement learning algorithms based on the CTDE framework. The most classic algorithm MADDPG [15] introduces critics that can observe global information in training to guide actor training, but only use actors with local observation information to take actions in testing. QMIX[19] uses a hybrid network to combine the local value functions of a single agent, and adds global state information assistance in the training and learning process to improve the performance of the algorithm. MAVEN [16] is able to solve complex multi-agent tasks by introducing latent spaces for hierarchical control by value-mixing and policy-based approaches. However, these multi-agent reinforcement learning algorithm using the CTDE framework is difficult to scale to large-scale multi-agent environments, because there will be hard-to-observe global information that prevents the agents from training better policies.

For large-scale multi-agent environments, Yang et al. [31] introduced the mean–field theory, which approximates the interaction of many agents as the interaction between the central agent and the average effects from neighboring agents. However, partially observed multi-agent mean–field reinforcement learning algorithms still have a space to improve. Some researchers further decompose the Q-function of the mean field based multi-agent reinforcement learning algorithm. Zhang et al. [34] trained agents through the CTDE paradigm, transforming each agent's Q-function into its local Q-function and its mean field Q-function, but this approach is not strictly partially observable. Gu et al. [6] proposes a mean field multi-agent reinforcement learning algorithm with local training and decentralized execution. The Q-function is decomposed by grouping the observable neighbor states of each agent in a multi-agent system, so that the Q-function can be updated locally. In addition, some researchers have focused on improving the mean action in mean field reinforcement learning. Fang et al. [5] adds the idea of mean field to MADDPG, and proposes a multi-agent reinforcement learning algorithm based on weighted mean field, so that MADDPG can adapt to large-scale multi-agent environment. Wang et al. [28] propose a weighted mean-field multi-agent reinforcement learning algorithm based on reward attribution decomposition by approximating the weighted mean field as a joint optimization of implicit reward distribution between a central agent and its neighbors. Zhou et al. [37] uses the average action of neighbor agents as a label, and trained a mean field prediction network to replace the average action. Subramanian et al. [23] proposed two multi-agent mean field reinforcement learning algorithms based on partially observable settings: POMFQ(FOR) and POMFQ(PDO), extracting partial samples from Dirichlet or Gamma distribution to estimate partial observable mean action. Although these methods achieve good results, they do not fully consider the feature information of surrounding neighbors.

Graph Neural Networks (GNNs) are able to mine graph structures from data for learning. In multi-agent reinforcement learning, GNNs can be used to model interactions between agents. In recent work, graph attention mechanisms have been used for multi-agent reinforcement learning. Zhang et al. [32] integrated the importance of the information of surrounding agents based on the multi-head attention mechanism, effectively integrate the key information of the graph to represent the environment and improve the cooperation strategy of agents with the help of multi-agent reinforcement learning. DCG [2] decomposed the joint value function of all agents into gains between pairs of agents according to the coordination graph, which can flexibly balance the performance and generalization ability of agents. Li et al. [13] proposed a deep implicit coordination graph (DICG) structure that can adapt to dynamic environments and learn implicit reasoning about joint actions or values through graph neural networks. Ruan et al. [20] proposed a graph-based coordination strategy, which decomposes the joint team strategy into a graph generator and a graph-based coordination strategy to realize the coordination behavior between agents. MAGIC [17] more accurately represented the interactions between agents during communication by modifying the standard graph attention network and compatible with differentiable directed graphs.

In the dynamic MARL system where competition and confrontation coexist, it is very difficult to directly apply the graph neural network, because the agent will die, the graph structure of the constructed large-scale agent system has the problem of large spatial dimension. However, graph neural networks can better mine the relationship between features, and the introduction of mean-field theory can further improve the advantages of mean-field multi-agent reinforcement learning.

Our approach differs from related work above in that it uses a graph attention mechanism to select surrounding agents that are more important to the central agent in a partially observable environment. GAMFQ uses a graph attention module and a mean field module to describe how an agent is influenced by the actions of other agents at each time step, where graph attention consists of a graph attention encoder and a differentiable attention mechanism, and finally outputs a dynamic graph to represent the effectiveness of the neighborhood agent to the central agent. The mean field module approximates the influence of a neighborhood agent on a central agent as the average influence of the effective neighborhood agents. Using these two modules together is able to efficiently estimate the average action of surrounding agents in partially observable situations. GAMFQ does not require global information about the environment to learn decentralized agent policies from the environment.

3 | MOTIVATION & PRELIMINARIES

In this section, we represent discrete-time non-cooperative multi-agent task modeling as a stochastic game (SG). SG can be defined as a tuple $< S, A^1, \ldots, A^N, r^1, \ldots, r^N, p, \gamma >$, where S represents the true state of the environment. Each agent $j \in \{1, \ldots, N\}$ chooses an action at each time step $a^j \in A^j$. The reward function for agent j is $r^j : S \times A^1 \times \cdots \times A^N \rightarrow R$. State transitions are dynamically represented as $p : S \times A^1 \times \cdots \times A^N \rightarrow \Omega(S)$. γ is a constant representing the discount factor. It represents a stable state, and in this stable state, all agents will not deviate from the best strategy given to others. The disadvantage is that it cannot be applied to the coexistence of multiple agents. Yang et al. [31] introduced mean field theory, which approximates the interaction of many agents as the interaction between the average effect of a central agent and neighboring agents, and solves the scalability problem of SG.

The Nash equilibrium of general and random games can be defined as a strategy tuple $\left(\pi_*^1, \cdots, \pi_*^N\right)$, for all $s \in S$ and $\forall \pi^i \in \Pi^i$, there is $v^j\left(s, \pi_*^1, \cdots, \pi_*^i, \cdots, \pi_*^N\right) \geq v^j\left(s, \pi_*^1, \cdots, \pi^i, \cdots, \pi_*^N\right)$. This shows that when all other agents are implementing their equilibrium strategy, no one agent will deviate from this equilibrium strategy and receive a strictly higher reward. When all agents follow the Nash equilibrium strategy, the Nash Q-function of agent j is $Q_*^j(s, a)$. Partially observable stochastic games can generate a partially observable Markov decision process (POMDP), we review the partially observable Markov decision (Dec-POMDP) in Section 3.1 and analyze the partially observable model from a theoretical perspective. Section 3.2 first introduces the globally observable mean-field multi-agent reinforcement learning, and then introduces the partially observable mean-field reinforcement learning algorithm (POMFQ) based on the POMDP framework, and analyzes the existing part of the observable in detail. The limitation of mean-field reinforcement learning POMFQ(FOR)[23] is that the feature information of surrounding neighbors is not fully considered. In a partially observable setting, each agent j observable neighborhood agent information o^j can be used to better mine the relationship between features through a graph attention network. Introducing graph attention networks into partially observable mean-field multi-agent reinforcement learning can further improve their performance, and Section 3.3 briefly introduces graph attention networks.

3.1 | Partially observable Markov decision process

We mainly study partially observable Markov decisions (Dec-POMDP) [3, 18, 33]. The partially observable Markov decision process of n agents can be represented as a tuple $\left\langle N, S, \left\{A^i\right\}_{i=1}^n, T, Z, R, O, \gamma\right\rangle$, where $N = \{1, \ldots, n\}$ represents the set of agents, S represents the global state, A^j represents the set of action spaces of the j-th agent, Z represents the observation space of the agents, and the agent j receives observation $o^j \in O^j$ through the observation function $Z(s, j) : S \times N \rightarrow O$, and the transition function $T : S \times A^1 \times \ldots \times A^n \times S \mapsto [0, 1]$ represents the environment transitions from a state to another one. At each time step t, the agent j chooses an action $a_t^j \in A^j$, gets a reward $r_t^j : S \times A^j \mapsto R$ w.r.t. a state and an action. $\gamma \in [0, 1]$ is a reward discount factor. Agent j has a stochastic policy π^j conditioned on its observation o^j or action observation history $\tau^j \in \left(Z \times A^j\right)$, and according to the all agents's joint policy $\pi \overset{\Delta}{=} \left[\pi^1, \ldots, \pi^N\right]$, The value function of agent j under the joint strategy π is the value function $v_\pi^j(s) = \sum_{t=0}^{\infty} \gamma^t E_{\pi,p}\left[r_t^j | s_0 = s\right]$ can be obtained, and then the Q-function can be formalized as $Q_\pi^j(s, a) = r^j(s, a) + \gamma E_{s' \sim p}\left[v_\pi^j\left(s'\right)\right]$. Our work is based on the POMDP framework.

3.2 | Partially Observable Mean Field Reinforcement Learning

Mean-field theory-based reinforcement learning algorithm [31] approximates interactions among multiple agents as two-agent interactions, where the second agent corresponds to the average effect of all other agents. Yang et al. [31] decomposes the multi-agent Q-function into pairwise interacting local Q-functions as follows:

$$Q_\pi^j(s, a) = \frac{1}{N^j} \sum_{k \in N(j)} Q_\pi^j\left(s, a^j, a^k\right) \tag{1}$$

where N^j is the index set of the neighbors of the agent j and a^j represents the discrete action of the agent j and is represented by one-shot coding. Mean field Q-function is cyclically updated according to Eq.2-5:

$$Q_\pi^j\left(s_t, a_t^j, \bar{a}_t^j\right) = (1 - \alpha) Q_\pi^j\left(s_t, a_t^j, \bar{a}_t^j\right) + \alpha\left[r_t^j + \gamma v^j\left(s_{t+1}\right)\right] \tag{2}$$

where

$$v^j\left(s_{t+1}\right)=\sum_{a^j_{t+1}} \pi^j\left(a^j_{t+1} \mid s_{t+1}, \tilde{a}^j_t\right) Q^j_\pi\left(s_{t+1}, a^j_{t+1}, \tilde{a}^j_t\right) \tag{3}$$

$$\bar{a}^j_t = \frac{1}{N}\sum_{k\neq j} a^k_t, a^k_t \sim \pi^k\left(\cdot \mid s_t, \bar{a}^k_{t-1}\right) \tag{4}$$

$$\pi^j\left(a^j_t \mid s_t, \bar{a}^j_{t-1}\right)=\frac{\exp\left(-\beta Q^j_\pi\left(s_t, a^j_t, \bar{a}^j_{t-1}\right)\right)}{\sum_{a^{j'}_t \in A^j} \exp\left(-\beta Q^j_\pi\left(s_t, a^{j'}_t, \bar{a}^j_{t-1}\right)\right)} \tag{5}$$

where $\bar{a}^j_t$ is the mean action of the neighborhood agent, r^j_t is the reward for agent j at time step t, v^j is the value function of agent j, and β is the Boltzmann parameter. Literature [31] assumes that each agent has global information, and for the central agent, the average action of the neighboring agents is updated by Eq. 4. However, in a partially observable multi-agent environment, the way of calculating the average action in Eq. 4 is no longer applicable.

In the case of partial observability, Subramanian et al. [23] take U samples from the Dirichlet distribution to update the average action of Eq. 4, and achieve better performance than the mean field reinforcement learning algorithm. The formula is as follows:

$$\begin{gathered} D^j(\theta) \propto \theta_1^{\eta_1-1+c_1} \cdots \theta_L^{\eta_L-1+c_L}; \\ \tilde{a}^j_{i,t} \sim D^j(\theta; \eta + c); \tilde{a}^j_t = \frac{1}{U}\sum_{i=1}^{i=U} \tilde{a}^j_{i,t} \end{gathered} \tag{6}$$

where L denotes the size of the action space, $c_1, \ldots, c_L$ denotes the number of occurrences of each action, η is the Dirichlet parameter, θ is the classification distribution. But the premise of the Dirichlet distribution is to assume that the characteristics of each agent are independent to achieve better clustering based on the characteristics of neighboring agents. In fact, in many multi-agent environments, the characteristics of each agent has a certain correlation, but the Dirichlet distribution does not consider this correlation, which makes it unable to accurately describe the central agent and the neighborhood agents. There will be some deviations in the related information. Figure 1 shows the process of a battle between the red and green teams, in which each agent can observe the information of the friendly agent, and the action space of the agent is $\{up, down, left, right\}$. The central agent enclosed by the red circle is affected by the surrounding friendly agents. We use the Dirichlet distribution to simulate and calculate the probability of the central agent moving in each direction, as shown below:

$$\begin{cases} p_{up} = 0.31 \\ p_{down} = 0.42 \\ p_{left} = 0.14 \\ p_{right} = 0.13 \end{cases} \tag{7}$$

It can be obtained that the probability of the agent moving down is the highest, which is essentially due to the large number of agents moving *down*. However, moving *up* is the optimal action for the agent to form an encirclement trend with friends. The Dirichlet distribution results in a local optimal solution rather than finding the optimal action.

Zhang et al. [35] believes that the correlation between two agents is crucial for multi-agent reinforcement learning. First, the paper calculates the correlation coefficient between each pair of agents, and then shields the communication among weakly correlated agents, thereby reducing the dimensionality of the state-action value network in the input space. Inspired by Zhang et al. [35], for large-scale partially observable multi-agent environments, it is more necessary to select the importance of neighborhood agents. In our paper, we will adopt a graph attention method to filter out more important neighborhood agents, discard unimportant agent information, and achieve more accurate estimation of the average actions of neighborhood agents.

3.3 | Graph Attention Network

Graph neural network [29] can better mine the graph structure form between data. Graph Attention Network (GAT) [25] is composed of a group of graph attention layers, each graph attention layer acts on the node feature vector of node i denoting as m_i through a weight matrix W , and then uses softmax to normalize the neighbor nodes of the central node:

$$e_{ij} = (Wm_i \| Wm_j) \tag{8}$$

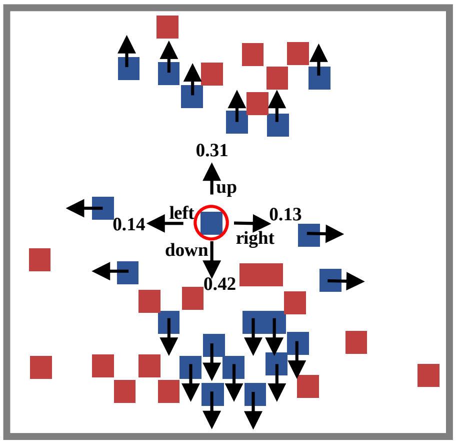


FIGURE 1 A battle environment of the red and blue groups, where the red agent in the center is distributed by Dirichlet to calculate the action.

$$\alpha_{ij} = \mathrm{softmax} j\left(e_{ij}\right) = \frac{\exp\left(e_{ij}\right)}{\sum_{k\in N_j} \exp\left(e_{jk}\right)} \tag{9}$$

where e_{ij} is the attention coefficient of each node, indicating the importance of node i to node j. Finally, the output features are obtained by weighting the input features h_i, and the update rule for each node j is:

$$e_j = \sigma\left(\sum_{i\in N_j} \alpha_{ij} W h_i\right) \tag{10}$$

where e_j represents the feature of node j, N_j is the set of adjacent nodes of node j, and $\sigma(\cdot)$ is a nonlinear activation function.

4 | APPROACH

In this section, we propose a novel method called Partially Observable Mean Field Multi-Agent Reinforcement Learning based on Graph–Attention (GAMFQ), which can be applied to large-scale partially observable MARL tasks, where the observation range of each agent is limited, and the feature information of other agents in the fixed neighborhood is intelligently observed. The overall architecture of the GAMFQ algorithm is depicted in Figure 2, including two important components: the Graph Attention Module and the Mean Field Module: (i) In our Graph–Attention Module, the information observed locally by each agent is spliced firstly. Then the high-dimensional feature representations are obtained by a latent space mapping process which followed by a one-layer LSTM network to obtain the time-series correlation of the target agent, and the hidden layer of the LSTM is used as the input of the graph attention module to initialize the constructed graph nodes. Then to enhance the aggregation of neighbor agents to target agent, a similar process is implemented as a FC mapping network followed by a GAT layer. After that, the final representation of agents are obtained by a MLP layer with the input of the representations of target agent and other observable agents. Finally, we adopt layer-normalized method to obtain the adjacency matrix $\left\{G^t\right\}_1^N$ via Gumbel Softmax. (ii) The Mean Field Module utilizes the adjacency matrix $\left\{G^t\right\}_1^N$ from Graph Attention Module to obtain adopting action from important neighbor agents, in which the joint Q-function of each agent j approximates the Mean-Field Q-function $Q^j(s,a) \approx Q^j_{\mathrm{POMF}}\left(s, a^j, \tilde{a}^j\right)$ of important neighbor agents, where the Q-value is partially observable mean-field(POMF) Q-value, and $\tilde{a}^j$ is the average action of the important neighborhood a gents that is partially observable by agent j. Each component is described in detail below.

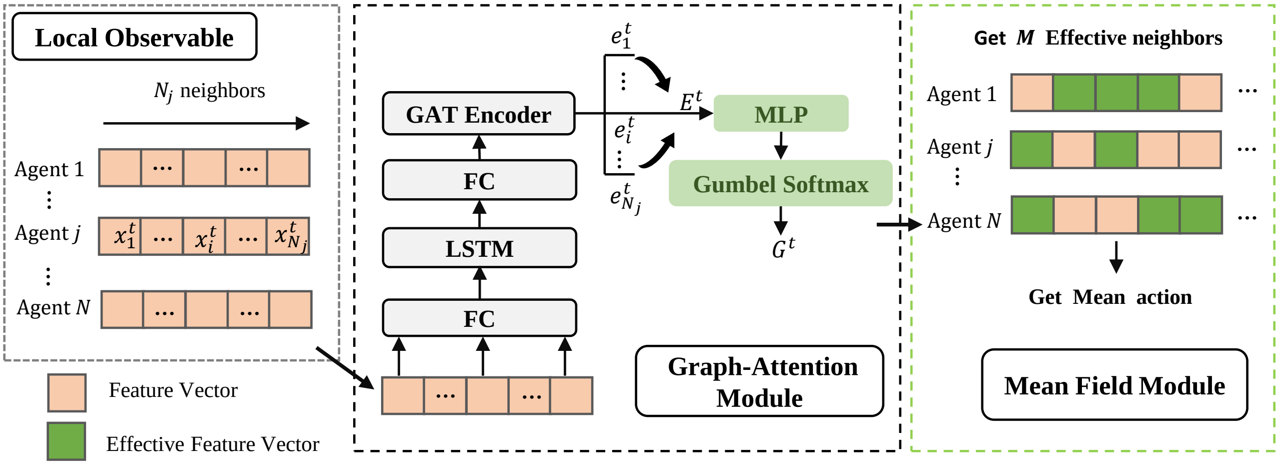


FIGURE 2 Schematic of GAMFQ. Each agent can observe the feature information of other agents within a fixed range, input it into the Graph–Attention Module, and output an adjacency matrix to represent the effectiveness of the neighborhood agent to the central agent.

4.1 | Graph–Attention Module

To more accurately re-determine the influence of agent j's neighbor N_j on itself, we need to be able to extract useful information from the local observations of agent j. The local observations of each agent include the embedding information of neighboring agents. For each agent j and each time step t, the information of a local observation of length L_j, is expressed as $o_j^t = \left(x_1^t, x_2^t, \cdots, x_{N_j}^t\right)$, where $x_{N_j}^t$ represents the feature of the N_j-th neighbor agent of agent j, and $o_j^t \in R^{N_j \times D}$, $x_i^t \in R^{1\times D}$. L_j is concatenated from the embedding features of each neighbor. Our goal is to learn an adjacency matrix $\left\{G^t\right\}_1^N$ to extract more important embedding information for the agent j from local observations at each time step t. Since graph neural networks can better mine the information of neighbor nodes, we propose a graph attention structure suitable for large-scale multi-agent systems. This structure focuses on information from different agents by associating weights to observations based on the relative importance of other agents in their local observations. The Graph–Attention structure is constructed by concatenating a graph attention encoder and a differentiable attention mechanism. For the local observation o_j^t of agent j at time step t, $o_j^{t'}$ is first encoded using a fully connected layer (FC) , and is passed to the LSTM layerin order to generate the hidden state h_j^t and cell state c_j^t of agent j, where h_j^t serves as the input of the graph attention module to initialize the constructed graph nodes:

$$h_j^t, c_j^t = LSTM\left(e\left(o_j^t\right), h_j^t, c_j^t\right) \tag{11}$$

where $e(\cdot)$ is a fully connected layer representing the observed encoder. h_j^t is encoded as a message:

$$m_j^t = e\left(h_j^t\right) \tag{12}$$

where m_j^t is the aggregated information of the neighborhood agents observed by agent j at time step t. The input encoding information M^t is passed to the GAT encoder and hard attention mechanism, where the hard attention mechanism consists of MLP and Gumbel Softmax function. Finally, the output adjacency matrix $\left\{G^t\right\}_1^N$ is used to determine which agents in the neighborhood have an influence on the current agent. The GAT encoder helps to efficiently encode the agent's local information, which is expressed as:

$$\{M^t\}_1^N = f_{Sched}\left(m_1^t, \cdots, m_N^t\right) \tag{13}$$

Additionally, we take the form of the same attention mechanism as GAT [25], expressed as:

$$\alpha_{ij}^S = \frac{\exp\left(LeakyReLU\left(a_S^T\left[W_S m_i^t || W_S m_j^t\right]\right)\right)}{\sum_{k\in N_j^t \cup \{j\}} \exp\left(LeakyReLU\left(a_S^T\left[W_S m_j^t || W_S m_k^t\right]\right)\right)} \tag{14}$$

where $LeakyReLU(\cdot)$ is the activation function, $a_S \in R^D$ is the weight vector, $N_j^t \cup \{j\}$ represents the central agent j and its observable neighborhood agent set, and $W_S \in R^{D\times D}$ is the weight matrix. The node feature of agent j is expressed as:

$$e_j^t = ELU\left(\sum_{i\in N_j^t \cup j} \alpha_{ij}^S W_S m_i^t\right) \tag{15}$$

where $ELU(\cdot)$ is an exponential linear unit function. Connecting the features of each node in pairs: $E_{i,j}^t = \left(e_i^t || e_j^t\right)$, we can get a matrix $E^t \in R^{N\times N_j \times 2D}$, where $E_{i,j}^t$ represents the relevant features of agent j. Taking E^t as the input of MLP which is followed by a Gumbel Softmax function, the connected vector G_j^t can be obtained. The connected vector G_j^t consists of elements g_{ij}, where i represents the neighbors of the central agent j. The element $g_{ij}^t = 1$ in the adjacency matrix indicates that the action of the agent i will have an impact on the agent j. Conversely, $g_{ij}^t = 0$ means that the agent's actions have no effect on the agent j.

4.2 | Mean Field Module

This Graph-Attention method selects important M_j agents from the neighbors N_j of agent j, and compute the average of the actions of the choosed neighbor agents:

$$\tilde{a}_t^j = \frac{1}{M_j} \sum_{k\in N_j} a_t^k \cdot G_j^t, \quad a_t^k \sim \pi^k\left(\cdot \mid s_t, \tilde{a}_t^k\right) \tag{16}$$

where $\cdot$ is the element-wise multiplication.

In the above formula, a^k represents the important neighborhood agent for agent j. Then the Q–value of each agent is shown in Eq. 17. Note that the Q–value here is a partially observable Q–value.

$$Q_{\text{GAMF}}^j\left(s_t^j, a_t^j, \tilde{a}_t^j\right) = (1-\alpha)Q_{\text{GAMF}}^j\left(s_t^j, a_t^j, \tilde{a}_t^j\right) + \alpha\left[r_t^j + \gamma v\left(s_{t+1}^j\right)\right] \tag{17}$$

where the value function v^j is expressed as

$$v^j\left(s_{t+1}^j\right) = \sum_{a_{t+1}^j} \pi^j\left(a_{t+1}^j \mid s_{t+1}^j, \tilde{a}_t^j\right) Q_{\text{GAMF}}^j\left(s_{t+1}^j, a_{t+1}^j, \tilde{a}_t^j\right) \tag{18}$$

According to the above graph attention mechanism, more important neighborhood agents are obtained. The new average action $\tilde{a}_t^j$ is calculated by Eq.16, and then the strategy π_t^j of agent j is updated by the following formula:

$$\pi^j\left(a_t^j \mid s_t^j, \tilde{a}_{t-1}^j\right) = \frac{\exp\left(-\beta Q_{\text{GAMF}}^j\left(s_t^j, a_t^j, \tilde{a}_{t-1}^j\right)\right)}{\sum_{a_t^{j'} \in A^j} \exp\left(-\beta Q_{\text{GAMF}}^j\left(s_t^j, a_t^{j'}, \tilde{a}_{t-1}^j\right)\right)} \tag{19}$$

4.3 | Theoretical Proof

This subsection is devoted to proving that the setting of GAMFQ is close to the Nash equilibrium. Subramanian et al. [23] showed that in partially observable cases, the fixed observation radius (FOR) setting is close to a Nash equilibrium, where the mean action of each agent's neighborhood agents is approximated by a dirichlet distribution. First, we state some assumptions, which are the same as literature[23], and are followed by all the theorems and analyses below.

Assumption 1. For any i and j, there is $\lim_{t\to\infty} \tau_j^i(t) = \infty$. $w.p.1$.

This assumption guarantees a probability of 1 that old information is eventually discarded.

Assumption 2. Suppose some measurability conditions are as follow: (1) $x(0)$ is $\mathcal{F}(0)$-measurable. (2) For each i,j and t, $w_i(t)$ is $\mathcal{F}(t+1)$-measurable. (3) For each i, j and t, $\alpha_i(t)$ and $\tau_j^i(t)$ are $\mathcal{F}(t)$-measurable. (4) For each i and t, satisfy B $\left[w_i(t)|\mathcal{F}(t)\right] = 0$. (5) B $\left[w_i^2(t)|\mathcal{F}(t)\right] \leq A + B \max_j \max_{\tau\leq t} \left|x_j(\tau)\right|^2$, where A and B are deterministic constants.

Assumption 3. The learning rates satisfy $0 \leq \alpha_i(t) < 1$.

Assumption 4. Suppose some conditions for the F mapping are as follows: (1) If $x \leq y$, then $F(x) \leq F(y)$, that is, F is monotonic; (2) F is continuous; (3) When $t \to \infty$, F is limited to the interval $[x^* - D, x^* + D]$, where x^* is some arbitrary point; (4) If $e \in \mathcal{R}^n$ is a vector that satisfies all components equal to 1, then $F(x) - pe \leq F(x+pe) \leq F(x+pe) + pe$, where p is a positive scalar.

Assumption 5. Each action-value pair can be accessed indefinitely, and the reward is limited.

Assumption 6. Under the limit $t \to \infty$ of infinite exploration, the agent's policy is greedy.

This assumption ensures that the agent is rational.

Assumption 7. In each stage of a stochastic game, a Nash equilibrium can be regarded as a global optimum or saddle point.

Based on these assumptions, Subramanian et al. [23] give the following lemma.

Lemma 1. [23] When the Q-function is updated using the partially observable update rule in Eq.2, and assumptions 3, 5, and 7 hold, the following holds for $t \to \infty$:

$$|Q_*(s_t, a_t) - Q_{POMF}(s_t, a_t, \tilde{a}_t)| \leq 2D \tag{20}$$

where Q_* is the Nash Q-value, Q_{POMF} is the partially observable mean-field Q-function, and D is the bound of the F map. The probability that the above formula holds is at least δ^{L-1} , where $L = |A|$.

In our GAMFQ setting, for partially observable neighborhood agents, we choose to select a limited number of important agents by using graph attention, and then update the POMF Q function. The following theorem proves that the setting of GAMFQ is close to Nash equilibrium.

Theorem 1. The distance between the MFQ (globally observable) mean action $\bar{a}$ and the GAMFQ (partially observable) mean action $\tilde{a}$ satisfies the following formula:

$$\left|\tilde{a}_t^j - \bar{a}_t^j\right| \leq \sqrt{\frac{1}{2N_j}\log\frac{2}{\delta}} \tag{21}$$

When $t \to \infty$, the probability >= δ, where N_j is the number of observed neighbor agents, $\tilde{a}$ is the partially observable mean action obtained by graph attention in Eq. 16, $\bar{a}$ is the globally observable mean action in Eq. 4.

Assuming that each agent is globally observable, the mean of important agents selected by graph attention is close to the true underlying global observable $\bar{a}$. Since the GAMF Q-function is updated by taking finite samples through graph attention, the empirical mean is $\tilde{a}$.

Theorem 2. If the Q-function is Lipschitz continuous with respect to the mean action, i.e. M is constant, then the MF Q-function Q_{MF} and GAMF Q-function Q_{GAMF} satisfy the following relation:

$$\left|Q_{GAMF}\left(s_t, a_t, \tilde{a}_{t-1}\right) - Q_{MF}\left(s_t, a_t, \bar{a}_{t-1}\right)\right| \leq M \times L \times \log\frac{2}{\delta} \times \frac{1}{2N_j} \tag{22}$$

When the limit $t \to \infty$, the probability is $\geq (\delta)^{L-1}$, where $L = |A|$, A is the action space of the agent.

In the proof of theorem 2, first consider a Q-function that is Lipschitz continuous for all $\bar{a}$ and $\tilde{a}$.According to theorem 1, the above formula can further deduce the result of theorem 2. The total number of components is equal to the action space L. The bound of theorem 1 is probability >= δ, and since there are L random variables, the probability of theorem 2 is at least $(\delta)^{L-1}$. When the first $L-1$ random variable is fixed, the deterministic last $\bar{a}$ component satisfies the relationship that the sum of the individual components is 1. Since each agent's action is represented by a one-hot encoding, the $\tilde{a}'$ component of GAMFQ also satisfies the relationship that the sum of the individual components is 1, and the component of the agent's average action does not change due to the application of graph attention. The proof of theorem 2 ends.

Theorem 3. A stochastic process in form $x_i(t+1) = x_i(t) + \alpha_i(t)\left(F_i\left(x^i(t)\right) - x_i(t) + w_i(t)\right)$ remains bounded in the range $[x_* - 2D, x_* + 2D]$ on limit $t \to \infty$ if assumptions 1,2,3 and 4 are satisfied, and are guaranteed not to diverge to infinity.Where D is the boundary of the F map in assumption 4(4).

This theorem can be proved in terms of Tsitsiklis[24] and by extension. The result of theorem 3 can then be used to derive theorem 4.

Theorem 4. When the Q-function is updated using the partially observable update rule in Eq.17, and assumptions 3, 5, and 7 hold, the following holds for $t \to \infty$:

$$|Q_*(s_t, a_t) - Q_{GAMF}(s_t, a_t, \tilde{a}_t)| \leq 2D \tag{23}$$

where Q_* is the Nash Q-value, Q_{GAMF} is the partially observable mean-field Q-function, and D is the bound of the F map. The probability that the above formula holds is at least δ^{L-1} , where $L = |A|$.

Theorem 4 shows that the GAMFQ update is very close to the Nash equilibrium at the limit $t \to \infty$, i.e. reaching a plateau for stochastic policies. Therefore, the strategy of Eq.19 is approximately close to this plateau. Theorem 4 is an application of theorem 3, using assumptions 3, 5 and 7 .However, in MARL, reaching a Nash equilibrium is not optimal, but only a fixed-point guarantee. Therefore, to achieve better performance, each selfish agent will still tend to pick a limited number of samples. To balance theory and performance when selecting agents from the neighborhood, an appropriate number of agents (more efficient agents) need to be used for better multi-agent system performance. This paper uses the graph attention structure to filter out more important proxies, which can better approximate the Nash equilibrium.

4.4 | Algorithm

The implementation of GAMFQ follows the related work of the previous POMFQ [23], the difference is that the graph attention structure is used to select the neighborhood agents that are more important to the central agent when updating the average action. Algorithm 1 gives the pseudocode of the GAMFQ algorithm. It obtains effective neighbor agents by continuously updating the adjacency matrix G_j^t to update the agent's strategy.

Algorithm 1 Partially Observable Mean Field MARL Based on Graph–Attention

```
Initialize the weights of Q-function $Q_{\phi^j}$, $Q_{\phi_-^j}$, replay buffer $B$, GAT encoder, MLP layers and mean action $\bar{a}^j$ for each agent $j \in 1, \ldots, N$.
for $episode = 1, 2, \ldots, E$ do
    for $t \leq T$ and not terminal do
        For each agent $j$, calculate the hidden state $h_j^t$ according to Eq.11, and encode $h_j^t$ as a message $m_j^t$ (Eq.12).
        For each agent $j$, sample $a^j$ fron policy induced by $Q_{\phi^j}$(Eq.19).
        For each agent $j$, pass the encoded information $m_j^t$ to the GAT encoder and hard attention mechanism to output the adjacency matrix $G_j^t$.
        For each agent $j$, calculate the new neighborhood agent mean action $\bar{a}^j$ by Eq.16.
        Receive the full state of environment $s_t$, action $a = \left[a^1, \ldots, a^N\right]$, reward $\left[r = r^1, \ldots, r^N\right]$, and the next state $s' = \left[s^1, \ldots, s^N\right]$.
        Store transition $\langle s, a, r, s', \bar{a} \rangle$ in $B$, where $\bar{a} = \left[\bar{a}^1, \ldots, \bar{a}^N\right]$ is the mean action.
    end for
    for $j = 1, \ldots, N$ do
        Sample a minibatch of K experiences $\langle s, a, r, s', \bar{a} \rangle$ from replay buffer $B$.
        Set $y^j = r^j + \gamma v_\phi \left(s'\right)$ according to Eq.18.
        minimize the loss $L\left(\phi^j\right) = \left(y^j - Q_{\psi_j}\left(s', a^j, \bar{a}^j\right)\right)^2$ to update Q network.
    end for
    For each agent $j$, update params of target network :$\phi^j \leftarrow \tau\phi^j + (1-\tau)\phi^j$.
end for
```

5 | EXPERIMENTS

In this section, we describe three different tasks based on the MAgent framework and give some experimental setup and training details for evaluating the GAMFQ performance.

5.1 | Environments and Tasks

Subramanian et al. [23] designed three different cooperative-competitive strategies in the MAgent framework [36] as experimental environments, and our experiments adopt the same environments. In these three tasks, the map size is set to 28*28, where the observation range of each agent is 6 units. The state space is the concatenation of the feature information of other agents within each agent's field of view, including location, health, and grouping information. The action space includes 13 move actions and 8 attack actions. In addition, each agent is required to handle at most 20 other agents that are closest. We will evaluate against these three tasks:

- **Multibattle environment:** There are two groups of agents fighting each other, each containing 25 agents. The agent gets -0.005 points for each move, -0.1 points for attacking an empty area, 200 points for killing an enemy agent, and 0.2 points for a successful attack. Each agent is 2*2 in size, has a maximum health of 10 units, and a speed of 2 units. After the battle, the team with the most surviving agents wins. If both teams have the same number of surviving agents, the team with the highest reward wins. The reward for each team is the sum of the rewards for the individual agents in the team.

- **Battle-Gathering environment:** There is a uniform distribution of food in the environment, each agent can observe the location of all the food. In addition to attacking the enemy to get rewards, each agent can also eat food to get rewards. Agents get 5 points for attacking enemy agents, and the rest of the reward settings are the same as the Multibattle environment.

- **Predator-Prey environment:** There are 40 predators and 20 prey, where each predator is a square grid of size 2*2 with a maximum health of 10 units and a speed of 2 units. Prey is a 1*1 square with a maximum health of 2 units and a speed of 2.5 units. To win the game, the predator must kill more prey, and the prey must find a way to escape. In addition, predators and prey have different reward functions, predators get -0.3 points for attacking space, 1 point for successfully attacking prey, 100 points for killing prey, -1 point for attacked prey, and 0.5 points for dying. Unlike the Multibattle environment, when the round ends for a fairer duel, if the two teams have the same number of surviving agents, it is judged as a draw.

5.2 | Evaluation

We consider four algorithms for the above three games: MFQ, MFAC [31], POMFQ(FOR) and GAMFQ, where MFQ and MFAC are baselines and POMFQ(FOR) [23] is the state-of-the-art algorithm.

The original baselines MFQ and MFAC were proposed by Yang et al. [31] based on global observability, and the idea was to approximate the influence of the neighborhood agents on the central agent as their average actions, thereby updating the actions of the neighborhood agents. We fix the observation radius of each agent in the baseline MFQ and MFAC and apply it to a partially observable environment, where neighbor agents are agents within a fixed range. The POMFQ(FOR) algorithm introduces noise in the mean action parameters to encourage exploration, uses Bayesian inference to update the Dirichlet distribution, and samples 100 samples from the Dirichlet distribution to estimate partially observable mean field actions. The GAMFQ algorithm judges the effectiveness of neighborhood agents within a fixed range through the graph attention mechanism, selects more important neighborhood agents, and updates the average action by averaging the actions of these agents.

5.3 | Hyperparameters

In the three tasks, each algorithm was trained for 2000 epochs in the training phase, generating two sets of A and B sets of models. In the test phase, 1000 rounds of confrontation were conducted, of which the first 500 rounds were the first group A of the first algorithm against the second group B of the second algorithm, and the last 500 groups were the opposite. The hyperparameters of MFQ, MFAC, POMFQ(FOR) and GAMFQ are basically the same. Table 1 lists the hyperparameters during training of the four algorithms, and the remaining parameters can be seen in [23].

6 | RESULTS AND DISCUSSION

In this section, we evaluate the performance of GMAFQ in three different environments, including Multibattle, Battle-Gathering, and Predator-Prey. We benchmark against two algorithms, MFQ and MFAC, and compare with the state-of-the-art POMFQ

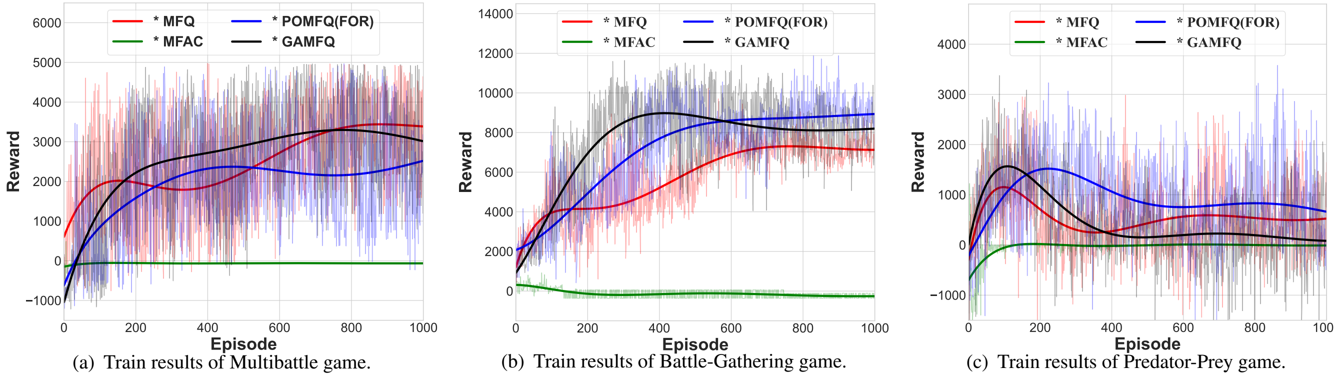
TABLE 1 Hyperparameters for four algorithms training.

Parameter	Value	Description
α	10^{-4}	learning rate
β	decays linearly from 1 to 0	exploration rate
γ	0.95	discount rate
B	1024	replay buffer
h	64	the hidden layer size in GAT
K	64	mini-batch
$temperature$	0.1	the soft-max layer temperature of the actor in MFAC

(FOR). We implement our method and comparative methods on three different tasks. Note that we only used 50 agents in our experiments and did not test more agents, this is because the proportion of other agents that each agent can see is more important than the absolute number.

6.1 | Reward

Figure 3 shows how the reward changes as the number of iterations increases during training. We plot the reward changes for the four algorithms in different game environments during the first 1000 iterations. Since each algorithm is self-training which results in a large change in the reward of the algorithm, we use the least squares method to fit the reward change graph. In Figure 3, the solid black line represents the reward change graph of the GAMFQ algorithm. From Figure 3 (a), (b) and (c), it can be seen that the reward of the GAMFQ algorithm can increase rapidly, indicating that the GAMFQ algorithm can converge rapidly in the early stage, and the convergence performance is better than the other three algorithms.



(a) Train results of Multibattle game.

(b) Train results of Battle-Gathering game.

(c) Train results of Predator-Prey game.

FIGURE 3 Train results of three games. The reward curve for each algorithm is fitted by the least squares method.

6.2 | Elo Calculation

We use ELO Score [11] to evaluate the performance of the two groups of agents, the advantage of which is that it takes into account the strength gap between the opponents themselves. ELO ratings are commonly used in chess to evaluate one-on-one situations, and this approach can similarly be extended to N-versus-N situations. For the algorithm proposed in the paper, we record the total rewards of the two teams of agents during each algorithm confrontation, which are R_1 and R_2, respectively. Then the expected win rates of the two groups of agents are:

$$E_1 = \frac{1}{1 + 10^{(R_2 - R_1)/400}}, E_2 = \frac{1}{1 + 10^{(R_1 - R_2)/400}} \tag{24}$$

where $E_1 + E_2 = 1$. By analyzing the actual and predicted winning rates of the two groups of agents, the new ELO score of each team after the game ends can be obtained:

$$R_1{}' = R_1 + K(S_1 - E_1), R_2{}' = R_2 + K(S_2 - E_2) \tag{25}$$

where R_1 represents the actual winning or losing value, 1 means the team wins, 0.5 means the two teams are tied, and 0 means the team loses. K is represented as a floating coefficient. To create a gap between agents, we set K to 32. For each match, we faced off 500 times and calculated the average ELO value for all matches.

TABLE 2 The ELO Score of four algorithms in Multibattle environment.

Task	Algorithm1	Algorithm2	ELO Score1	ELO Score2
GAMFQ vs POMFQ(FOR)	GAMFQ-1	POMFQ(FOR)-2	**3579**	820
	GAMFQ-2	POMFQ(FOR)-1	2696	**2838**
GAMFQ vs MFQ	GAMFQ-1	MFQ-2	**2098**	1508
	GAMFQ-2	MFQ-1	**2535**	1695
GAMFQ vs MFAC	GAMFQ-1	MFAC-2	**1350**	-49
	GAMFQ-2	MFAC-1	-856	**-78**
POMFQ(FOR) vs MFQ	POMFQ(FOR)-1	MFQ-2	**3145**	2577
	POMFQ(FOR)-2	MFQ-1	2569	**2857**
POMFQ(FOR) vs MFAC	POMFQ(FOR)-1	MFAC-2	-205	**-64**
	POMFQ(FOR)-2	MFAC-1	**826**	-42
MFQ vs MFAC	MFQ-1	MFAC-2	-142	**-49**
	MFQ-2	MFAC-1	**610**	-46

TABLE 3 The ELO Score of four algorithms in Battle-Gathering environment.

Task	Algorithm1	Algorithm2	ELO Score1	ELO Score2
GAMFQ vs POMFQ(FOR)	GAMFQ-1	POMFQ(FOR)-2	7770	**8931**
	GAMFQ-2	POMFQ(FOR)-1	8293	**9310**
GAMFQ vs MFQ	GAMFQ-1	MFQ-2	6374	**10870**
	GAMFQ-2	MFQ-1	**8510**	8313
GAMFQ vs MFAC	GAMFQ-1	MFAC-2	**5525**	10
	GAMFQ-2	MFAC-1	**10751**	-31
POMFQ(FOR) vs MFQ	POMFQ(FOR)-1	MFQ-2	8526	**8760**
	POMFQ(FOR)-2	MFQ-1	**8632**	8227
POMFQ(FOR) vs MFAC	POMFQ(FOR)-1	MFAC-2	**12722**	0
	POMFQ(FOR)-2	MFAC-1	**12171**	-88
MFQ vs MFAC	MFQ-1	MFAC-2	**12649**	49
	MFQ-2	MFAC-1	**13788**	-48

Table 2, 3, 4 shows the ELO scores of the four algorithms on the three tasks. It can be seen from Table 2 that in Multibattle environment, the GAMFQ algorithm has the highest ELO score of 3579, which is significantly better than the other three algorithms. As shown in Table 3, in Battle-Gathering environment, the ELO score of the MFQ algorithm is the highest, and the ELO score of the GAMFQ algorithm is average. This is because the collection environment contains food, and some algorithms tend to eat food to get rewards quickly, rather than attacking enemy agents. However, the final game winning or losing decision is made by comparing the number of remaining agents between the two teams of agents. As shown in Table 4, in Predator-Prey environment, the ELO score of the GAMFQ algorithm has the highest ELO score of 860, which is significantly better than the other three algorithms. From the experimental results in the three environments, we can summarize that ELO score of the GAMFQ algorithm is better than other three algorithms, showing better performance.

TABLE 4 The ELO Score of four algorithms in Predator-Prey environment.

Task	**Algorithm1**	**Algorithm2**	**ELO Score1**	**ELO Score2**
GAMFQ vs POMFQ(FOR)	GAMFQ-1	POMFQ(FOR)-2	**421**	-32
	GAMFQ-2	POMFQ(FOR)-1	**16**	7
GAMFQ vs MFQ	GAMFQ-1	MFQ-2	**714**	-27
	GAMFQ-2	MFQ-1	**-15**	-94
GAMFQ vs MFAC	GAMFQ-1	MFAC-2	**860**	-28
	GAMFQ-2	MFAC-1	16	16
POMFQ(FOR) vs MFQ	POMFQ(FOR)-1	MFQ-2	**66**	18
	POMFQ(FOR)-2	MFQ-1	13	**24**
POMFQ(FOR) vs MFAC	POMFQ(FOR)-1	MFAC-2	**16**	-16
	POMFQ(FOR)-2	MFAC-1	**47**	16
MFQ vs MFAC	MFQ-1	MFAC-2	**16**	-16
	MFQ-2	MFAC-1	**174**	17

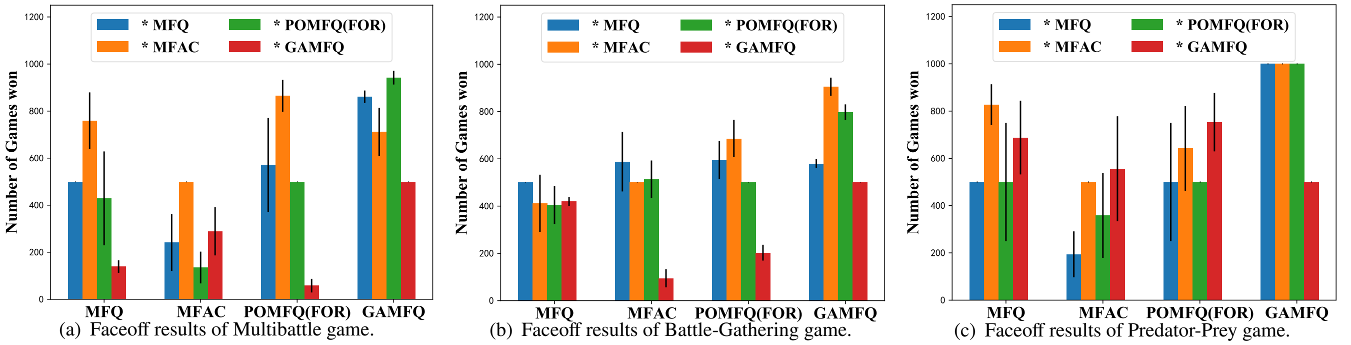
6.3 | Results

Figure 4 shows the face-off results of the four algorithms in the three tasks. Figure 4(a) shows the faceoff results of Multibattle game. The different colored bars for each algorithm represent the results of an algorithm versus others. We do not conduct adversarial experiments between the same algorithms because we consider that the adversarial properties of the same algorithms are equal. The vertical lines in the bar graph represent the standard deviation of wins for groups A and B over 1,000 face-offs. Figure 4(a) shows GAMFQ against three other algorithms, all with a win rate above 0.7.

Figure 4(b) shows the faceoff results of Battle-Gathering game. In addition to getting rewards for killing enemies, agents can also get rewards from food. It can be seen that MFQ loses to all other algorithms, MFAC and POMFQ (FOR) perform in general, and our GAMFQ is clearly ahead of other algorithms.

Figure 4(c) shows thwe faceoff results of Predator-Prey game.The standard deviation of this game is significantly higher than the previous two games, due to the fact that both groups A and B are trying to beat each other in the environment. It can be seen that the GAMFQ algorithm is significantly better than other three algorithms, reaching a winning rate of 1.0.

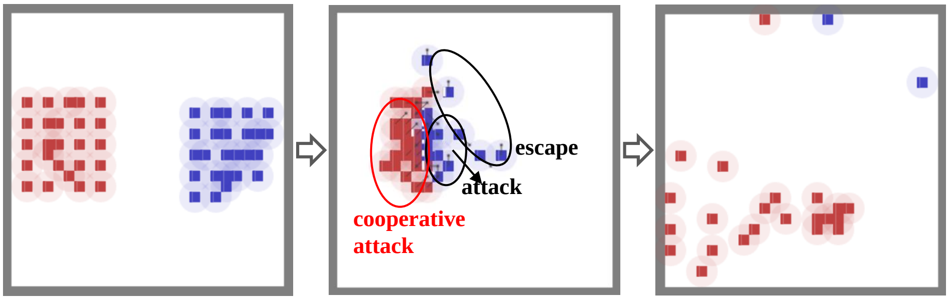
Experiments in the above three multi-agent combat environments show that GAMFQ can show good performance over MFQ, MFAC and POMFQ(FOR).



(a) Faceoff results of Multibattle game. (b) Faceoff results of Battle-Gathering game. (c) Faceoff results of Predator-Prey game.

FIGURE 4 Faceoff results of three games. The * in the legend indicates the enemy. For example, the first blue bar in the bar graph corresponding to the GAMFQ algorithm is the result of the confrontation between GAMFQ and MFQ, and we do not conduct confrontation experiments between the same algorithms.

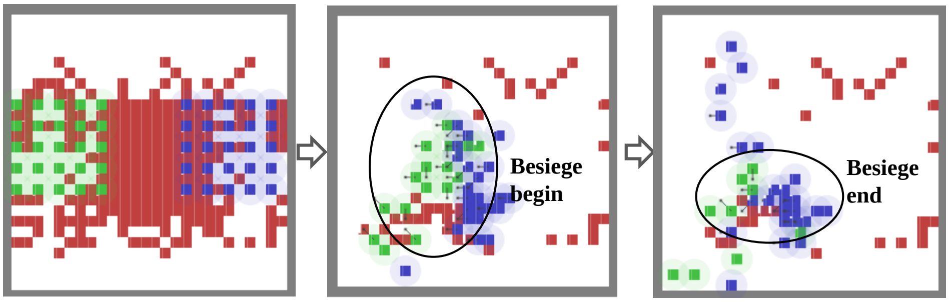
6.4 | Visualization



(a) Multibattle begin (b) Multibattle process (c) Multibattle end

🟦 **GAMFQ alive：22** 🟩 **POMFQ(FOR) alive：2**

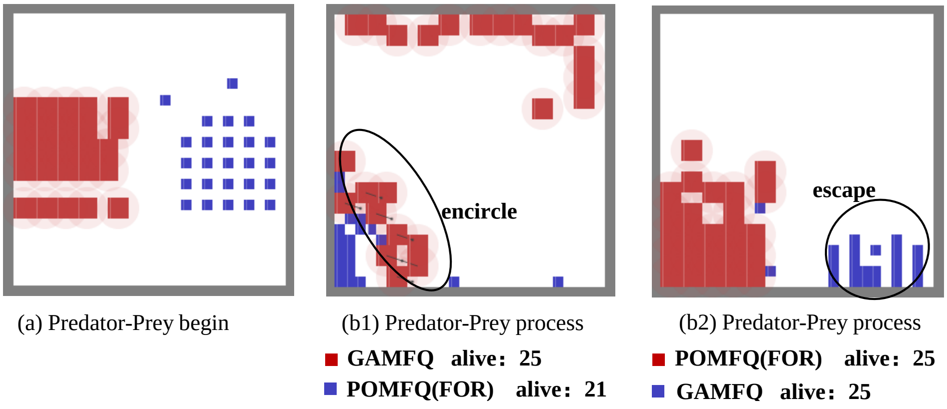
FIGURE 5 Visualization of the standoff between GAMFQ and POMFQ (FOR) in a Multibattle game.



(a) Battle-Gathering begin (b) Battle-Gathering process (c) Battle-Gathering end

🟥 **food** 🟦 **GAMFQ alive：24** 🟩 **POMFQ(FOR) alive：9**

FIGURE 6 Visualization of the standoff between GAMFQ and POMFQ (FOR) in a Battle-Gathering game.



(a) Predator-Prey begin

(b1) Predator-Prey process

(b2) Predator-Prey process

FIGURE 7 Visualization of the standoff between GAMFQ and POMFQ (FOR) in a Predator-Prey game.

To visualize the effectiveness of the GAMFQ algorithm, we visualize the confrontation between GAMFQ and POMFQ (FOR) in a Multibattle environment, as shown in Figure 5, where the red side is GMAFQ and the blue side is POMFQ (FOR). It can be seen from the confrontation process that for the GAMFQ algorithm, when an agent decides to attack, the surrounding agents will also decide to attack under its influence, forming a good cooperation mechanism. On the contrary, for the POMFQ (FOR) algorithm, some blue-side agents are chosen to attack, some are chosen to escape, and no common fighting mechanism was formed. Similarly, in the Battle-Gathering environment of Figure 6, GAMFQ can learn the surrounding mechanism well. In the Predator-Prey environment of Figure 7, when GAMFQ acts as a predator, the technique of surrounding the prey POMFQ (FOR) can be learned. On the contrary, when POMFQ (FOR) acted as a predator, it failed to catch the prey GMAFQ.

6.5 | Ablation study

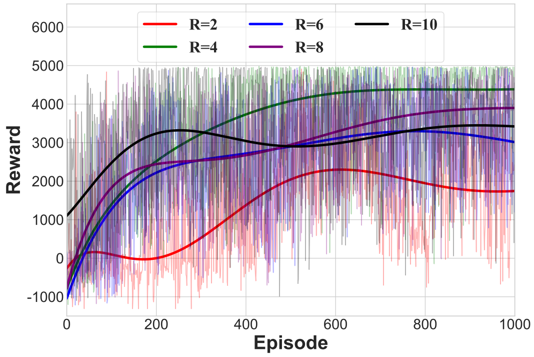


FIGURE 8 Ablation study. R represents the observation radius of the agent.

Figure 8 is an ablation study that investigates the performance of the GAMFQ algorithm for different observation radius in a Multibattle environment. where the solid line represents the least squares fit of the reward change. It can be seen from the figure that when the number of training is small, the performance of the algorithm is better as the observation distance increases. But with the increase of training times, when R=4, the performance of the algorithm is the best, so the appropriate observation distance can achieve better performance. What is more important in this paper is to explore the effect of the ratio of observable distance to the number of agents on the performance of the algorithm, so there is no experiment with more agents.

7 | CONCLUSION

In this paper, we proposed a new multi-agent reinforcement learning algorithm, Graph Attention-based Partially Observable Mean Reinforcement Learning (GAMFQ), to address the problem of large-scale partially observable multi-agent environments. Although existing methods are close to Nash equilibrium, they do not take into account the direct correlation of agents. Based on the correlation between agents, GAMFQ uses a Graph-Attention module to describe how each agent is affected by the actions of other agents at each time step. Experimental results on three challenging tasks in the MAgents framework illustrate that, our proposed method outperforms baselines in all these games and outperforms the state-of-the-art partially observable mean-field reinforcement learning algorithms. In the future, we will further explore the correlation between agents to extend to more common cooperation scenarios.

Conflict of interest

The authors declare no potential conflict of interests.

Article Description

The expanded version of this article is published in Drones 2023, 7(7), 476, with a DOI of https://doi.org/10.3390/drones7070476.

References

[1] A. T. Azar et al., *Drone deep reinforcement learning: A review*, Electronics **10** (2021), no. 9, 999.

[2] W. Boehmer, V. Kurin, and S. Whiteson, *Deep coordination graphs*, *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, *Proceedings of Machine Learning Research*, vol. 119, PMLR, 980–991.

[3] Q. Cai, Z. Yang, and Z. Wang, *Reinforcement learning from partial observation: Linear function approximation with provable sample efficiency*, *International Conference on Machine Learning*, PMLR, 2485–2522.

[4] J. Fan et al., *A theoretical analysis of deep q-learning*, *Learning for Dynamics and Control*, PMLR, 486–489.

[5] B. Fang et al., *Large-scale multi-agent reinforcement learning based on weighted mean field*, *Cognitive Systems and Signal Processing - 5th International Conference, ICCSIP 2020, Zhuhai, China, December 25-27, 2020, Revised Selected Papers*, *Communications in Computer and Information Science*, vol. 1397, Springer, 309–316.

[6] H. Gu et al., *Mean-field multi-agent reinforcement learning: A decentralized network approach*, arXiv preprint arXiv:2108.02731 (2021).

[7] J. Gu et al., *A metaverse-based teaching building evacuation training system with deep reinforcement learning*, IEEE Transactions on Systems, Man, and Cybernetics: Systems (2023).

[8] Q. Hao, *Very large scale multi-agent reinforcement learning with graph attention mean field*, https://openreview.net/forum?id=MdiVU9lMmVS (2023).

[9] K. He, P. Doshi, and B. Banerjee, *Reinforcement learning in many-agent settings under partial observability*, *The 38th Conference on Uncertainty in Artificial Intelligence*.

[10] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, *A survey and critique of multiagent deep reinforcement learning*, Autonomous Agents and Multi-Agent Systems **6** (2019), no. 33, 750–797.

[11] M. Jaderberg et al., *Human-level performance in first-person multiplayer games with population-based deep reinforcement learning*, ArXiv **abs/1807.01281** (2018).

[12] M. Laurière et al., *Learning mean field games: A survey*, arXiv preprint arXiv:2205.12944 (2022).

[13] S. Li et al., *Deep implicit coordination graphs for multi-agent reinforcement learning*, *AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021*, ACM, 764–772.

[14] M. L. Littman, *Markov games as a framework for multi-agent reinforcement learning*, *Machine learning proceedings 1994*, Elsevier, 1994. 157–163.

[15] R. Lowe et al., *Multi-agent actor-critic for mixed cooperative-competitive environments*, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 6379–6390.

[16] A. Mahajan et al., *Maven: Multi-agent variational exploration*, Advances in Neural Information Processing Systems **32** (2019), 7611–7622.

[17] Y. Niu, R. R. Paleja, and M. C. Gombolay, *Multi-agent graph-attention communication and teaming*, *AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021*, ACM, 964–973.

[18] F. A. Oliehoek and C. Amato, *A concise introduction to decentralized POMDPs*, Springer, 2016.

[19] T. Rashid et al., *QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning*, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, *Proceedings of Machine Learning Research*, vol. 80, PMLR, 4292–4301.

[20] J. Ruan et al., *Gcs: Graph-based coordination strategy for multi-agent reinforcement learning*, arXiv preprint arXiv:2201.06257 (2022).

[21] L. M. Schmidt et al., *An introduction to multi-agent reinforcement learning and review of its application to autonomous mobility*, arXiv preprint arXiv:2203.07676 (2022).

[22] H. Shi et al., *Marl sim2real transfer: Merging physical reality with digital virtuality in metaverse*, IEEE Transactions on Systems, Man, and Cybernetics: Systems (2022).

[23] S. G. Subramanian et al., *Partially observable mean field reinforcement learning*, *AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021*, ACM, 537–545.

[24] J. N. Tsitsiklis, *Asynchronous stochastic approximation and q-learning*, Machine learning **16** (1994), no. 3, 185–202.

[25] P. Velickovic et al., *Graph attention networks*, stat **1050** (2017), 20.

[26] O. Vinyals et al., *Grandmaster level in starcraft ii using multi-agent reinforcement learning*, Nature **575** (2019), no. 7782, 350–354.

[27] L. Wang et al., *Neural policy gradient methods: Global optimality and rates of convergence*, arXiv preprint arXiv:1909.01150 (2019).

[28] T. Wu et al., *Weighted mean-field multi-agent reinforcement learning via reward attribution decomposition*, *International Conference on Database Systems for Advanced Applications*, Springer, 301–316.

[29] Z. Wu et al., *A comprehensive survey on graph neural networks*, IEEE transactions on neural networks and learning systems **32** (2020), no. 1, 4–24.

[30] Q. Xie et al., *Learning while playing in mean-field games: Convergence and optimality*, *International Conference on Machine Learning*, PMLR, 11436–11447.

[31] Y. Yang et al., *Mean field multi-agent reinforcement learning*, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, *Proceedings of Machine Learning Research*, vol. 80, PMLR, 5567–5576.

[32] H. Zhang et al., *H2gnn: Hierarchical-hops graph neural networks for multi-robot exploration in unknown environments*, IEEE Robotics and Automation Letters **7** (2022), no. 2, 3435–3442.

[33] K. Zhang, Z. Yang, and T. Başar, *Multi-agent reinforcement learning: A selective overview of theories and algorithms*, Handbook of Reinforcement Learning and Control (2021), 321–384.

[34] T. Zhang et al., *MFVFD: A multi-agent q-learning approach to cooperative and non-cooperative tasks*, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, ijcai.org, 500–506, .

[35] Y. Zhang et al., *Coordination between individual agents in multi-agent reinforcement learning*, *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, AAAI Press, 11387–11394.

[36] L. Zheng et al., *Magent: A many-agent reinforcement learning platform for artificial collective intelligence*, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, AAAI Press, 8222–8223.

[37] S. Zhou et al., *Multi-agent mean field predict reinforcement learning*, *2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA)*, IEEE, 625–629.

[38] Z. Zhou and G. Liu, *Romfac: A robust mean-field actor-critic reinforcement learning against adversarial perturbations on states*, arXiv preprint arXiv:2205.07229 (2022).

How to cite this article: Min Yang, Guanjun Liu, Ziyuan Zhou, Jiacun Wang (2023), Partially Observable Mean Field Multi-Agent Reinforcement Learning Based on Graph Attention Network for UAV Swarms, *Drones*, *2023;7(7):476*. https://www.mdpi.com/2504-446X/7/7/476