drones

Article

Partially Observable Mean Field Multi-Agent Reinforcement Learning Based on Graph Attention Network for UAV Swarms

Min Yang [1], Guanjun Liu [1,*], Ziyuan Zhou [1] and Jiacun Wang [2]

[1] Department of Computer Science and Technology, Tongji University, Shanghai 201804, China; minyang@tongji.edu.cn (M.Y.); ziyuanzhou@tongji.edu.cn (Z.Z.)
[2] Department of Computer Science and Software Engineering, Monmouth University, West Long Branch, NJ 07764, USA; jwang@monmouth.edu
* Correspondence: liuguanjun@tongji.edu.cn

Abstract: Multiple unmanned aerial vehicles (Multi-UAV) systems have recently demonstrated significant advantages in some real-world scenarios, but the limited communication range of UAVs poses great challenges to multi-UAV collaborative decision-making. By constructing the multi-UAV cooperation problem as a multi-agent system (MAS), the cooperative decision-making among UAVs can be realized by using multi-agent reinforcement learning (MARL). Following this paradigm, this work focuses on developing partially observable MARL models that capture important information from local observations in order to select effective actions. Previous related studies employ either probability distributions or weighted mean field to update the average actions of neighborhood agents. However, they do not fully consider the feature information of surrounding neighbors, resulting in a local optimum often. In this paper, we propose a novel partially multi-agent reinforcement learning algorithm to remedy this flaw, which is based on graph attention network and partially observable mean field and is named as the GPMF algorithm for short. GPMF 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. The graph attention module consists of a graph attention encoder and a differentiable attention mechanism, outputting 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. Aiming at the typical task scenario of large-scale multi-UAV cooperative roundup, the proposed algorithm is evaluated based on the MAgent framework. Experimental results show that GPMF outperforms baselines including state-of-the-art partially observable mean field reinforcement learning algorithms, providing technical support for large-scale multi-UAV coordination and confrontation tasks in communication-constrained environments.

Keywords: multi-UAV systems; graph attention network; multi-agent reinforcement learning; mean field theory; partially observable

Citation: Yang, M.; Liu, G.; Zhou, Z.; Wang, J. Partially Observable Mean Field Multi-Agent Reinforcement Learning Based on Graph Attention Network for UAV Swarms. *Drones* **2023**, *7*, 476. https://doi.org/10.3390/drones7070476

Academic Editor: Diego González-Aguilera

Received: 22 May 2023
Revised: 14 July 2023
Accepted: 14 July 2023
Published: 20 July 2023

1. Introduction

The Multi-UAV system [1] has gained significant traction in various domains in recent years, in which multiple unmanned aerial vehicles collaborate to accomplish tasks. It has been widely applied in many fields, including Multi-UAV transportation, Multi-UAV area search and Multi-UAV formation flying [2]. It is noteworthy that a Multi-UAV cooperative transportation system [3–5] has been developed and implemented by our research group following the trend.

Multi-agent systems (MAS) [6,7] consist of multiple entities called agents that interact in a shared environment aiming to achieve some individual or collective objective. Multi-UAV system can be considered as a specific type of MAS. In a Multi-UAV system, each UAV is treated as an intelligent agent that perceives the environment, makes decisions, and

executes actions to collaboratively accomplish tasks. Therefore, the theories and methods in MAS research can provide guidance for the coordinated control of Multi-UAV systems. Among them, reinforcement learning, as a widely applied approach, can be used to achieve decision-making and collaboration among agents. Recently, reinforcement learning [8] has found widespread applications in various domains, including video games [9–11], task division [12], and education [13]. Multi-Agent Reinforcement Learning (MARL) [14] is an extension of reinforcement learning in MAS, and it has been applied in some real-world scenarios such as autonomous mobile [15,16] and multi-UAV collaborative scenarios. Shi et al. [3] successfully migrated MARL to real-world multi-UAV handling tasks by employing Recurrent Multi-Agent Deep Deterministic Policy Gradient (R-MADDPG) and domain randomization techniques. Liu et al. [17] proposed the Extensible Multi-Agent Deep Deterministic Policy Gradient (Ex-MADDPG) algorithm to address dynamic task assignment problems in UAV swarms, in which the practicality and effectiveness of the Ex-MADDPG algorithm were validated using a swarm of nine UAVs.

For large-scale MAS [18], it is unrealistic for agents to observe the entire environment globally limiting their ability to find appropriate actions. Furthermore, as the number of agents increases, joint optimization of all agents in a multi-agent environment may result in a huge joint state-action space, which also brings scalability challenges. Therefore, traditional MARL is difficult to apply to large-scale multi-agent environments, especially when the number of agents increases exponentially. To tackle this drawback, recent studies address the scalability issues of multi-agent reinforcement learning [19–21] by introducing mean field theory, i.e., the multi-agent problem is reduced to a simple two-agent problem. However, Yang et al. [19] assumes that each agent can observe global information, which is unreasonable in real-world MAS. As shown in Figure 1, multiple UAVs cooperate to perform search for fire detection, in which each UAV cannot get the global environment information, but it can obtain the information of other UAVs within the limited communication range. Thus, the adaptive communication strategy are required in the multi-UAV tasks such as collaborative search and rescue mission, which are typical partially observable scenarios.

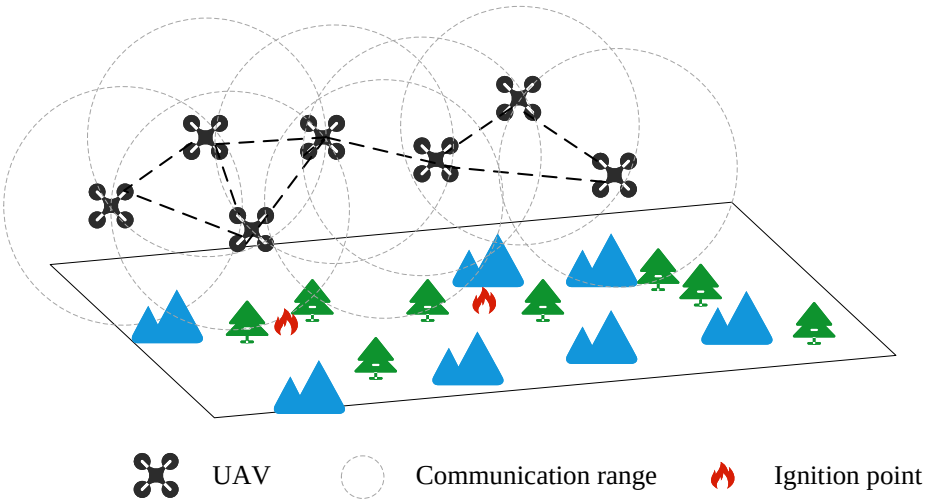


Figure 1. Multi-UAV system in forest fire detection.

Therefore, it is necessary to study large-scale multi-agent reinforcement learning algorithms in partially observable cases [22]. 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 [23,24]. Another way uses probability distribution or weighted mean field to update the mean action of neighborhood agents [25–28]. The graph attention and mean field mechanism are also utilized in [29],

which measure the interaction strength between agents with fixed relative positions and global information. Differently, we consider the dynamic changes 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 theory for partially observable scenarios.

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 [30–32] 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 actions more efficiently.

The main contributions of this paper are as follows:

- We propose a Graph attention network supported Partially observable Mean Field Multi-agent reinforcement learning (GPMF) algorithm, which can learn a decentralized agent policy from an environment without requiring global information of an environment. This is particularly valuable in scenarios with large-scale agents and limited observability, where existing methods lack sufficient judgment of the importance of neighbor agents.
- We provide theoretical evidence that the settings of the GPMF algorithm approach Nash equilibrium, showcasing its sound theoretical foundation.
- Experiments on three challenging game tasks in the MAgents framework show that GPMF outperforms two baseline algorithms as well as the state-of-the-art partially observable mean field reinforcement learning algorithms. These results validate the effectiveness of GPMF in solving dynamic cooperation problems in multi-UAV scenarios and facilitating collaboration among UAVs.

2. Related Work

2.1. Multi-UAV System Based on Multi-Agent Reinforcement Learning

The development of multi-agent reinforcement learning provides new solutions for multi-UAV missions. Qie et al. [33] formulated the Multi-UAV Target Assignment and Path Planning (MUTAPP) problem as a multi-agent system, and adopted the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm to solve the MUTAPP problem. For large-scale multi-UAV systems, Azzam et al. [34] developed MARL-based cooperative navigation of UAV swarms via centralized training and decentralized execution (CTDE), and the method was extended to work with a large number of agents , without retraining or changing the number of agents during training. Wang et al. [35] proposed a novel MARL paradigm, weighted mean field reinforcement learning, and conducted experiments on a large-scale UAV swarm confrontation environment to verify the effectiveness and scalability of the method. However, most of the group goals of existing MARL-based multi-UAV cooperation are based on a single cooperation or confrontation environment, and the complex environment of cooperation and confrontation has not been fully considered. In addition, the UAV observation range in the real scene is limited, so it is necessary to consider the solution of large-scale multi-UAV system cooperation and confrontation under the limitation of communication.

2.2. Large-Scale Partially Observable Multi-Agent Reinforcement Learning

Large-scale partially observable multi-agent reinforcement learning can provide technical support for the realization of large-scale multi-UAV tasks under communication constraints. For large-scale multi-agent environments, Yang et al. [19] 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. [23] 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. [24] proposed 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. [25] added the idea of mean field to MADDPG, and proposed 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] proposed 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 the central agent and its neighbors. Zhou et al. [26] used the mean action of neighbor agents as a label, and trained a mean field prediction network to replace the mean action. Subramanian et al. [27] proposed two multi-agent mean field Q-learning algorithms based on partially observable (POMFQ) settings: Fixed Observation Radius (FOR) and Probabilistic Distance-based Observability (PDO), extracting partial samples from Dirichlet or Gamma distribution to estimate partially observable mean action. Although these methods achieved good results, they did not fully consider the feature information of surrounding neighbors.

2.3. Graph Neural Networks (GNNs)

Graph Neural Networks (GNNs) are able to mine graph structures from data. In multi-agent reinforcement learning, GNNs can be used to model interactions between agents. In recent works, graph attention mechanism has been used for multi-agent reinforcement learning. Zhang et al. [36] integrated the importance of the information of surrounding agents based on the multi-head attention mechanism, effectively integrated the key information of the graph to represent the environment and improved the cooperation strategy of agents with the help of multi-agent reinforcement learning. DCG [37] 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. [38] proposed a deep implicit coordination graph (DICG) structure that can adapt to dynamic environments and learn implicit reasoning about joint actions or values via graph neural networks. Ruan et al. [39] 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 [40] accurately represented the interactions between agents during communication by modifying the standard graph attention network and compatible with differentiable directed graphs.

Our approach differs from related works 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. GPMF 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 mean action of surrounding agents in partially observable situations. GPMF does not require global information about the environment to learn decentralized agent policies from the environment.

3. Motivation and Preliminaries

In this section, we model discrete-time non-cooperative multi-agent task 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 states of the environment and A^j is the set of actions of the j-th agent. The reward function for agent j is $r^j : S \times A^1 \times \cdots \times A^N \to R$. State transitions are specified as $p : S \times A^1 \times \cdots \times A^N \to \Omega(S)$. γ is the discount factor, that represents the importance of future rewards. It is a constant between 0 and 1. The disadvantage is that it cannot be applied to the coexistence of multiple agents. Yang et al. [19] 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 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 Q-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) [27] is that the feature informations of surrounding neighbors are not fully considered. In a partially observable setting, the observable neighborhood agent information o^j of agent 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 describes graph attention networks.

3.1. Partially Observable Markov Decision Process

We mainly study Partially Observable Markov Decisions Process (POMDP) [14,22,41,42]. The POMDP 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 denotes the global state, A^j is 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 $Z(s, j) : S \times N \to O$, 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' joint policy $\pi \triangleq \left[\pi^1, \ldots, \pi^N\right]$, the value function of agent j under the joint strategy π is $v_\pi^j(s) = \sum_{t=0}^{\infty} \gamma^t E_{\pi,p}\left[r_t^j | s_0 = s\right]$, and then the Q-function can be formalized as $Q_\pi^j(s, a) = r^j(s, a) + \gamma E_{s' \sim p}\left[v_\pi^j(s')\right]$. Our work is based on the POMDP framework.

3.2. Partially Observable Mean Field Q-Learning

Mean field reinforcement learning [19] 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. [19] 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 denotes the discrete action of the agent j and is represented by one-shot coding. Mean field Q-function is cyclically updated according to Equations (2)–(5):

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

where

$$v^j(s_{t+1}) = \sum_{a_{t+1}^j} \pi^j\left(a_{t+1}^j \,|\, s_{t+1}, \tilde{a}_t^j\right) Q_\pi^j\left(s_{t+1}, a_{t+1}^j, \tilde{a}_t^j\right) \tag{3}$$

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

$$\pi^j\left(a_t^j \mid s_t, \bar{a}_{t-1}^j\right) = \frac{\exp\left(-\beta Q_\pi^j\left(s_t, a_t^j, \bar{a}_{t-1}^j\right)\right)}{\sum\limits_{a_t^{j'}\in A^j} \exp\left(-\beta Q_\pi^j\left(s_t, a_t^{j'}, \bar{a}_{t-1}^j\right)\right)} \tag{5}$$

where $\bar{a}_t^j$ is the mean action of the neighborhood agents, r_t^j 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. Yang et al. [19] assumed that each agent has global information, and for the central agent, the mean action of the neighboring agents is updated by Equation (4). However, in a partially observable multi-agent environment, the way of calculating the mean action in Equation (4) is no longer applicable. In the case of partial observability, Subramanian et al. [27] take U samples from the Dirichlet distribution to update the mean action of Equation (4), and achieve better performance than the mean field reinforcement learning algorithm. The formula is as follows:

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

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 have 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 2 shows a multi-robot system, each robot has a limited observation range and does not know all the information of the environment. The goal of the robot is to find the exit by observing the state of the surrounding robots. The action space of the agent is $\{up, down, left, right\}$. A central agent surrounded by a red circle is influenced by surrounding agents. We use the Dirichlet distribution to simulate and calculate the probability of the central agent moving in all directions, where the Dirichlet parameter η is set to 1, and the result is as follows:

$$\begin{cases} p_{up} = 0.25 \\ p_{down} = 0.5 \\ p_{left} = 0.143 \\ p_{right} = 0.1 \end{cases} \tag{7}$$

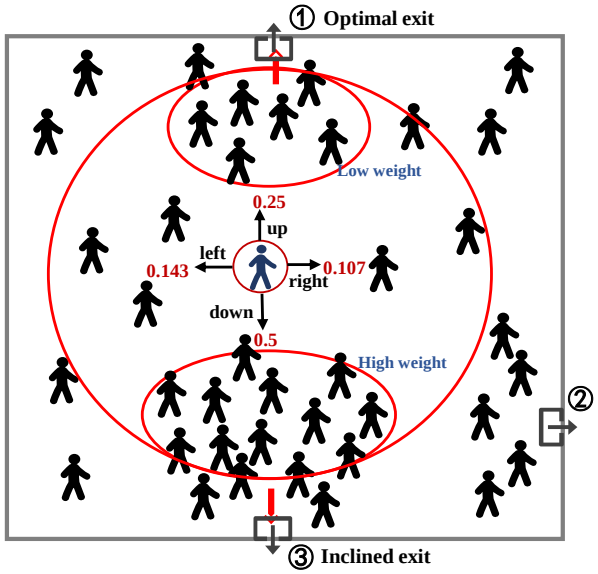


Figure 2. A multi-agent system looking for an exit, where the red agent in the center is assigned by Dirichlet to compute actions.

It can be seen that the probability of the agent moving *down* is the highest, which is essentially due to a large number of agents moving to *down*, so the robot is more inclined to move in the direction of exit 3. However, the closest exit to it is actually exit 1, and moving up is the best move. The Dirichlet distribution results in a local optimal solution rather than finding the optimal action.

Zhang et al. [43] believe that the correlation between two agents is crucial for multi-agent reinforcement learning. First, the paper [43] calculated the correlation coefficient between each pair of agents, and then shielded the communication among weakly correlated agents, thereby reducing the dimensions of the state-action value network in the input space. Inspired by Zhang et al. [43], for large-scale partially observable multi-agent environments, it is more necessary to select important neighborhood agents. In our paper, we adopt a graph attention network to filter out more important neighborhood agents, discard unimportant agent information, and achieve more accurate estimation of the mean action of neighborhood agents.

3.3. Graph Attention Network

Graph neural network [30] can better mine the graph structure form among data elements. Graph attention network (GAT) [44] is composed of a group of graph attention layers, which model the effect of neighbor nodes on central node. The attention score can be computed as Equations (8) and (9).

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

$$\alpha_{ij} = \text{softmax}j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in N_i} \exp(e_{ik})} \tag{9}$$

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

$$e_i = \sigma\left(\sum_{j \in N_i} \alpha_{ij} W h_j\right) \tag{10}$$

where e_i represents the feature of node i, N_i is the set of adjacent nodes of node i, and $\sigma(\cdot)$ is a nonlinear activation function.

4. Approach

In this section, we propose a Graph attention network supported Partially observable Mean Field Multi-agent reinforcement learning (GPMF) algorithm, which can be applied to large-scale partially observable multi-UAV cooperative pursuit tasks, where he communication range of each UAV is limited, and it can only observe the characteristic information of other UAVs within a fixed range. The multi-UAV cooperative pursuit problem is essentially a mixed scenario of competition and cooperation. Competition means that each UAV competes for resources to maximize rewards, while cooperation means that UAVs cooperate to achieve a common goal. Incorporating the multi-UAV cooperative pursuit problem into a multi-agent system (MAS), as shown in Figure 3, each UAV is abstracted as an agent, and then trained in a dynamic environment based on our proposed GPMF framework. The overall architecture of the GPMF algorithm is depicted in Figure 4. It has two important components: the Graph Attention Module and the Mean Field Module. In the 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. In 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 mean action of the important neighborhood agents that are partially observed by agent j. Each component is described in detail below.

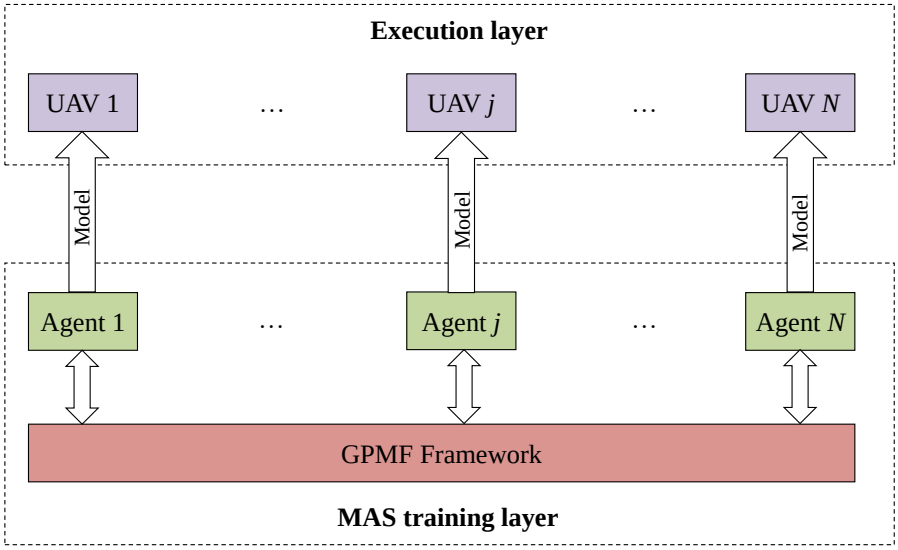


Figure 3. Abstract diagram of multi-UAV system.

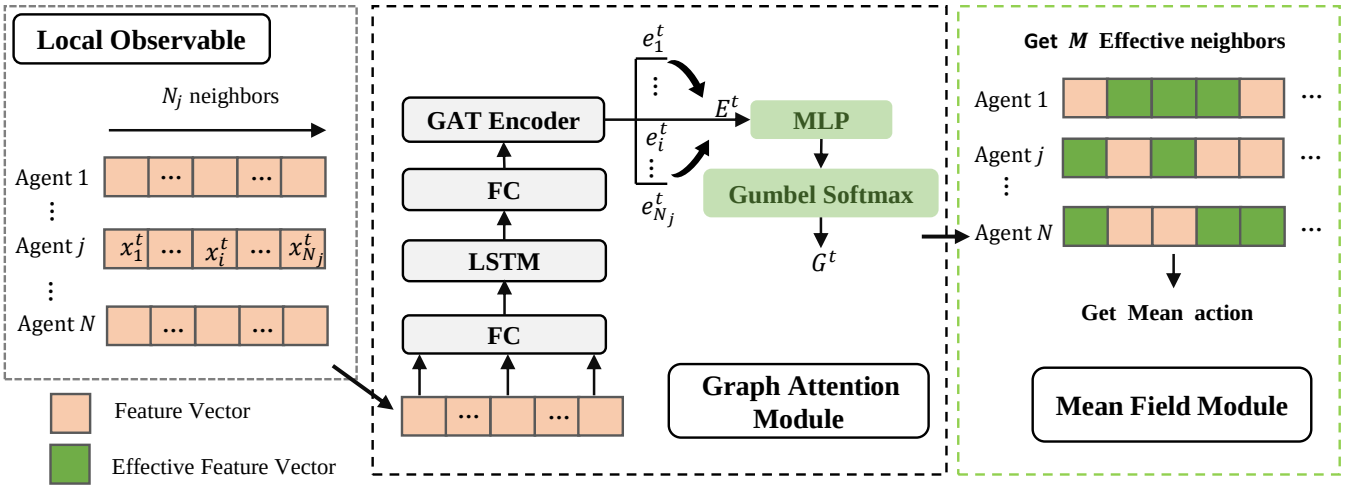


Figure 4. Schematic of GPMF. Each agent can observe the feature information of other agents within a fixed range, input it into the Graph Attention Module, and output an adjacency matrix to represent the effectiveness of the neighborhood agent to the central agent.

4.1. Graph Attention Module

To more accurately 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 concatenation of 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 MAS. This structure focuses on information from different agents by associating weights with observations based on the relative importance of other agents in their local observations. The graph attention structure is constructed by overlaying 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 layer 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:

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

Additionally, we take the form of the same attention mechanism as GAT [44], expressed as follows:

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

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, then 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 Module selects important M_j agents from the neighbors N_j of agent j, and compute the mean 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 Equation (17). Note that the Q-value here is a partially observable Q-value.

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

where the value function v^j is expressed as

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

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

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

4.3. Theoretical Proof

This subsection is devoted to proving that the setting of GPMF is close to the Nash equilibrium [45]. Subramanian et al. [27] 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 [27], 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,* $\mathrm{B}[w_i(t)|\mathcal{F}(t)] = 0$*. (5)* $\mathrm{B}\left[w_i^2(t)|\mathcal{F}(t)\right] \leq A + B \max_j \max_{\tau\leq t} \left|x_j(\tau)\right|^2$*, 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. [27] give the following lemma:

Lemma 1 ([27])**.** *When the Q-function is updated using the partially observable update rule in Equation (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 GPMF 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 GPMF Q-function is close to Nash equilibrium.

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

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

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

Proof of Theorem 1. 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 GPMF 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 GPMF Q-function Q_{GPMF} satisfy the following relation:*

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

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

Proof of Theorem 2. To prove 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 $>= \delta$, 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 GPMF also satisfies the relationship that the sum of the individual components is 1, and the component of the agent's mean 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.*

Proof of Theorem 3. This theorem can be proved by extending the results in Tsitsiklis [46]. 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 Equation (17), and Assumptions 3, 5, and 7 hold, the following holds for $t \to \infty$:*

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

where Q_* is the Nash Q-function, Q_{GPMF} 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 $L = |A|$.

Proof of Theorem 4. Theorem 4 shows that the GPMF 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 Equation (19) is approximately close to this plateau. Theorem 4 is an application of Theorem 3, using Assumptions 3, 5 and 7. However, in MARL, 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 GPMF follows the related work of the previous POMFQ [27], 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 mean action. Algorithm 1 gives the pseudocode of the GPMF 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 Network

Initialize the weights of Q-function Q_{ϕ^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 Equation (11), and encode h_j^t as a message m_j^t (Equation (12)).
    For each agent j, sample a^j fron policy induced by Q_{ϕ^j}(Equation (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 Equation (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.
  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 $y^j = r^j + \gamma v_{\phi_-^j}^{GPMF}(s')$ according to Equation (18).
    Minimize the loss $L(\phi^j) = \frac{1}{K} \sum \left(y^j - Q_{\phi_j}(s', a^j, \bar{a}^j)\right)^2$ to update Q network.
  For each agent j, update params of target network :$\phi_-^j \leftarrow \tau\phi^j + (1-\tau)\phi_-^j$.

In Algorithm 1, for each agent j, the Q-function is parameterized by ϕ and trained by minimizing the loss function $L(\phi^j)$, where y^j is the target mean field value calculated with the weights ϕ_-^j and finally trained by the gradient optimizer.

5. Experiments

For the multi-UAV cooperative pursuit scenario, the multi-agent reinforcement learning training platform Magent is used to simulate and verify the GPMF algorithm. The same experimental environments as [27] are adopted in our work, in which three three different strategies are designed in the MAgent framework [47]. It is noteworthy that the simulation environment is a partially observable multi-agent cooperative task environment, which is used to simulate the cooperative pursuit of multi-UAV swarms. In these three

tasks, the map size is set to 28*28, where the observation range of each UAV is 6 units. The state space is the concatenation of the feature information of other UAVs within each UAV's field of view, including location, health, and grouping information. The action space includes 13 move actions and 8 attack actions. In addition, each UAV is required to handle at most 20 other UAVs that are closest. To ensure a comprehensive understanding of the evaluation process, we provide detailed descriptions of the experimental setup and the training procedures are employed within the MAgent framework. This allow us to analyze the performance of the GPMF and evaluate its effectiveness in dealing with dynamic multi-UAV cooperative pursuit tasks.

5.1. Environment and Tasks

We will evaluate on three tasks: Multibattle game, Battle-Gathering and Predator-Prey.

- Multibattle game: There are two groups of UAVs participating in the competition scenario, each consisting of 25 UAVs. UAV gets -0.005 points for each movement, -0.1 points for attacking open space, 200 points for collision causing enemy UAV to fall, and 0.2 points for successful collision. Each UAV has a size of 2×2, a maximum health of 10 units, and a speed of 2 units. At the end of the mission, the team with the most surviving UAVs wins. If two teams have the same number of surviving UAVs, the team with the highest reward wins. Each team's reward is the sum of the rewards of the individual UAVs in the team.
- Battle-Gathering game: The distribution of resources in the environment is uniform, and each UAV can observe the location of all resources. In addition to rewards for shooting down enemy UAVs, each UAV can also occupy resources to obtain rewards. UAVs get 5 points for collides enemy UAVs, and the rest of the reward settings are the same as the Multibattle environment.
- Predator-Prey game: There are 40 predators (big UAVs) and 20 preys (small UAVs), respectively simulating two groups of UAVs. Each large UAV is a square with a size of 2×2, with a maximum health of 10 units and a speed of 2 units. The small UAV is a 1×1 square with a maximum health of 2 units and a speed of 2.5 units. In order to complete the task, the large UAV must collide with more small UAVs, and the small UAVs must try to evade. In addition, large UAVs and small UAVs have different reward functions. Specifically, large UAVs get -0.3 points for attacking spaces, 1 point for successful collisions with small UAVs, 100 points for shooting down small UAVs, -1 point for collisions with small UAVs, and large UAV drops earn 0.5 points. Unlike the Multibattle environment, when the round ends for a fairer duel, if the two teams have the same number of surviving UAVs, it is judged as a draw.

5.2. Evaluation

We consider four algorithms for the above three games: Mean Field Q-learning (MFQ), Mean Field Actor-Critic (MFAC) [19], Partially Observable Mean Field Q Learning-Fixed Observation Radius (POMFQ(FOR)) [27] and GPMF, where MFQ and MFAC are baselines and POMFQ(FOR) is the state-of-the-art algorithm.

The original baselines MFQ and MFAC were proposed by Yang et al. [19] based on global observability, and the idea was to approximate the influence of the neighborhood agents on the central agent as their mean 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 GPMF algorithm evaluates the effectiveness of neighborhood agents within a fixed range through the graph attention mechanism, selects more important neighborhood agents, and updates the mean 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. We would like to emphasize that our primary objective was to compare the performance of different algorithms, rather than conducting adversarial experiments between the same algorithms. Our goal was to assess the effectiveness of various approaches and identify the algorithm that performs best in the given tasks. The hyperparameters of MFQ, MFAC, POMFQ(FOR) and GPMF are basically the same. Table 1 lists the hyperparameters during training of the four algorithms, and the remaining parameters can be seen in [27].

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

6. Results and Discussion

In this section, we evaluate the performance of GPMF in three different environments, including Multibattle, Battle-Gathering, and Predator-Prey.

We benchmark against three algorithms, MFQ, MFAC and POMFQ(FOR), in which POMFQ(FOR) achieved the state-of-the-art performance to our best knowledge.

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

6.1. Reward

Figure 5 shows how the reward changes as the number of iterations increases during training. We plot the reward changes for the four algorithms in different environments during the first 1000 iterations. Since each algorithm is self-training which results in a large change in the reward, we use the least squares method to fit the reward curve. In Figure 5, the solid black line represents the reward of the GPMF algorithm. From Figure 5a–c, it can be seen that the reward of the GPMF algorithm increases rapidly, indicating that the GPMF algorithm can converge rapidly in the early stage, and the convergence performance is better than the other three algorithms.

6.2. ELO Calculation

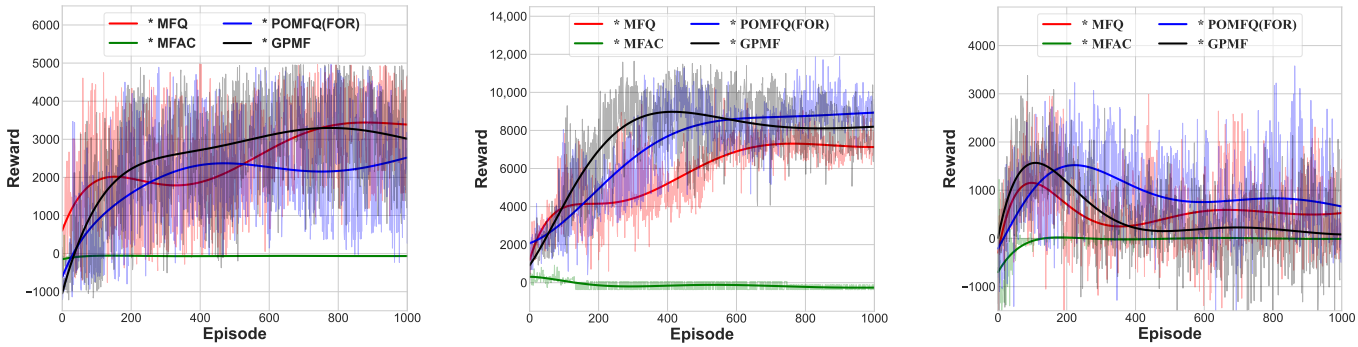
We use ELO Score [48] to evaluate the performance of the two groups of agents, the advantage of which is that it takes into account the strength gap between the opponents themselves. ELO ratings are commonly used in chess to evaluate one-on-one situations, and this approach can similarly be extended 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.



(**a**) Multibattle game (**b**) Battle-Gathering game (**c**) Predator-Prey game

Figure 5. Train results of three tasks. The reward curve for each algorithm is fitted by the least squares method.

Tables 2–4 show the ELO scores of the four algorithms on the three tasks. It can be seen from Table 2 that in Multibattle environment, the GPMF 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 GPMF algorithm is average. This is because some algorithms favor capturing resources for a quick reward rather than colliding with enemy UAVs. However, the final game winning or losing decision is made by comparing the number of remaining UAVs between the two teams of UAVs. As shown in Table 4, in Predator-Prey environment, the ELO score of the GPMF 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 GPMF algorithm is better than other three algorithms, showing better performance.

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

Algorithm 1	Algorithm 2	ELO Score 1	ELO Score 2
GPMF-1	POMFQ(FOR)-2	**3579**	820
GPMF-2	POMFQ(FOR)-1	2696	**2838**
GPMF-1	MFQ-2	**2098**	1508
GPMF-2	MFQ-1	**2535**	1695
GPMF-1	MFAC-2	**1350**	−49
GPMF-2	MFAC-1	−856	−**78**
POMFQ(FOR)-1	MFQ-2	**3145**	2577
POMFQ(FOR)-2	MFQ-1	2569	**2857**
POMFQ(FOR)-1	MFAC-2	−205	−**64**
POMFQ(FOR)-2	MFAC-1	**826**	−42
MFQ-1	MFAC-2	−142	−**49**
MFQ-2	MFAC-1	**610**	−46

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

Algorithm 1	Algorithm 2	ELO Score 1	ELO Score 2
GPMF-1	POMFQ(FOR)-2	7770	**8931**
GPMF-2	POMFQ(FOR)-1	8293	**9310**
GPMF-1	MFQ-2	6374	**10,870**
GPMF-2	MFQ-1	**8510**	8313
GPMF-1	MFAC-2	**5525**	10
GPMF-2	MFAC-1	**10,751**	−31
POMFQ(FOR)-1	MFQ-2	8526	**8760**
POMFQ(FOR)-2	MFQ-1	**8632**	8227
POMFQ(FOR)-1	MFAC-2	**12,722**	0
POMFQ(FOR)-2	MFAC-1	**12,171**	−88
MFQ-1	MFAC-2	**12,649**	49
MFQ-2	MFAC-1	**13,788**	−48

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

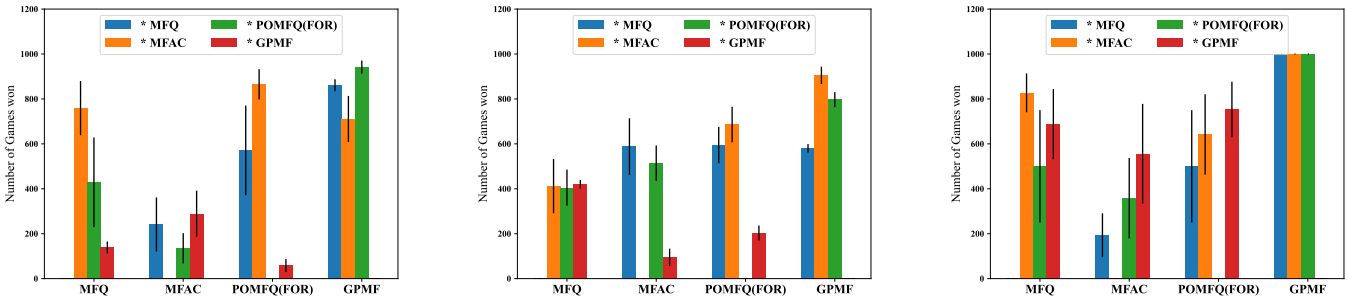
Algorithm 1	Algorithm 2	ELO Score 1	ELO Score 2
GPMF-1	POMFQ(FOR)-2	**421**	−32
GPMF-2	POMFQ(FOR)-1	**16**	7
GPMF-1	MFQ-2	**714**	−27
GPMF-2	MFQ-1	−**15**	−94
GPMF-2	MFAC-1	16	16
POMFQ(FOR)-1	MFQ-2	**66**	18
POMFQ(FOR)-2	MFQ-1	13	**24**
POMFQ(FOR)-1	MFAC-2	**16**	−16
POMFQ(FOR)-2	MFAC-1	**47**	16
MFQ-1	MFAC-2	**16**	−16
MFQ-2	MFAC-1	**174**	17

6.3. Results

Figure 6 shows the face-off results of the four algorithms in the three tasks. Figure 6a shows the faceoff results of Multibattle game. The different colored bars for each algorithm represent the results of an algorithm versus others. The vertical lines in the bar graph represent the standard deviation of wins for groups A and B over 1000 face-offs. Figure 6a shows GPMF against three other algorithms, all with a win rate above 0.7.

Figure 6b shows the faceoff results of Battle-Gathering game. In addition to the rewards for shooting down UAVs, UAVs can also get rewards for occupying resources. It can be seen that MFQ loses to all other algorithms, MFAC and POMFQ(FOR) perform in general, and our GPMF is clearly ahead of other algorithms.

Figure 6c shows the 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 GPMF algorithm is significantly better than other three algorithms, reaching a winning rate of 1.0.



(**a**) Multibattle game (**b**) Battle-Gathering game (**c**) Predator-Prey game

Figure 6. 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 GPMF algorithm is the result of the confrontation between GPMF and MFQ.

6.4. Visualization

To visualize the effectiveness of the GPMF algorithm, we visualize the confrontation between GPMF and POMFQ(FOR) in a Multibattle environment, as shown in Figure 7, where the red side is GPMF and the blue side is POMFQ(FOR). It can be seen from the confrontation process that for the GPMF algorithm, when a UAV decides to attack, the surrounding UAVs will also decide to attack under its influence, forming a good cooperation mechanism. On the contrary, for the POMFQ(FOR) algorithm, some blue-side UAVs are chosen to attack, some are chosen to escape, and no common fighting mechanism was formed. Similarly, in the Battle-Gathering environment of Figure 8, GPMF can learn the besiege mechanism well. In the Predator-Prey environment of Figure 9, when GPMF 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 GPMF.

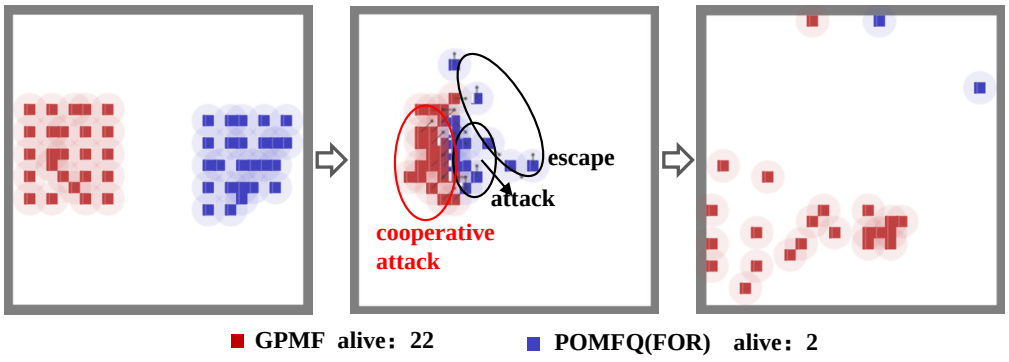


Figure 7. Visualization of the standoff between GPMF and POMFQ(FOR) in a Multibattle game.

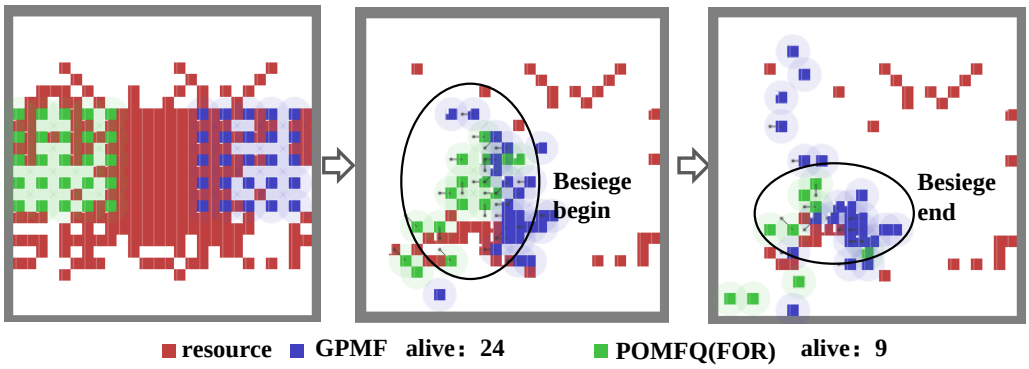


Figure 8. Visualization of the standoff between GPMF and POMFQ(FOR) in a Battle-Gathering game.

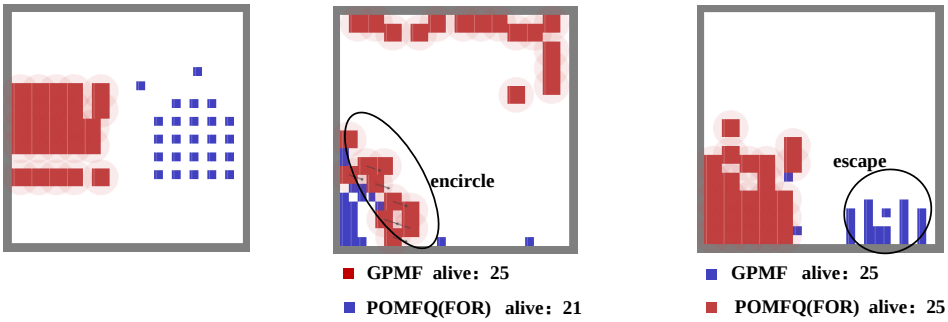


Figure 9. Visualization of the standoff between GPMF and POMFQ(FOR) in a Predator-Prey game.

6.5. Ablation Study

Figure 10 is an ablation study that investigates the performance of the GPMF 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 episodes 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.

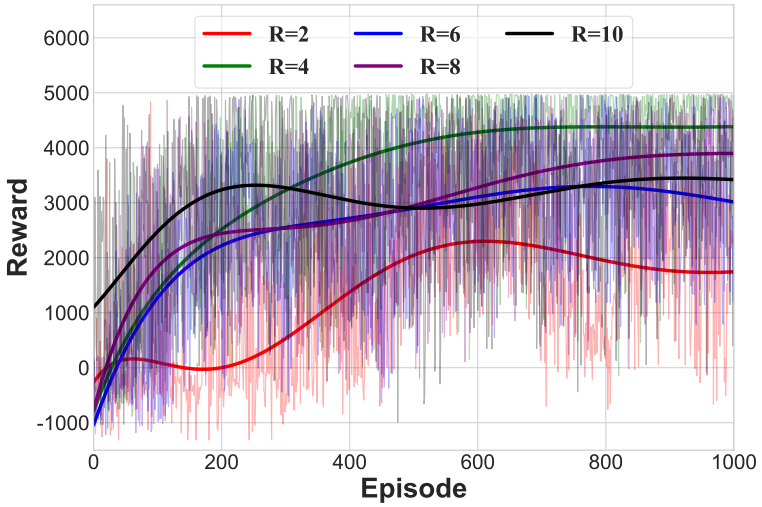


Figure 10. Ablation study. R represents the observation radius of an agent.

7. Conclusions

In this paper, we propose a novel multi-agent reinforcement learning algorithm, Graph attention network supported Partially observable Mean Field Multi-agent reinforcement learning (GPMF), to address large-scale partially observable multi-UAV cooperative pursuit problem. While existing methods approach Nash equilibrium, they fail to consider direct inter-agent correlations. By abstracting the UAVs of a multi-UAV system into agents, exploiting inter-agent correlations, a graph attention module is incorporated to capture the influence of actions taken by other agents at each time step. Experimental results on three challenging tasks within the MAgents framework demonstrate that our proposed approach outperforms baselines and state-of-the-art partially observable mean field reinforcement learning algorithms in all of these tasks. This further indicates the ability of our algorithm to improve communication and cooperation among UAVs.

In addition, we recognize the importance of investigating the robustness of multi-UAV system and improving the resilience of the GPMF algorithm against various interferences. One key aspect we aim to address is enhancing the reliability and effectiveness of communication protocols [7]. By developing more robust and efficient communication protocols, we can improve the overall resilience and robustness of multi-UAV system, enabling UAVs to effectively coordinate and adapt to changing environments and disturbances.

Author Contributions: Conceptualization, M.Y. and G.L.; methodology, M.Y.; investigation, Z.Z.; writing—original draft preparation, M.Y.; writing—review and editing, G.L. and J.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Shanghai Science and Technology Committee (No. 22511105500), the National Nature Science Foundation of China (No. 62172299) and the Fundamental Research Funds for the Central Universities (No. 2023-4-YB-05).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MARL	Multi-agent reinforcement learning
GAN	graph attention network
POMF	partially observable mean field
MAS	multi-agent system
POMDP	partially observable Markov decision process

References

1. Frattolillo, F.; Brunori, D.; Iocchi, L. Scalable and Cooperative Deep Reinforcement Learning Approaches for Multi-UAV Systems: A Systematic Review. *Drones* **2023**, *7*, 236. [CrossRef]
2. Wang, J.; Han, L.; Dong, X.; Li, Q.; Ren, Z. Distributed sliding mode control for time-varying formation tracking of multi-UAV system with a dynamic leader. *Aerosp. Sci. Technol.* **2021**, *111*, 106549. [CrossRef]
3. Shi, H.; Liu, G.; Zhang, K.; Zhou, Z.; Wang, J. MARL Sim2real Transfer: Merging Physical Reality With Digital Virtuality in Metaverse. *IEEE Trans. Syst. Man Cybern. Syst.* **2022**, *53*, 2107–2117. [CrossRef]
4. Weng, Q.L.; Liu, G.J.; Zhou, P.; Shi, H.R.; Zhang, K.W. Co-TS: Design and Implementation of a 2-UAV Cooperative Transportation System. *Int. J. Micro Air Veh.* **2023**, *15*, 17568293231158443. [CrossRef]
5. Zhou, P.; Liu, G.; Wang, J.; Weng, Q.; Zhang, K.; Zhou, Z. Lightweight unmanned aerial vehicle video object detection based on spatial-temporal correlation. *Int. J. Commun. Syst.* **2022**, *35*, e5334. [CrossRef]
6. Uhrmacher, A.M.; Weyns, D. *Multi-Agent Systems: SIMULATION and Applications*; CRC press: Boca Raton, FL, USA, 2009.
7. Cui, Y.; Luo, B.; Feng, Z.; Huang, T.; Gong, X. Resilient state containment of multi-agent systems against composite attacks via output feedback: A sampled-based event-triggered hierarchical approach. *Inf. Sci.* **2023**, *629*, 77–95. [CrossRef]

8. Zhou, Z.; Liu, G.; Tang, Y. Multi-Agent Reinforcement Learning: Methods, Applications, Visionary Prospects, and Challenges. *arXiv* **2023**, arXiv:2305.10091.
9. Vinyals, O.; Babuschkin, I.; Czarnecki, W.M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D.H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* **2019**, *575*, 350–354. [CrossRef]
10. Zhou, Z.; Liu, G.; Zhou, M. A Robust Mean-Field Actor-Critic Reinforcement Learning Against Adversarial Perturbations on Agent States. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, 1–12. [CrossRef]
11. Zhou, Z.; Liu, G. Robustness Testing for Multi-Agent Reinforcement Learning: State Perturbations on Critical Agents. *arXiv* **2023**, arXiv:2306.06136.
12. Guo, X.; Bi, Z.; Wang, J.; Qin, S.; Liu, S.; Qi, L. Reinforcement learning for disassembly system optimization problems: A survey. *Int. J. Netw. Dyn. Intell.* **2023**, *2*, 1–14. [CrossRef]
13. Gu, J.; Wang, J.; Guo, X.; Liu, G.; Qin, S.; Bi, Z. A Metaverse-Based Teaching Building Evacuation Training System With Deep Reinforcement Learning. *IEEE Trans. Syst. Man Cybern. Syst.* **2023**, *53*, 2209–2219. [CrossRef]
14. Zhang, K.; Yang, Z.; Başar, T. Multi-agent reinforcement learning: A selective overview of theories and algorithms. In *Handbook of Reinforcement Learning and Control*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 321–384.
15. Schmidt, L.M.; Brosig, J.; Plinge, A.; Eskofier, B.M.; Mutschler, C. An Introduction to Multi-Agent Reinforcement Learning and Review of its Application to Autonomous Mobility. *arXiv* **2022**, arXiv:2203.07676.
16. Zhang, Z.; Liu, J.; Liu, G.; Wang, J.; Zhang, J. Robustness verification of swish neural networks embedded in autonomous driving systems. *IEEE Trans. Comput. Soc. Syst.* **2022**, 1–10. [CrossRef]
17. Liu, B.; Wang, S.; Li, Q.; Zhao, X.; Pan, Y.; Wang, C. Task Assignment of UAV Swarms Based on Deep Reinforcement Learning. *Drones* **2023**, *7*, 297. [CrossRef]
18. Hernandez-Leal, P.; Kartal, B.; Taylor, M.E. A survey and critique of multiagent deep reinforcement learning. *Auton. Agents Multi-Agent Syst.* **2019**, *6*, 750–797. [CrossRef]
19. Yang, Y.; Luo, R.; Li, M.; Zhou, M.; Zhang, W.; Wang, J. Mean Field Multi-Agent Reinforcement Learning. In Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, 10–15 July 2018; Volume 80, pp. 5567–5576.
20. Xie, Q.; Yang, Z.; Wang, Z.; Minca, A. Learning while playing in mean-field games: Convergence and optimality. In Proceedings of the International Conference on Machine Learning. PMLR, Virtual, 18–24 July 2021; pp. 11436–11447.
21. Laurière, M.; Perrin, S.; Geist, M.; Pietquin, O. Learning Mean Field Games: A Survey. *arXiv* **2022**, arXiv:2205.12944.
22. Cai, Q.; Yang, Z.; Wang, Z. Reinforcement learning from partial observation: Linear function approximation with provable sample efficiency. In Proceedings of the International Conference on Machine Learning. PMLR, Baltimore, MD, USA, 17–23 July 2022; pp. 2485–2522.
23. Zhang, T.; Ye, Q.; Bian, J.; Xie, G.; Liu, T. MFVFD: A Multi-Agent Q-Learning Approach to Cooperative and Non-Cooperative Tasks. In Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event/ Montreal, QC, Canada, 19–27 August 2021; pp. 500–506. [CrossRef]
24. Gu, H.; Guo, X.; Wei, X.; Xu, R. Mean-field multi-agent reinforcement learning: A decentralized network approach. *arXiv* **2021**, arXiv:2108.02731.
25. Fang, B.; Wu, B.; Wang, Z.; Wang, H. Large-Scale Multi-agent Reinforcement Learning Based on Weighted Mean Field. In Proceedings of the Cognitive Systems and Signal Processing—5th International Conference, ICCSIP 2020, Zhuhai, China, 25–27 December 2020; Volume 1397, pp. 309–316.
26. Zhou, S.; Ren, W.; Ren, X.; Yi, X. Multi-Agent Mean Field Predict Reinforcement Learning. In Proceedings of the 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA), Dalian, China, 25–27 August 2020; pp. 625–629.
27. Subramanian, S.G.; Taylor, M.E.; Crowley, M.; Poupart, P. Partially Observable Mean Field Reinforcement Learning. In Proceedings of the AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, 3–7 May 2021; pp. 537–545.
28. Wu, T.; Li, W.; Jin, B.; Zhang, W.; Wang, X. Weighted Mean-Field Multi-Agent Reinforcement Learning via Reward Attribution Decomposition. In Proceedings of the International Conference on Database Systems for Advanced Applications, Virtual Event, 11–14 April 2022; pp. 301–316.
29. Hao, Q. Very Large Scale Multi-Agent Reinforcement Learning with Graph Attention Mean Field. 2023. Available online: https://openreview.net/forum?id=MdiVU9lMmVS (accessed on 5 March 2023).
30. Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Philip, S.Y. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 4–24. [CrossRef]
31. Fan, S.; Liu, G.; Li, J. A heterogeneous graph neural network with attribute enhancement and structure-aware attention. *IEEE Trans. Comput. Soc. Syst.* **2023**. [CrossRef]
32. Lou, X.; Liu, G.; Li, J. ASIAM-HGNN: Automatic Selection and Interpretable Aggregation of Meta-Path Instances for Heterogeneous Graph Neural Network. *Comput. Inform.* **2023**, *42*, 257–279. [CrossRef]
33. Qie, H.; Shi, D.; Shen, T.; Xu, X.; Li, Y.; Wang, L. Joint optimization of multi-UAV target assignment and path planning based on multi-agent reinforcement learning. *IEEE Access* **2019**, *7*, 146264–146272. [CrossRef]

34. Azzam, R.; Boiko, I.; Zweiri, Y. Swarm Cooperative Navigation Using Centralized Training and Decentralized Execution. *Drones* **2023**, *7*, 193. [CrossRef]
35. Wang, B.; Li, S.; Gao, X.; Xie, T. Weighted mean field reinforcement learning for large-scale UAV swarm confrontation. *Appl. Intell.* **2023**, *53*, 5274–5289. [CrossRef]
36. Zhang, H.; Cheng, J.; Zhang, L.; Li, Y.; Zhang, W. H2GNN: Hierarchical-Hops Graph Neural Networks for Multi-Robot Exploration in Unknown Environments. *IEEE Robot. Autom. Lett.* **2022**, *7*, 3435–3442. [CrossRef]
37. Boehmer, W.; Kurin, V.; Whiteson, S. Deep Coordination Graphs. In Proceedings of the 37th International Conference on Machine Learning, ICML 2020, Virtual Event, 13–18 July 2020; Volume 119, pp. 980–991.
38. Li, S.; Gupta, J.K.; Morales, P.; Allen, R.E.; Kochenderfer, M.J. Deep Implicit Coordination Graphs for Multi-agent Reinforcement Learning. In Proceedings of the AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, 3–7 May 2021; pp. 764–772.
39. Ruan, J.; Du, Y.; Xiong, X.; Xing, D.; Li, X.; Meng, L.; Zhang, H.; Wang, J.; Xu, B. GCS: Graph-based Coordination Strategy for Multi-Agent Reinforcement Learning. *arXiv* **2022**, arXiv:2201.06257.
40. Niu, Y.; Paleja, R.R.; Gombolay, M.C. Multi-Agent Graph-Attention Communication and Teaming. In Proceedings of the AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, 3–7 May 2021; pp. 964–973.
41. Littman, M.L. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning Proceedings 1994*; Elsevier: Amsterdam, The Netherlands, 1994; pp. 157–163.
42. Oliehoek, F.A.; Amato, C. *A Concise Introduction to Decentralized POMDPS*; Springer: Berlin/Heidelberg, Germany, 2016.
43. Zhang, Y.; Yang, Q.; An, D.; Zhang, C. Coordination Between Individual Agents in Multi-Agent Reinforcement Learning. In Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence, Virtual Event, 2–9 February 2021; pp. 11387–11394.
44. Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. *Stat* **2017**, *1050*, 20.
45. Fan, J.; Wang, Z.; Xie, Y.; Yang, Z. A theoretical analysis of deep Q-learning. In Proceedings of the Learning for Dynamics and Control, Online Event, 10–11 June 2020; pp. 486–489.
46. Tsitsiklis, J.N. Asynchronous stochastic approximation and Q-learning. *Mach. Learn.* **1994**, *16*, 185–202. [CrossRef]
47. Zheng, L.; Yang, J.; Cai, H.; Zhou, M.; Zhang, W.; Wang, J.; Yu, Y. MAgent: A Many-Agent Reinforcement Learning Platform for Artificial Collective Intelligence. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; pp. 8222–8223.
48. Jaderberg, M.; Czarnecki, W.M.; Dunning, I.; Marris, L.; Lever, G.; Castañeda, A.G.; Beattie, C.; Rabinowitz, N.C.; Morcos, A.S.; Ruderman, A.; et al. Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. *arXiv* **2018**, arXiv:abs/1807.01281.