Chinese Journal of Aeronautics, (2025), 38(9): 103484

Chinese Society of Aeronautics and Astronautics
& Beihang University

Chinese Journal of Aeronautics

cja@buaa.edu.cn
www.sciencedirect.com

FULL LENGTH ARTICLE

Large-scale UAV swarm path planning based on mean-field reinforcement learning

Yaozhong ZHANG [a,*], Meiyan DING [a], Yao YUAN [a], Jiandong ZHANG [a], Qiming YANG [a], Guoqing SHI [a], Jianming JIANG [b]

[a] *School of Electronics and Information, Northwestern Polytechnical University, Xi'an 710072, China*
[b] *College of Computer Science and Technology, Zhejiang Normal University, Jinhua 321004, China*

Received 18 July 2024; revised 22 September 2024; accepted 11 December 2024
Available online 11 March 2025

KEYWORDS

DDPG;
Mean field games;
Partially observable;
PO-WMFDDPG;
UAV swarm

Abstract In this paper, a deep deterministic policy gradient algorithm based on Partially Observable Weighted Mean Field Reinforcement Learning (PO-WMFRL) framework is designed to solve the problem of path planning in large-scale Unmanned Aerial Vehicle (UAV) swarm operations. We establish a motion control and detection communication model of UAVs. A simulation environment is carried out with No-Fly Zone (NFZ), the task assembly point is established, and the long-term reward and immediate reward functions are designed for large-scale UAV swarm path planning problem. Considering the combat characteristics of large-scale UAV swarm, we improve the traditional Deep Deterministic Policy Gradient (DDPG) algorithm and propose a Partially Observable Weighted Mean Field Deep Deterministic Policy Gradient (PO-WMFDDPG) algorithm. The effectiveness of the PO-WMFDDPG algorithm is verified through simulation, and through the comparative analysis with the DDPG and MFDDPG algorithms, it is verified that the PO-WMFDDPG algorithm has a higher task success rate and convergence speed.

1. Introduction

In recent years, with the presence of autonomy and self-organizations in Unmanned Aerial Vehicles (UAVs), the immense combat potential of UAVs on the battlefield has been gradually validated. Especially in the battlefield of the Russo-Ukrainian conflict, numerous successful applications of combat operations have been achieved. It can be considered that large-scale UAV swarm have become one of the important forms of combat on modern battlefields.[1] UAVs are gradually replacing humans to perform dangerous tasks in complex environments and unmanned areas. For such tasks, quick and effective path planning is a vital component to ensure the safety of UAV flight.[2]

In the early stage of research work on the path planning problem, based on the graph theory, most researchers developed traditional algorithms such as the A* algorithm, Rapidly-exploring Random Trees (RRT) algorithm, the ant

* Corresponding author.
E-mail address: zhang_y_z@nwpu.edu.cn (Y. ZHANG).

Peer review under responsibility of Editorial Committee of CJA

ELSEVIER Production and hosting by Elsevier

https://doi.org/10.1016/j.cja.2025.103484

colony algorithm,[3] and so on. Traditional algorithms must acquire some or all task environment information prior to path planning.[4] However, in real-world scenarios, uncertainty in environmental information is inevitable, with factors such as sensor noise and imperfect motion execution mechanisms contributing to this uncertainty. As the complexity and uncertainty of task environments increase, UAV task path planning encounters greater challenges. Recently, there has been substantial growth in research focused on UAV path planning using Artificial Intelligence (AI) algorithms, with significant advancements in the application of Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL) theories in this domain.[5] RL as an AI method, is extensively employed to describe and solve the problem of an agent learning strategies to maximize rewards or achieve specific goals through interactions with the environment. Concurrently, Deep Learning (DL) algorithms emulate the neurons of the human brain by constructing multi-layer Artificial Neural Networks (ANNs) to learn and recognize data features. The integration of neural networks with RL gives rise to the DRL method. This method leverages the robust data processing and computational capabilities of ANNs to address more complex problems, particularly in continuous environments.[6] However, over time, single-agent deep reinforcement learning algorithms have become unable to meet the needs of actual tasks. Therefore, Multi-Agent Reinforcement Learning (MARL) algorithms have emerged and achieved significant results in fields such as multi-UAV collaborative applications,[7,8] multi-robot coordinated control,[9] and multi-player virtual games.[10,11]

MARL algorithms usually perform well in task environments with fewer agents, but their performance is poor in swarm tasks with large-scale agents, mainly facing the following challenges:

(1) Complexity of joint states and action spaces. As the number of agents increases, the size of the joint state/action space grows exponentially, making the learning process more difficult. The state space and action space that need to be searched are too large and difficult to converge. The strategy training of the agent is prone to fall into the local optimum, which makes the entire system unable to reach the global optimal solution;
(2) Environmental uncertainty. In a large-scale agent system, the behavior of any agent may have a significant impact on the behavior decision of another agents. Therefore, as the number of agents increases, the uncertainty of the environment will also increase, making the algorithm training process more difficult;
(3) Difficulty of designing communication interaction protocols. In large-scale agent systems, corresponding communication interaction protocols need to be designed to ensure collaborative decision-making. This process needs to consider aspects such as information exchange, strategy coordination, and decision-making cooperation among agents. As the number of agents increases, the design and implementation of communication interaction protocols become more complex and challenging.

It is difficult to solve the collaborative control problem in a large-scale agent swarm environment using traditional MARL algorithms. At present, some researchers have combined MARL algorithms with Game Theory (GT) and Mean Field Games (MFG) theory[12] to design better control and optimization strategies, thereby achieving collaborative decision-making and goal optimization in large-scale agent systems. Yang et al.[13] firstly proposed the Mean Field Reinforcement Learning (MFRL) method, which approximates the interaction within the group of agents by equating it to the interaction between a single agent and the average effect of the remaining agents. The optimal strategy learning of a single agent is affected by the dynamic changes of the group, and the dynamics of the group are determined by the pattern changes of each strategy. The designed Mean Field Q-learning (MFQ) algorithm and Mean Field Actor-Critic (MFAC) algorithm performed well in simulation experiments. In situations with large-scale agents and complex environment, mean field reinforcement learning algorithms can be effectively applied to multi-agent task environments.

Since the mean field reinforcement learning algorithm was proposed, it has been widely used in various discrete-time reinforcement learning environments. uz Zaman et al.,[14] under the premise of considering non-stationary Mean-Field Games (MFG) in an infinite horizon, investigated the large-scale Multi-Agent Reinforcement Learning (MARL) process of discrete-time Linear-Quadratic Mean-Field Games (LQ-MFG). They proposed a mean field equilibrium for LQ-MFG, iteratively calculated based on the Actor-Critic (AC) framework algorithm. Cui et al.[15] examined a discrete-time mean field control model with a common environment state, establishing approximate optimality in the case of finite agents and identifying the existence of an optimal fixed strategy. Numerous studies have demonstrated that MFG has certain advantages in addressing cooperative game problems involving large-scale agents. Mean field reinforcement learning algorithms have shown promising results in the cooperative control of large-scale agent swarms.

However, there remains room for improvement and some limitations. When employing mean field theory to average the action information of global agents, each agent in the system is simplistically considered equally important, which may not be suitable for complex task scenarios involving collaborative behaviors. In actual task scenarios, it is often challenging for a single agent in a large-scale agent swarm to obtain global observation information. Directly using mean field theory to average the action information of all agents may not be appropriate.

Therefore, this paper improves the mean field theory based on the locally observable information of each agent and introduces the multi-head attention mechanism into mean field theory, thereby dividing the global agents into multiple independent entities, performing self-attention calculations on each, and considering the influence weight of each agent on the current task based on the agent's current state. This approach enhances the robustness of the MFRL algorithm and effectively improves the algorithm's task success rate.

In summary, this paper proposes the Partially Observable Weighted Mean Field Deep Deterministic Policy Gradient (PO-WMFDDPG) algorithm, which makes the following innovations based on the Deep Deterministic Policy Gradient (DDPG)algorithm:

(1) Based on the mean field reinforcement learning framework, we integrate mean field theory with the DDPG algorithm, applying it to continuous drone swarm

reinforcement learning task environments, which effectively addresses issues related to large-scale UAV swarm task path planning.
(2) To better align with actual combat environments, we propose a partially observable mean field theory, enhancing the mean field theory based on the local observable information of each UAV.
(3) We introduce a multi-head attention mechanism to weight the mean field, thereby ensuring the effectiveness and rationality of mean field approximation in large-scale agent task environments.

The remainder of the paper is organized as follows. The problem modeling is given in Section 1. The theoretical basis of the PO-WMFDDPG algorithm and the corresponding algorithm content is given in Section 2. The specific design of the path planning model of UAV Swarm based on PO-WMFDDPG is given in Section 3. The design of the simulation experiment and the analysis of experimental results are given in Section 4. Finally, conclusions are drawn in Section 5.

2. Problem modeling

2.1. Task scenario

In a complex and ever-changing battlefield environment, UAVs must fully consider the impact of the complexity of the battlefield environment on their behavior when performing tasks.[16,17] To this end, terrain obstacles and enemy air defense fire units are designed to be equivalent to circular No-Fly Zone (NFZ). Under the premise of NFZs, the task path planning problem of UAV swarms is studied.

As shown in Fig. 1, there are multiple static or dynamic NFZs in the $W \times H$ two-dimensional battlefield environment. Our task swarm consisting of N UAVs needs to safely and quickly cross the battlefield environment to reach the predetermined task area.

2.2. UAV model

In order to facilitate the problem analysis, the UAV is formulated into a particle model. Its motion state is determined by position and velocity. The linear acceleration a_{vt}^i and angular acceleration $\alpha_{\alpha t}^i$ are used to control the speed and direction of the UAV, as shown in the Fig. 2.

The instantaneous state information $\boldsymbol{Q}_t^i$ of the UAV U^i at time t is expressed as:

$$\boldsymbol{Q}_t^i = \left[x_t^i, y_t^i, v\, ti, a_t^i\right]^{\mathrm{T}} \tag{1}$$

where x_t^i and y_t^i represent the position of the U^i at time t; v_t^i represents the velocity; α_t^i represents the heading angle, α_{vt}^i represents the linear acceleration, and $\alpha_{\alpha t}^i$ represents the angular acceleration.

So the instantaneous state information of the U^i at time $t+1$ is as follows:

$$\begin{cases} v_{t+1}^i = v_t^i + a_{vt}^i \cdot \Delta t \\ \alpha_{t+1}^i = \alpha_t^i + a_{\alpha t}^i \cdot \Delta t \\ x_{t+1}^i = x_t^i + v_{t+1}^i \cdot \cos \alpha_{t+1}^i \cdot \Delta t \\ y_{t+1}^i = y_t^i + v_{t+1}^i \cdot \sin \alpha_{t+1}^i \cdot \Delta t \end{cases} \tag{2}$$

$$\boldsymbol{Q}_{t+1}^i = \left[x_{t\,+\,1}^i, y_{t\,+\,1}^i, v\, t+1i, \alpha_{t\,+\,1}^i\right]^{\mathrm{T}} \tag{3}$$

where Δt represents the simulation time step.

2.3. UAV radar detection and communication model

During the task, the UAV obtains limited status information, and the main information comes from its radar detection and inter-UAV communication. The radar detection and inter-UAV communication model is constructed as shown in the Fig. 3.

The radar detection area of the UAV is a sector in front of the head, with a maximum detection distance of d_{probe} and a detection angle of $2\theta_{\text{probe}}$.

Considering the constraints of UAV radar performance, up to 5 detection targets can be stably tracked simultaneously. Set the safe distance d_{safe} and the danger distance d_{danger} for the UAV. When the distance between UAVs is less than the safe distance d_{safe}, there is a risk of collision, when the distance between UAVs is less than the danger distance d_{danger}, UAVs crashed in a collision.

The maximum communication distance within the UAV swarm is d_{comm}. Considering the communication node constraint, each UAV can communicate with up to 50 other UAVs in the swarm simultaneously.

3. PO-WMFDDPG algorithm model

3.1. DDPG algorithm

As shown in the Fig. 4, the DDPG algorithm integrates the idea of DL based on the Deterministic Policy Gradient (DPG) algorithm. With the help of the Deep Q-Network

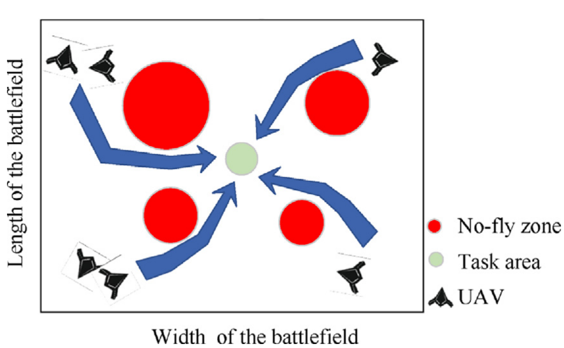


Fig. 1 UAV swarm path planning battlefield environment.

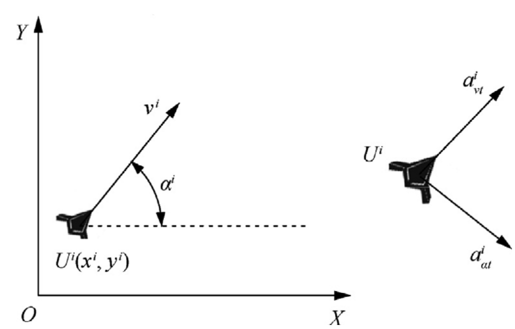


Fig. 2 UAV motion control model.

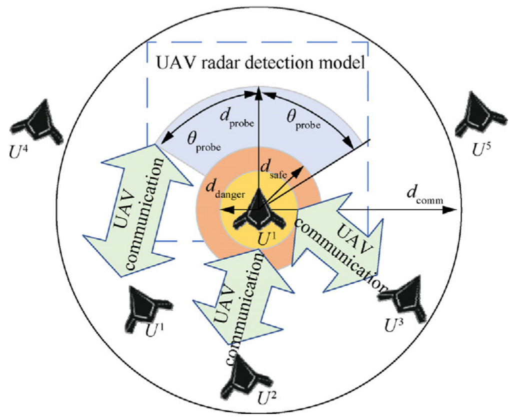


Fig. 3 Schematic of radar detection and inter-UAV communication model.

(DQN) algorithm, it superimposes the dual neural network of the eval network and the target network on the basis of the AC framework, effectively solving the problem of the difficulty of convergence of policy-based algorithms.[18,19]

The DDPG algorithm contains four neural networks: actor_eval network, actor_target network, critic_eval network and critic_target network. The actor_eval network updates the policy network parameter $\boldsymbol{\theta}_{\text{actor\_eval}}$ and obtains the optimal action $\boldsymbol{a}_t$ according to the current state $\boldsymbol{s}_t$ of the agent. After the agent executes the action $\boldsymbol{a}_t$, the state is updated to $\boldsymbol{s}_{t+1}$. The actor_target network parameter is $\boldsymbol{\theta}_{\text{actor\_target}}$, which is used to select the optimal action $\boldsymbol{a}_{t+1}$ in state $\boldsymbol{s}_{t+1}$. The critic_eval network updates the value network parameters $\boldsymbol{\theta}_{\text{critic\_eval}}$, evaluates the action $\boldsymbol{a}_t$ output by the actor_eval network based on the current state $\boldsymbol{s}_t$ of the agent, and outputs the corresponding value function $Q(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\theta}_{\text{critic\_eval}})$. The critic_target network parameters is $\boldsymbol{\theta}_{\text{critic\_target}}$, which evaluates the agent's choice of action $\boldsymbol{a}_{t+1}$ in state $\boldsymbol{s}_{t+1}$, and the output value function is $Q(\boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1}; \boldsymbol{\theta}_{\text{critic\_eval}})$.

The eval network updates the network parameters in real time, while the target network parameters follow the parameters of the eval neural network through a periodic soft update strategy and assist the training process of the eval neural network. This mechanism can disrupt the correlation, thereby reducing the correlation between the estimated Q value and the target Q value to a certain extent and improving the stability of the algorithm.[20]

The overall framework of the DDPG algorithm is as follows:

In order to improve the agent's exploration capability during the action selection phase of algorithm training, Ornstein-Uhlenbeck (OU) noise is added to increase the randomness of action selection, so that the algorithm may produce incoherent random actions in each exploration to avoid the algorithm falling into the local optimum. This mechanism transforms the algorithm's action selection from deterministic to random. The OU noise process is as follows:

$$\mathrm{d}x_t = \phi(\mu - x_t)\mathrm{d}t + \sigma W_t \tag{4}$$

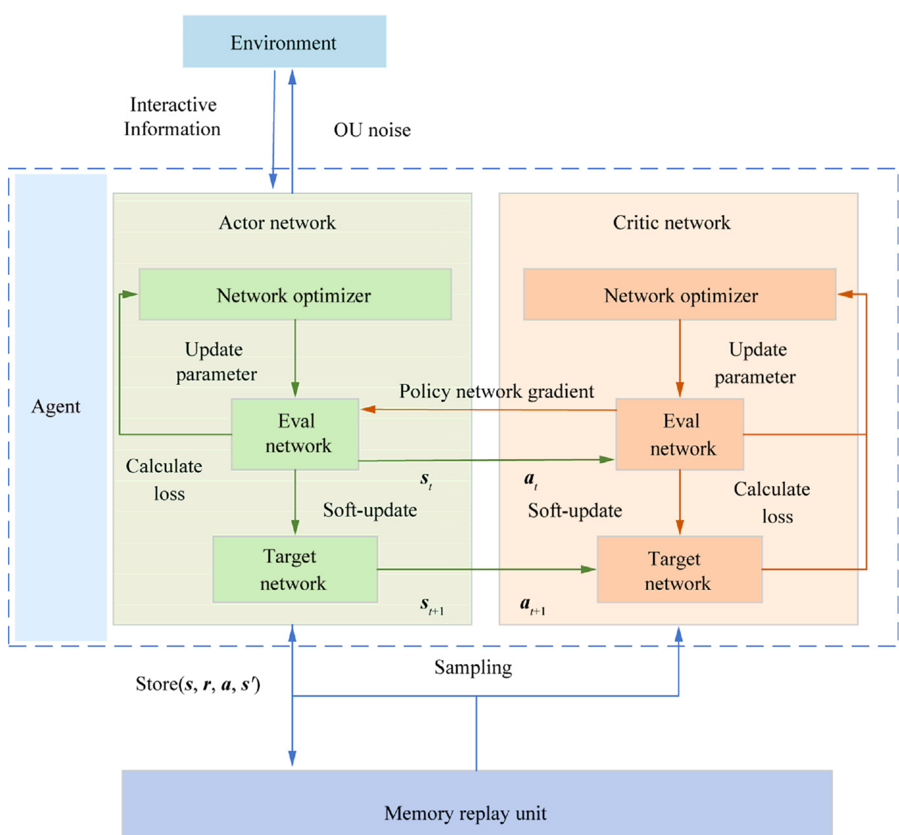


Fig. 4 Schematic of DDPG algorithm.

where ϕ is the parameter of the system's response degree to interference, $\phi > 0$; μ, σ are the mean and variance of OU noise; W_t is the Wiener process, that is, Brownian motion.

3.2. Mean field game reinforcement learning

In a multi-agent task environment, in order to solve the dimensionality explosion problem caused by large-scale agents, some multi-agent reinforcement learning methods use pairwise local interactions between agents [13] to decompose the Q function:

$$Q^i(\boldsymbol{s},\boldsymbol{a}) = \frac{1}{N^i}\sum_{k\in N(i)} Q^i\left(\boldsymbol{s},\boldsymbol{a}^i,\boldsymbol{a}^k\right) \tag{5}$$

where $\boldsymbol{N}(i)$ is the set of neighboring agents of agent i, and the set size is $N^i = |\boldsymbol{N}(i)|$. The pairwise approximation between agents and their neighbors reduces the complexity of interactions between agents and implicitly preserves the global interactions between agents.

Calculates the average action $\bar{\boldsymbol{a}}^i$ of all neighbor agents of agent i and represent the action $\boldsymbol{a}^k$ of any neighboring agent k as the sum of $\bar{\boldsymbol{a}}^i$ and a random perturbation $\boldsymbol{\delta a}^{i,k}$:

$$\begin{cases} \bar{\boldsymbol{a}}^i = \frac{1}{N^i}\sum_{k=1}^{N^i}\boldsymbol{a}^k \\ \boldsymbol{a}^k = \bar{\boldsymbol{a}}^i + \boldsymbol{\delta a}^{i,k}, \quad k\in \boldsymbol{N}(i) \end{cases} \tag{6}$$

According to Taylor's theorem, $Q^i(\boldsymbol{s},\boldsymbol{a})$ can be expanded to obtain:

$$\begin{aligned} Q^i(\boldsymbol{s},\boldsymbol{a}) &= \tfrac{1}{N^i}\sum_k Q^i(\boldsymbol{s},\boldsymbol{a}^i,\boldsymbol{a}^k) \\ &= \tfrac{1}{N^i}\sum_k\left[Q^i\left(\boldsymbol{s},\boldsymbol{a}^i,\bar{\boldsymbol{a}}^i\right) + \nabla_{\bar{a}^i}Q^i\left(\boldsymbol{s},\boldsymbol{a}^i,\bar{\boldsymbol{a}}^i\right)\cdot\boldsymbol{\delta a}^{i,k}\right] + \\ &\quad \sum_k\left[\tfrac{1}{2}\boldsymbol{\delta a}^{i,k}\cdot\nabla_{\tilde{\boldsymbol{a}}^{i,k}}Q^i\left(\boldsymbol{s},\boldsymbol{a}^i,\tilde{\boldsymbol{a}}^{i,k}\right)\cdot\boldsymbol{\delta a}^{i,k}\right] \\ &= Q^i\left(\boldsymbol{s},\boldsymbol{a}^i,\bar{\boldsymbol{a}}^i\right) + \nabla_{\bar{\boldsymbol{a}}^i}Q^i\left(\boldsymbol{s},\boldsymbol{a}^i,\bar{\boldsymbol{a}}^i\right)\cdot\left[\tfrac{1}{N^i}\sum_k\boldsymbol{\delta a}^{i,k}\right] + \\ &\quad \tfrac{1}{2N^i}\sum_k\left[\boldsymbol{\delta a}^{i,k}\cdot\nabla^2_{\tilde{\boldsymbol{a}}^{i,k}}Q^i\left(\boldsymbol{s},\boldsymbol{a}^i,\tilde{\boldsymbol{a}}^{i,k}\right)\cdot\boldsymbol{\delta a}^{i,k}\right] \\ &= Q^i\left(\boldsymbol{s},\boldsymbol{a}^i,\bar{\boldsymbol{a}}^i\right) + \tfrac{1}{2N^i}\sum_k R^i_{s\boldsymbol{a}^i}(a^k) \approx Q^i\left(\boldsymbol{s},\boldsymbol{a}^i,\bar{\boldsymbol{a}}^i\right) \end{aligned} \tag{7}$$

where $R^i_{s,a^i}(\boldsymbol{a}^k) = \boldsymbol{\delta a}^{i,k}\cdot\nabla^2_{\tilde{\boldsymbol{a}}^{i,k}}Q^i\left(\boldsymbol{s},\boldsymbol{a}^i,\tilde{\boldsymbol{a}}^{i,k}\right)\cdot\boldsymbol{\delta a}^{i,k}$ is the remainder of the Taylor polynomial, which is essentially a bounded random variable.

Under the mean field approximation, the RL problem for large-scale agents is transformed into solving the optimal strategy $\boldsymbol{\pi}^i_t$ for agent i, taking into account the average action $\bar{\boldsymbol{a}}^i$ of all neighboring agents. Calculate the policy $\boldsymbol{\pi}^i_t$ for each agent i by introducing an iterative process. The action $\boldsymbol{a}^k$ obeys the neighbor of agent i based on the average action α^k parameterized strategy $\boldsymbol{\pi}^k_t$, and the average action $\bar{\boldsymbol{a}}^i$ of the neighbor of agent i is:

$$\bar{\boldsymbol{a}}^i = \frac{1}{N^i}\sum_k \boldsymbol{a}^k, \boldsymbol{a}^k \sim \boldsymbol{\pi}^k_t\left(\cdot|\boldsymbol{s},\boldsymbol{a}^k_-\right) \tag{8}$$

After calculating $\bar{\boldsymbol{a}}^i$, depending on the current $\bar{\boldsymbol{a}}^i$, we determine a new Boltzmann strategy for each agent as follows:

$$\pi^i_t\left(\boldsymbol{a}^i|\boldsymbol{s},\bar{\boldsymbol{a}}^i\right) = \frac{\exp\left(-\beta Q^i\left(\boldsymbol{s},\boldsymbol{a}^i,\bar{\boldsymbol{a}}^i\right)\right)}{\sum_{\boldsymbol{a}^{i'}\in A^i}\exp\left(-\beta Q^i\left(\boldsymbol{s},\boldsymbol{a}^{i'},\bar{\boldsymbol{a}}^i\right)\right)} \tag{9}$$

3.3. Partially observable mean field

In multi-agent task scenarios, the state action values $Q^i(\boldsymbol{s},\boldsymbol{a})$ of the critical network in the DDPG algorithm are approximated using partially observable mean field theory, and the action $\boldsymbol{a}^i$ selection of agent i follows a deterministic strategy $\boldsymbol{a}^i = \mu_\theta(\boldsymbol{s})$.

In MFG theory, the set of all agents except agent i is called neighbor agents, and the neighbor agents are approximated by the mean field theory, which is not reasonable. For large-scale UAV swarm task environments, neighbor UAVs that are closer to UAV i will have a more obvious impact on the behavioral decision-making of UAV i, but the impact of neighbor UAVs that are far away from UAV i on the behavioral decision-making of UAV i can be basically ignored. When the task environment is large, it is unreasonable to simply use the mean field theory to approximate the interactions between all UAVs, which may cause the algorithm to have difficulty in converging or even fail to converge.

Therefore, a partially observable mean field is introduced to set an approximate neighborhood R_a for each agent, and the range of R_a is adjusted according to the environment used by the algorithm. As shown in the Fig. 5, in the UAV swarm task environment, the neighborhood R_a can be equivalent to the inter-UAV communication area of the UAV. The UAVs in the communication area are the neighboring UAVs.

In mean field theory, considering the influence of all neighboring agents observed by UAV i, the Q value function $Q^i(\boldsymbol{s},\boldsymbol{a})$ of UAV i is expressed as the average sum of the Q value functions $Q^i(\boldsymbol{s},\boldsymbol{a}^i,\boldsymbol{a}^k)$ generated under the independent influence of each neighboring agent k:

$$Q^i(\boldsymbol{s},\boldsymbol{a}) = \frac{1}{N^i}\sum_k Q^i\left(\boldsymbol{s},\boldsymbol{a}^i,\boldsymbol{a}^k\right) \approx Q^i_{\mathrm{POMF}}\left(\boldsymbol{s},\boldsymbol{a}^i,\bar{\boldsymbol{a}}^i\right) \tag{10}$$

3.4. Multi-head attention mechanism

When using the mean field theory to average the action information of all neighboring agents, it is not appropriate to directly consider each agent as equally important. Therefore, the multi-head attention mechanism is introduced. In the process of calculating the average action in the partially observable mean field, the multi-head attention mechanism will perform self-attention calculations on each neighboring agent in the approximate neighborhood range, thereby considering the different weight effects between the agents and obtaining

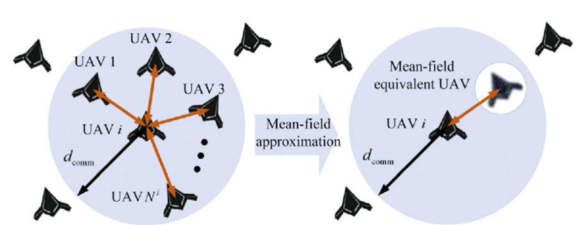


Fig. 5 Approximation effect of partially observable mean-field.

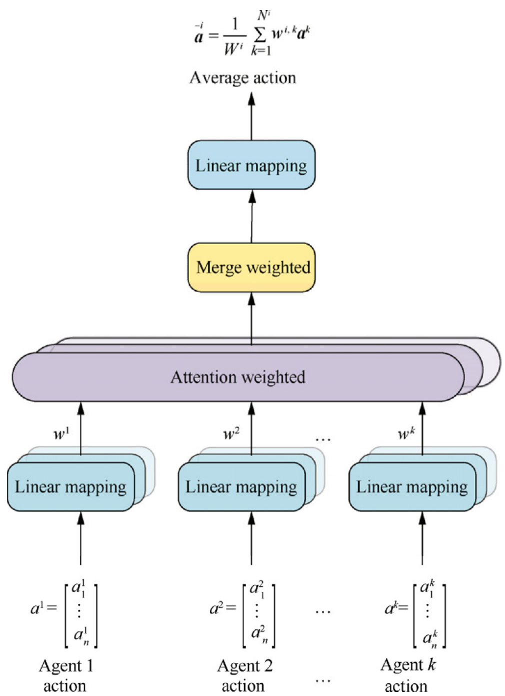


Fig. 6 Schematic of multi-head attention.

a partially observable weighted field, to enhance the robustness of the algorithm and improve the training performance of the algorithm, as shown in the Fig. 6.

In the partially observable mean field after adding the Multi-head Attention mechanism, the original average action is updated to the weighted average sum of the actions:

$$\bar{\boldsymbol{a}}^{i} = \frac{1}{W^{i}} \sum_{k=1}^{N^{i}} w^{i,k} \boldsymbol{a}^{k} \tag{11}$$

where $w^{i,k}$ is the weight coefficient between agents i and k, reflecting the intensity of the interaction between them.

3.5. PO-WMFDDPG algorithm

Based on the DDPG algorithm, we expand the discrete action space in MFQ to a continuous action space, and take advantage of the partially observable mean field theory to make the improved algorithm have broad application potential in large-scale agent reinforcement learning.

In the PO-WMFDDPG algorithm, the Q function in the DDPG algorithm[21] is updated to the partially observable weighted mean field Q function, and the actor_eval and critic_eval network loss functions are as follows:

$$\text{loss}_{\text{actor\_eval}} = -\text{mean}\left(Q^{i}_{\text{POMF}}\left(\boldsymbol{s}_t, \boldsymbol{a}^{i}_{t}, \bar{\boldsymbol{a}}^{i}_{t}; \boldsymbol{\theta}_{\text{critic\_eval}}\right)\right) \tag{12}$$

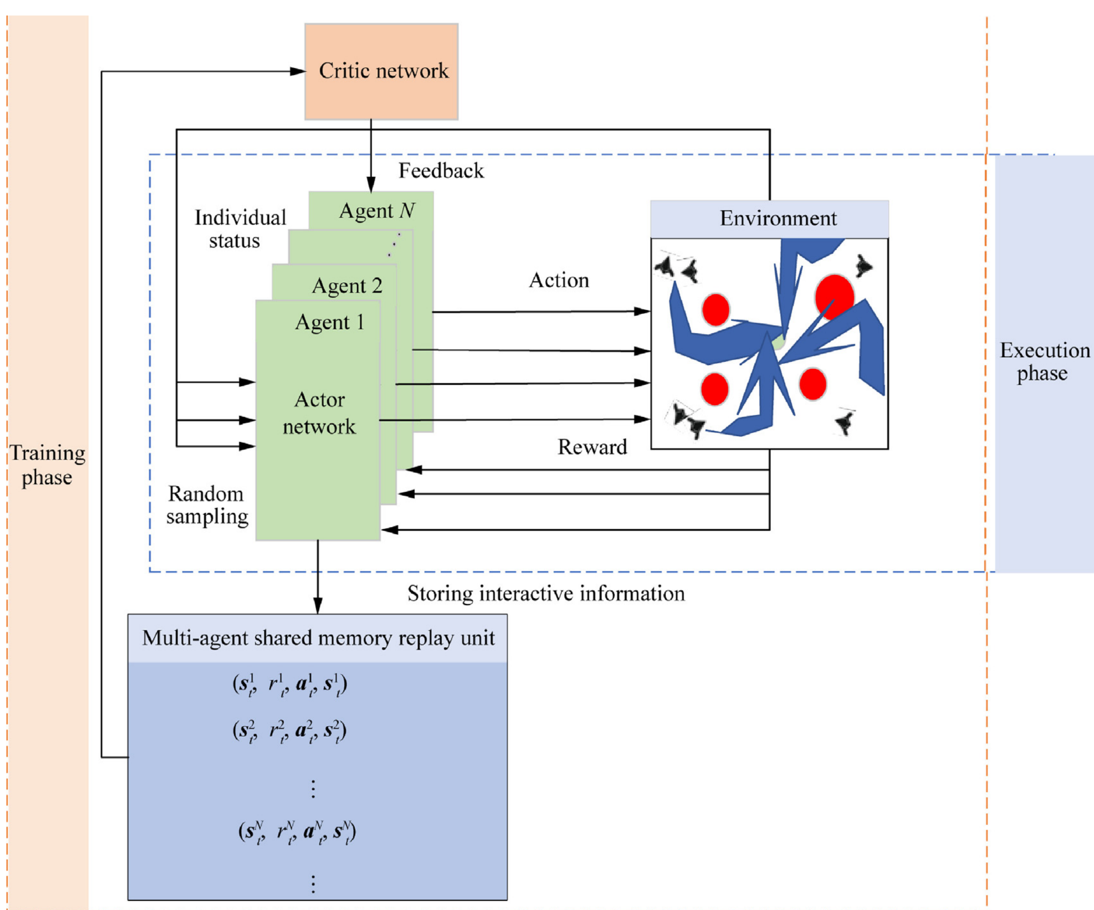


Fig. 7 Schematic of centralized training and distributed execution.

$$\begin{aligned}\text{loss}_{\text{critic\_eval}} = {} & r\left(\boldsymbol{s}_t^i, \boldsymbol{a}_t^i\right) + \\ & \gamma \cdot Q^{i'}_{\text{POMF}}\left(\boldsymbol{s}_{t+1}, \boldsymbol{a}^i_{t+1}, \bar{\boldsymbol{a}}^i_{t+1}; \boldsymbol{\theta}'_{\text{critic}}\right) - \\ & Q^i_{\text{POMF}}\left(\boldsymbol{s}_t, \boldsymbol{a}^i_t, \bar{\boldsymbol{a}}^i_t; \boldsymbol{\theta}'_{\text{critic}}\right)\end{aligned} \tag{13}$$

The critic_eval network update formula is as follows:

$$\begin{aligned}\delta_{\text{POMF}} = {} & R^i_t + \gamma Q^i_{\text{POMF}}\left(\boldsymbol{s}_{t+1}, \boldsymbol{a}^i_{t+1}, \bar{\boldsymbol{a}}^i_{t+1}\right) - \\ & Q^i_{\text{POMF}}\left(\boldsymbol{s}_t, \boldsymbol{a}^i_t, \bar{\boldsymbol{a}}^i_t\right)\end{aligned} \tag{14}$$

$$\boldsymbol{\theta}'_{\text{critic\_eval}} = \boldsymbol{\theta}_{\text{critic\_eval}} + \alpha \cdot \delta_{\text{POMF}} \tag{15}$$

where δ_{POMF} is the estimation error of the partially observable weighted mean field Q value; $Q^k_{\text{POMF}}\left(\boldsymbol{s}_t, \boldsymbol{s}_t, \bar{\boldsymbol{a}}^i_t\right)$ and $Q^k_{\text{POMF}}\left(\boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1}, \bar{\boldsymbol{a}}^i_{t+1}\right)$ are the partially observable mean field Q values of agent i after taking action considering the influence of the average action of neighboring agents at time t and $t+1$, that is, the value estimate.

The actor_eval network update formula is as follows:

$$\begin{aligned}\boldsymbol{\theta}'_{\text{actor\_eval}} \leftarrow {} & \boldsymbol{\theta}_{\text{actor\_eval}} + \alpha \cdot \nabla_{\theta_{\text{actor}}} \ln \pi_{\theta_{\text{actor}}}\left(\boldsymbol{s}_t, \boldsymbol{a}^i_t\right) \cdot \\ & Q^i_{\text{POMF}}\left(\boldsymbol{s}_t, \boldsymbol{a}^i_t, \bar{\boldsymbol{a}}^i_t\right)\end{aligned} \tag{16}$$

The critic_target and actor_target network parameter update formulas are as follows:

$$\boldsymbol{\theta}_{\text{critic\_target}} = \tau \boldsymbol{\theta}_{\text{critic\_eval}} + (1-\tau)\boldsymbol{\theta}_{\text{critic\_target}} \tag{17}$$

$$\boldsymbol{\theta}_{\text{actor\_target}} = \tau \boldsymbol{\theta}_{\text{actor\_eval}} + (1-\tau)\boldsymbol{\theta}_{\text{actor\_target}} \tag{18}$$

In order to improve the stability and efficiency of algorithm training, the "centralized training, distributed execution" framework[22] in the MADDPG algorithm is combined with the PO-WMFDDPG algorithm. The specific framework is shown in the Fig. 7.

"Centralized training" means that during the training phase of the algorithm, the experience data of all agents are concentrated into an experience playback buffer and used together for training the agents. This means that during the training process, each agent can share the experience of other agents to learn, thereby improving the efficiency of training. In the execution phase, each agent can interact with the environment independently and obtain independent evaluation values, thus avoiding too strong coupling between agents during training and improving the stability of training. This training and execution method enables agents to learn and make decisions in a multi-agent environment. The framework of the PO-WMFDDPG algorithm is shown in Fig. 8.

The pseudocode of PO-WMFDDPG algorithm is shown in Algorithm 1.

Algorithm 1. PO-WMFDDPG

1. Initialize the number of training rounds T, the maximum number of interactions per round $t_{\max}$, the number of agents N, the shared memory playback unit buffer, and the sampling size batch_size
2. Initialize reward discount factor γ, learning rate α, network parameter $\boldsymbol{\theta}_{\text{actor\_eval}}$, $\boldsymbol{\theta}_{\text{actor\_target}}$, $\boldsymbol{\theta}_{\text{critic\_eval}}$ and $\boldsymbol{\theta}_{\text{critic\_target}}$, target network update frequency T_{update}
3. for training round episode $= 1, 2, \cdots, T$:
4. Environment initialization, the initial state of the agent is $\boldsymbol{s}^i_t, i \in [1, 2, \cdots, N], t \in [1, 2, \cdots, t_{\max}]$
5. for number of interaction $t = 1, 2, \cdots, t_{\max}$:
6. Input $\boldsymbol{s}^i_t$ into the actor_eval network to get the agent action $\boldsymbol{a}^i_t = \mu_{\boldsymbol{\theta}_{\text{actor\_eval}}}\left(\boldsymbol{s}^i_t\right)$
7. According to OU noise, add noise to the actions output by the network $\boldsymbol{a}^i_t = \boldsymbol{a}^i_t + \delta_{\text{OU}}$
8. The agent performs action $\boldsymbol{a}^i_t$, interacts with the environment, and obtains interaction information r^i_t and $\boldsymbol{s}^i_{t+1}$
9 Update the multi-head attention weight matrix $\boldsymbol{W}$
10. According to the observable mean field theory, the influence of neighboring agents in the neighborhood $\boldsymbol{R}_a$ of agent i is approximated by the mean field, and the weighted average is calculated to obtain $\bar{\boldsymbol{a}}^i_t$
11. Store the interaction information $\left(\boldsymbol{s}^i_t, r^i_t, \boldsymbol{a}^i_t, \bar{\boldsymbol{a}}^i_t, \boldsymbol{s}^i_{t+1}\right)$ in buffer
12. Sampling from buffer to get batch_size training samples
13. Based on $\text{loss}_{\text{critic\_eval}} = \left[r(\boldsymbol{s}_t, \boldsymbol{a}_t) + \gamma \cdot Q'\left(\boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1}; \boldsymbol{\theta}_{\text{critic}_{\text{target}}}\right) - Q\left(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\theta}_{\text{critic}_{\text{eval}}}\right)\right]^2$ calculate the critic_eval network loss function and update the network parameter $\boldsymbol{\theta}_{\text{critic\_eval}}$
14. Based on $\text{loss}_{\text{actor\_eval}} = -\text{mean}(Q(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\theta}_{\text{critic\_eval}}))$ calculate actor_eval network loss function and update the network parameter $\boldsymbol{\theta}_{\text{actor\_eval}}$
15. if the target network update frequency T_{update} is reached:
16. Update the critic_target network parameter $\boldsymbol{\theta}_{\text{critic\_target}}$: $\boldsymbol{\theta}_{\text{critic\_target}} \leftarrow \tau \boldsymbol{\theta}_{\text{critic\_eval}} + (1-\tau)\boldsymbol{\theta}_{\text{critic\_target}}$
17. Update the actor_target network parameter $\boldsymbol{\theta}_{\text{actor\_target}}$: $\boldsymbol{\theta}_{\text{actor\_target}} \leftarrow \tau \boldsymbol{\theta}_{\text{actor\_eval}} + (1-\tau)\boldsymbol{\theta}_{\text{actor\_target}}$
18. end if
19. if this round has reached the end state:
20. break
21. end if
22. end for
23. end for

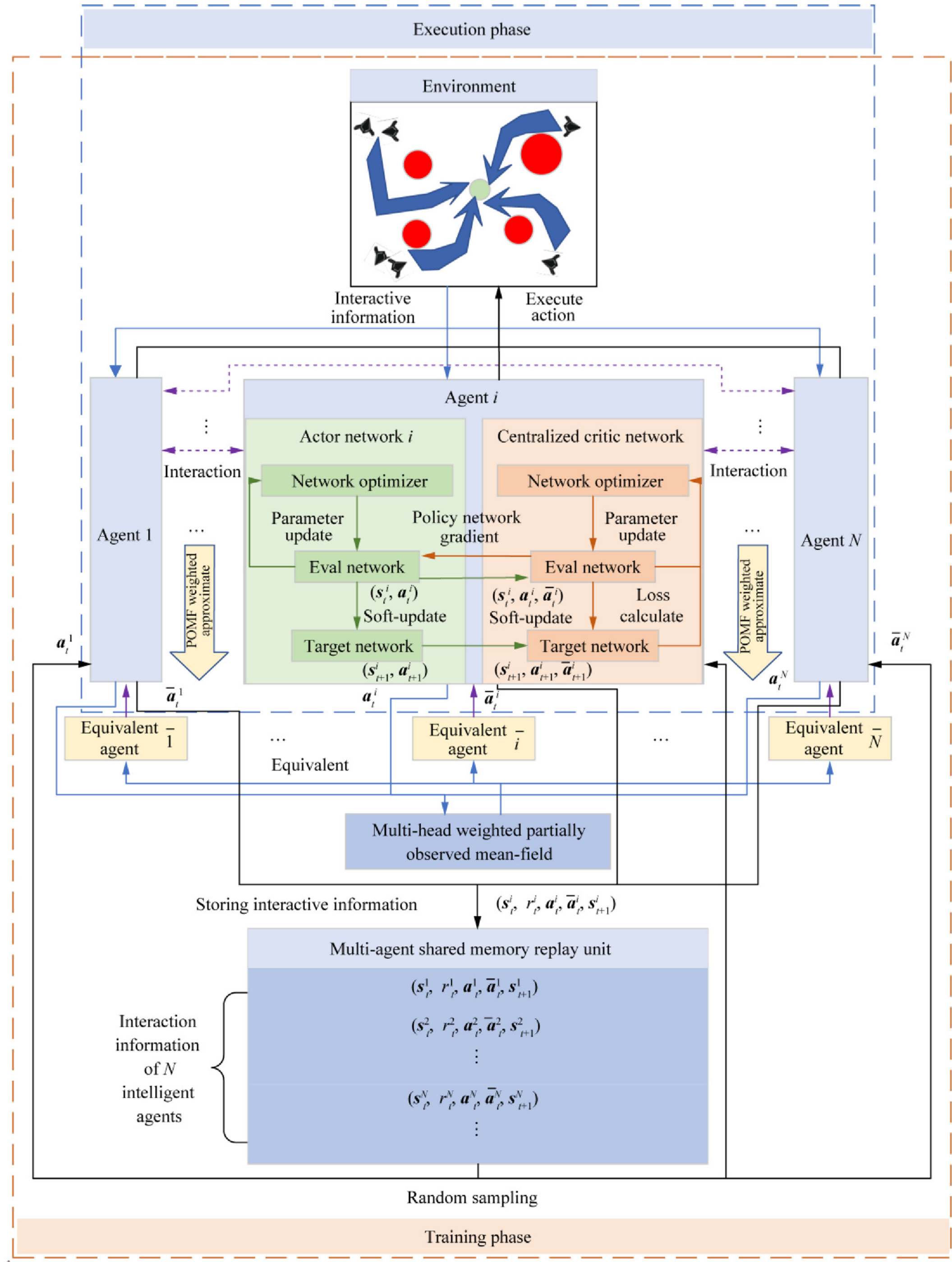


Fig. 8 Framework of PO-WMFDDPG algorithm.

4. UAV swarm path planning based on PO-WMFDDPG

4.1. Design of state and action space

The state space $\boldsymbol{S}^i$ of the UAV i includes: the UAV's own state information $\boldsymbol{I}^i_{\text{uav}}$, the UAV's task area state information $\boldsymbol{I}^i_{\text{task}}$, and the UAV's detection state information $\boldsymbol{I}^i_{\text{detect}}$:

$$\boldsymbol{S}^i = [\boldsymbol{I}^i_{\text{uav}}, \boldsymbol{I}^i_{\text{task}}, \boldsymbol{I}^i_{\text{detect}}] \tag{19}$$

The instantaneous state information $\boldsymbol{I}^i_{\text{uav}}$ of UAV i is as follows:

$$\boldsymbol{I}^i_{\text{uav}} = [x^i, y^i, v^i, \alpha^i] \tag{20}$$

Input the distance and azimuth of the task area relative to the UAV i as the task area state information $\boldsymbol{I}^i_{\text{task}}$ into the state space:

$$\boldsymbol{I}^i_{\text{task}} = [d^i, \beta^i] \tag{21}$$

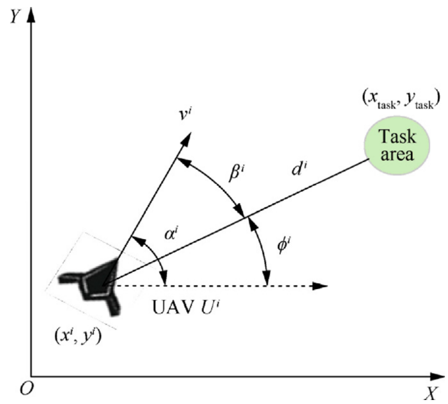


Fig. 9 Concept of UAV and task area.

$$\beta^i = \alpha^i - \phi^i \tag{22}$$

where I^i_{task} is the task area status information of UAV i, as shown in Fig. 9, d^i is the distance of UAV i relative to the task area, ϕ^i is the angle between UAV i and the center of the task area, and is defined as positive when rotating counterclockwise in the positive direction of the X-axis, β^i is the azimuth of the task area relative to UAV i, and is defined as positive when rotating counterclockwise in the direction of the UAV speed.

In the task path planning of large-scale UAV swarms, the stability inside the UAV swarm and the obstacle avoidance ability outside are important factors that must be considered. The UAV radar detection target schematic is shown in Fig. 10.

According to the radar tracking performance constraints, up to 5 targets can be tracked stably at the same time, and the detected target information $\boldsymbol{I}^i_{detect}$ is input into the state space:

$$\boldsymbol{I}^i_{detect} = \left[d^{i1}, \beta^{i1}, d^{i2}, \beta^{i2}, d^{i3}, \beta^{i3}, d^{i4}, \beta^{i4}, d^{i5}, \beta^{i5}\right] \tag{23}$$

where $\boldsymbol{I}^i_{detect}$ is the detection information of UAV i; d^{i1}, d^{i2}, d^{i3}, d^{i4} and d^{i5} are the distances of the detected targets, respectively, β^{i1}, β^{i2}, β^{i3}, β^{i4} and β^{i5} are the azimuth angles of the detected targets. If UAV i detects less than 5 targets, the corresponding information is set to 0.

In the UAV model linear acceleration α^i_v and angular acceleration α^i_α are used to control the motion of UAV i. The two parameters regulate the velocity magnitude and velocity direction of the UAV. Therefore, the two parameters are used as the action space of UAV i:

$$\boldsymbol{A}^i = \left[a^i_v, a^i_\alpha\right] \tag{24}$$

4.2. Design of neural network structure

The neural network framework of the PO-WMFDDPG algorithm includes four neural networks: actor_eval, actor_target, critic_eval and critic_target. After the introduction of mean field theory, the input of the critic_eval and critic_target networks include not only the state $\boldsymbol{s}^i$ and action $\boldsymbol{a}^i$, but also the average action $\bar{\boldsymbol{a}}^i$ of neighboring UAVs.

The PO-WMFDDPG algorithm network neuron design is shown in the Fig. 11.

The neural network structure of actor_eval and actor_target is the same. The network adopts a feedforward neural network. The input layer dimension is s_dim + a_dim + ā_dim = 20, which is the dimension of the state space. It passes through 3 hidden layers with dimensions of 64, 128, and 32 respectively. Finally, the UAV action selection is output through the output layer. The output layer dimension a_dim = 2 is the dimension of the action space.

The critic_eval and critic_target neural networks have the same structure. The network uses a feedforward neural network. The input layer dimension is s_dim + a_dim + ā_dim = 20, which includes the dimensions of the state space, action space, and average action of the neighbors. It passes through three hidden layers with dimensions of 64, 192, and 64 respectively, and finally passes through the output layer with a dimension of 1, which outputs the value evaluation of the UAV's current state-action pair.

4.3. Design of reward function

For the path planning task of UAV swarms, we design two types of reward functions, long-term rewards and immediate rewards. The long-term rewards r^i_{long} include arrival reward r^i_{goal}, out-of-bounds reward r^i_{bound}, and crash reward r^i_{crash}:

$$r^i_{long} = r^i_{goal} + r^i_{bound} + r^i_{crash} \tag{25}$$

When UAV i enters the task area, the UAV task is completed and receives the arrival reward r^i_{goal}; when UAV i flies out of the simulated battlefield environment, it receives a cer-

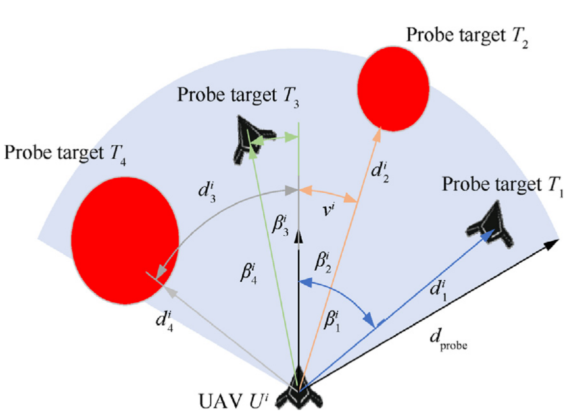


Fig. 10 Schematic of target detection by UAV.

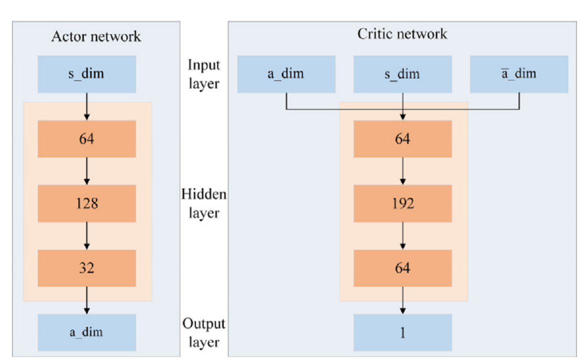


Fig. 11 Design of PO-WMFDDPG algorithm network.

tain penalty value r^i_{bound}; when UAV i collides with other UAVs or flies into the NFZ, the UAV crashes and receives a certain penalty value r^i_{crash}.

The instant reward function r^i_{step} of UAV i includes the instant path-finding reward function r^i_{task}, the instant collision reward function $r^i_{\text{collision}}$ and the single-step time penalty r^i_{time}, as shown below:

$$r^i_{\text{step}} = r^i_{\text{task}} + r^i_{\text{collision}} + r^i_{\text{time}} \tag{26}$$

The instant path-finding reward function r^i_{task} of UAV i is designed as follows:

$$r^i_{\text{task}} = \lambda\left(1 - \frac{d^i_{t+1}}{d^i_t}\right) + \eta\cos\beta^i_t \tag{27}$$

where d^i_t, d^i_{t+1} are the distances of the UAV relative to the task area at time t and $t+1$ respectively, and λ, η are importance factors.

The instant collision reward function $r^i_{\text{collision}}$ of UAV i is designed as follows:

$$r^i_{\text{collision}} = \begin{cases} \sum_{k=1}^{5}\frac{\lambda'}{k}\cdot\left(1-\frac{d^{ik}_t}{d^{ik}_{t+1}}\right), & d_{\text{danger}} \leqslant d^{ik}_{t+1} \leqslant d_{\text{safe}} \\ 0, & d^{ik}_{t+1} > d_{\text{safe}} \end{cases} \tag{28}$$

where d^{ik}_t and d^{ik}_{t+1} are the distances of UAV i relative to the detection target k at time t and $t+1$ respectively, and λ' is the importance factor. r^i_{time} is the single-step time penalty value of UAV i. A certain penalty value is given for each decision step to guide the UAV to reach the task area in the shortest time.

5. Simulation

5.1. Task scenario settings

In order to verify the effectiveness of PO-WMFDDPG, the UAV swarm task path planning is trained in a simulated battlefield environment. The major parameter settings during the algorithm training process are shown in Table 1.

5.2. Training process

The PO-WMFDDPG algorithm is used to train the path planning of the UAV swarm task. The changes in the loss values of the actor network and the critic network during the training process are shown in Figs. 12 and 13.

From the change curve of the loss function during the algorithm training process, we can see that the absolute value of the actor network loss function gradually decreases and eventually tends to stabilize and converge. The value of the critic network loss function shows an overall downward trend and converges during training. However, the actor network loss function and the critic network loss function fluctuate during the training process. This is because the probability of the UAV choosing random actions is relatively high in the early stage. As the number of training rounds increases, the UAVs gradually become intelligent. The probability of reaching the task area increases, and many new situations that could not be explored before are also explored. Therefore, the loss value will fluctuate with the change of the number of training rounds, but the overall trend is stable convergence.

Table 1 Training parameters for UAV swarm path planning.

Parameter	Value
Width of battlefield (m)	500
Length of battlefield (m)	500
Number of UAVs	80
UAV speed (m/s)	[1,2.5]
UAV linear acceleration (m/s^2)	[−1,1]
UAV angular acceleration (rad/s^2)	[−π/6, π/6]
Maximum distance UAV radar detection (m)	50
UAV radar detection angle (°)	75
Maximum distance of communication between UAVs (m)	50
Number of no-fly zones	20
Learning rate	0.005
Reward discount factor	0.98
Sampling sample size	128
Number of training rounds	1 000

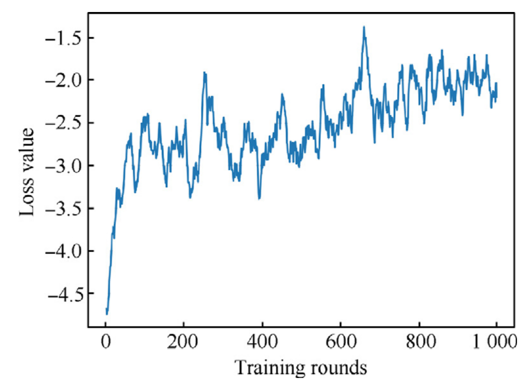


Fig. 12 Change of loss value of actor network in training.

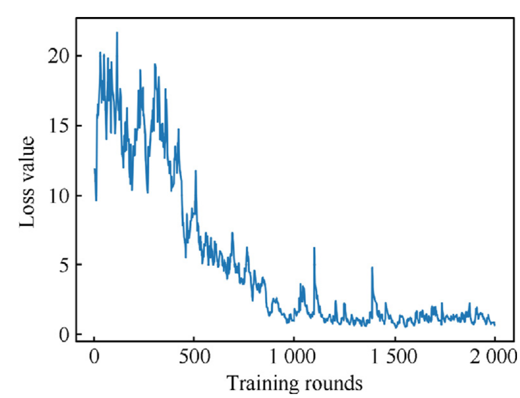


Fig. 13 Change of loss value of critic network in training.

During the training process of the algorithm, the average reward value of the UAV swarm is recorded, and the task success rate of the UAV swarm (the number of UAVs that successfully reach the task area/the total number of UAVs) is counted. The corresponding curve graphs are shown in Figs. 14 and 15.

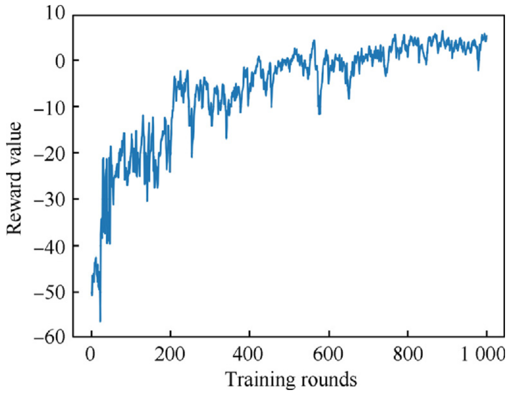


Fig. 14 Average reward value curve of UAV swarm in training.

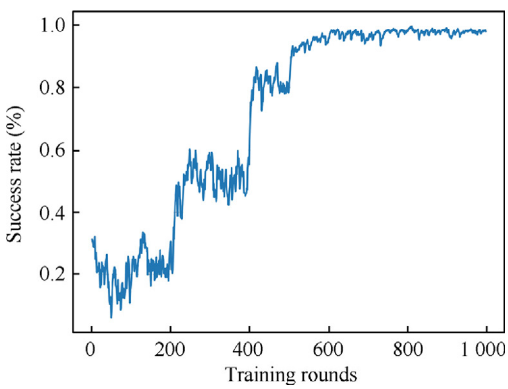


Fig. 15 Task success rate curve of UAV swarm in training.

The figures show that in the early stage of the training, the UAV is in the stage of actively exploring the environment. At this time, the reward value obtained by the UAV in a single round is a large negative value, and the success rate of the swarm task is also at a very low level. As the number of training rounds increases, the reward curve continues to rise, the task success rate steadily increases, the neural network gradually converges, and the UAVs learn certain experiences in the exploration process and gradually emerge with intelligence. After about 700 rounds of training, the reward value of the UAV changed from negative to positive and gradually stabilized. At the same time, the task success rate of the UAV swarm also stabilized at 98%.

In summary, in the task path planning training of UAV swarms using the PO-WMFDDPG algorithm, the network loss value converges stably, and the reward value and task success rate obtained by the UAV swarm gradually increase and stabilize, thus verifying the stability of the algorithm.

5.3. Verification process

The PO-WMFDDPG algorithm model is used for simulation verification to test the effectiveness of the algorithm. In a randomly generated task scenario, there are 80 UAVs and 20 NFZs. The simulation effect of task path planning is shown in Fig. 16.

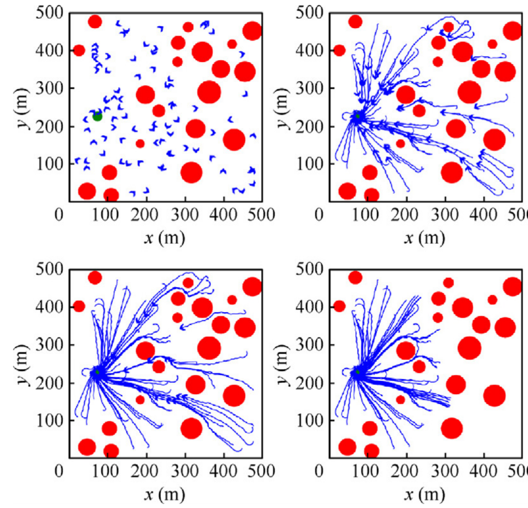


Fig. 16 Snapshots of UAV swarm path planning.

After training, the UAV swarm has shown good swarm intelligence. While maintaining internal stability of the swarm, it can avoid external NFZs. All 80 UAVs arrived at the task area safely.

In order to verify the execution effect of the algorithm under different task environment situations, the number of UAVs and the number of NFZs are changed. The task execution process is shown in Fig. 17. All UAVs can reach the task area safely.

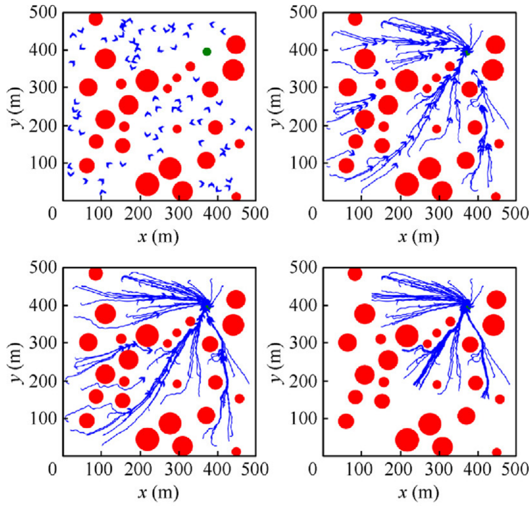
In order to verify the applicability of the UAV swarm path planning algorithm, the battlefield environment with no NFZ and mobile NFZs were constructed for testing. The task execution effect of the UAV swarm is shown in Figs. 18 and 19.

In the environment without NFZ, the task environment is simpler and the UAV swarm successfully completes the task. In a moving NFZs environment, the NFZ will move randomly within a certain range, which is a great test for the algorithm's obstacle avoidance ability. In the simulation, most UAVs can effectively avoid the moving NFZs and reach the task area successfully, but only 5 UAVs failed to evade the moving NFZs. Therefore, the feasibility and generalization ability of the PO-WMFDDPG algorithm model in large-scale UAV swarm path planning tasks is demonstrated.

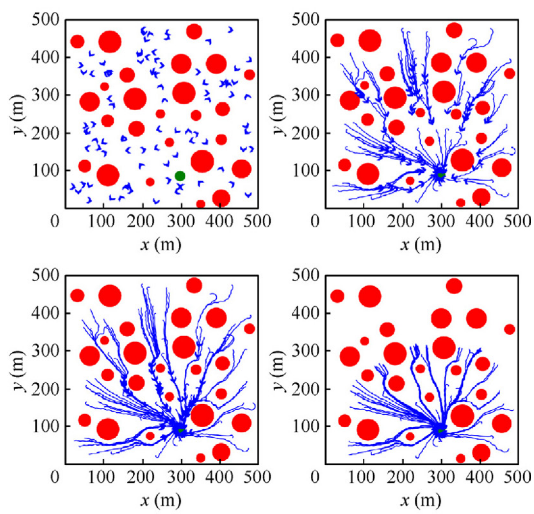
5.4. Algorithm comparison and analysis

In order to verify the performance of the algorithm, we use the DDPG algorithm and the MFDDPG (globally observable Mean Field DDPG) algorithm to train the UAV swarm task path planning and compare the training results with those of PO-WMFDDPG. In the experiment, the number of UAVs in the swarm is 80, and there are 20 NFZs. The average reward value and task success rate curves of the UAV swarm under the three different algorithm training are shown in Figs. 20 and 21.

It can be seen that in large-scale UAV swarm task training, using the MFDDPG and PO-WMFDDPG algorithms of the mean field theory, the reward value curve and the task success rate curve have a significant improvement after 200 rounds of training and reaches a stable value after 700 rounds of training. The reward value of the DDPG algorithm increased signifi-



(a) 70 UAVs, 25 no-fly zones



(b) 90 UAVs, 25 no-fly zones

Fig. 17 Snapshots of UAV swarm path planning in different situations.

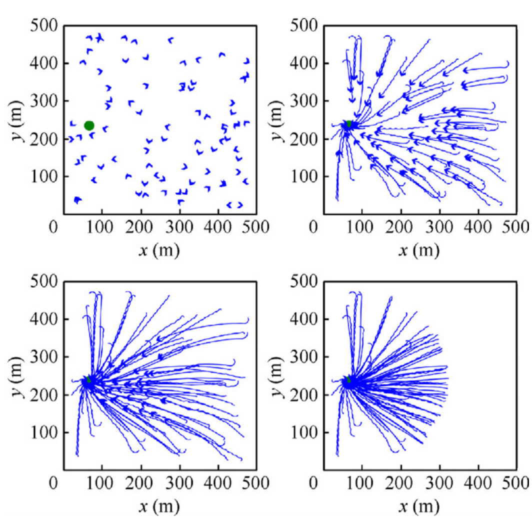


Fig. 18 Snapshots of UAV swarm path planning without NFZ.

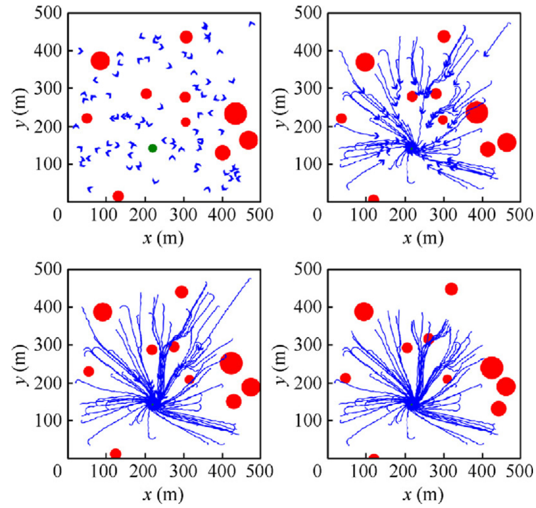


Fig. 19 Snapshots of UAV swarm path planning in moving NFZs environment.

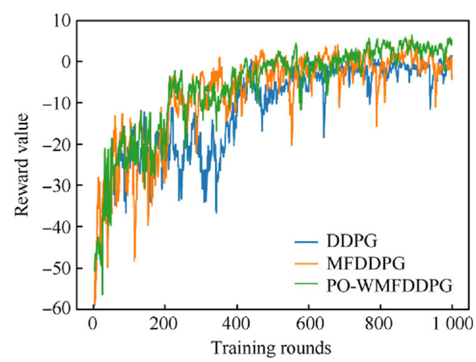


Fig. 20 Comparison of the average reward value of UAV swarm path planning in training.

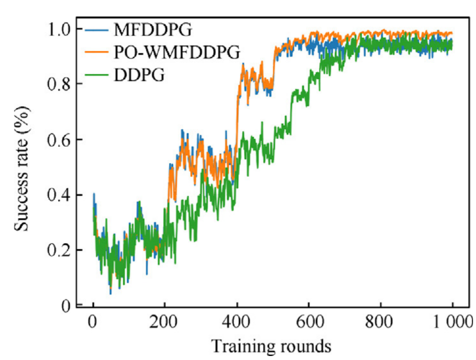


Fig. 21 Comparison of task success rate of UAV swarm path planning in training.

cantly after 400 rounds of training and finally stabilized after 800 rounds of training. After 1 000 rounds of training, the reward value and task success rate of the PO-WMFDDPG curve are better than those of the other two algorithms. The reward value and task success rate of the MFDDPG and DDPG algorithms are not much different after stable convergence. In large-scale UAV swarm task training, the partially observable mean field theory can effectively improve the training effect and network convergence speed.

Using the models trained by the three algorithms, there are 20 NFZs in the task environment. Only the number of UAVs in the swarm is changed for testing. The task success rate curve is plotted as shown in Fig. 22.

From the figure above, we can see that the UAV swarms after trained by the three algorithms, all perform well in UAV swarm task path planning, and the success rate of the task can reach more than 98% in all of them. Among them, the DDPG algorithm performs well when the number of UAVs is small, with a task success rate of up to 99%, which gradually decreases as the number of UAVs increases, and drops to 92.8% when the number of UAVs reaches 100. The MFDDPG algorithm incorporates the mean-field theory, which is rather less effective than the DDPG algorithm when the number of UAVs is small, and the task success rate gradually increases with the gradual increase of UAVs, and decreases after the number of UAVs reaches 50. The reason for this is that in MFDDPG it is a mean-field approximation of all UAV movements globally, and it is counterproductive to consider interactions between distant UAVs instead. The PO-WMFDDPG algorithm uses a partially observable mean-field approximation theory, which stabilizes the task success rate above 98% as the number of UAVs increases. When the number of UAVs reaches or exceeds 100, the task success rates of all three algorithms exhibit varying degrees of decline. Incorporating mean field game theory, both the PO-WMFDDPG algorithm and the MFDDPG algorithm continue to perform well when the swarm size exceeds 100. The PO-WMFDDPG algorithm takes into account the partial observability of individual agents, maintaining a task success rate above 90% when the number of agents ranges from 100 to 120, thus effectively executing path planning tasks for the UAV swarm. However, when the number of agents exceeds 120, the risk of collisions within the swarm significantly increases due to the large swarm size in a battlefield environment measuring 500×500, leading to a substantial decline in task success rates. The MFDDPG algorithm requires consideration of global individual information and demonstrates slightly inferior performance compared to the PO-WMFDDPG algorithm; the task success rate sharply declines when the number of agents exceeds 110. Meanwhile, the DDPG algorithm suffers from the curse of dimensionality due to the increase in swarm size, resulting in a surge in computational demands that deteriorate algorithm performance. When the number of agents exceeds 100, the task success rate decreases significantly, and at 200 agents, it becomes impossible to conduct path planning tasks for the UAV swarm. Overall, as the number of UAVs increases, the performance of the PO-WMFDDPG algorithm remains stable within a certain range, clearly outperforming the other two algorithms. However, within a fixed-sized battlefield environment, the increase in the number of UAVs must be constrained; otherwise, it will severely impact the execution of the algorithm.

In order to compare and validate the generalization capabilities of the three algorithms, a drone swarm of 80 units was established, and the number of fixed no-fly zones was varied. The trained PO-WMFDDPG algorithm, MFDDPG algorithm, and DDPG algorithm models were employed to perform maneuvering decision-making for large-scale drone swarm task path planning. The variation curves of task success rates corresponding to the increasing number of no-fly zones for the three algorithms are presented as Fig. 23.

When no no-fly zones are present, the battlefield environment is significantly simplified, allowing the PO-WMFDDPG algorithm, the MFDDPG algorithm, and the DDPG algorithm to maintain a high success rate in task execution. The PO-WMFDDPG algorithm can achieve a 100% task success rate, while the other two algorithms can sustain success rates exceeding 98%. As the number of no-fly zones increases, the complexity of the environment escalates, requiring individual drones not only to maintain internal cluster stability but also to avoid collisions with external no-fly zones. When the number of no-fly zones is less than or equal to 20, the PO-WMFDDPG algorithm's task success rate remains around 98%, with the algorithm's performance remaining relatively stable, successfully executing the path planning for drone swarms. However, when the number of no-fly zones exceeds 20, the task success rate of the PO-WMFDDPG algorithm experiences a slight decline, and this decline becomes more pronounced when the number of no-fly zones surpasses 36, due to the overly dense distribution of no-fly zones. The MFDDPG algorithm exhibits an overall trend that is slightly lower than that of the PO-WMFDDPG algorithm, with a sharp decrease in task success rate when the number of no-fly zones exceeds 30, resulting in poor task execution performance. The DDPG algorithm experiences a rapid decline in task success rate when the number of no-fly zones exceeds 26, performing worse compared to the other two algorithms. In summary, the PO-WMFDDPG algorithm demonstrates a

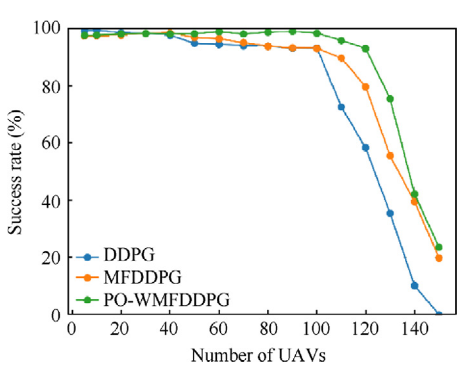


Fig. 22 Success rate curves of path planning under different number of UAVs.

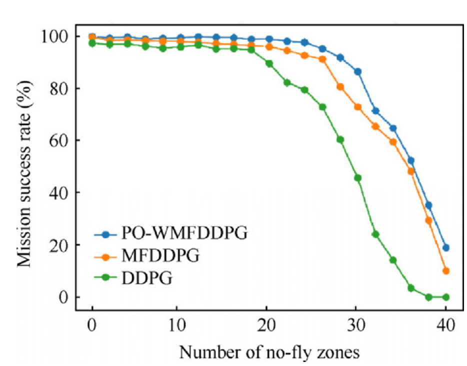


Fig. 23 Success rate curves of path planning under different number of no-fly zones.

stronger generalization capability, ensuring effective task execution even in novel environments, and its performance is significantly superior to that of the MFDDPG and DDPG algorithms.

In summary, the PO-WMFDDPG algorithm demonstrates superior performance in task path planning for large-scale UAV swarms. The algorithm not only outperforms the MFDDPG and DDPG algorithms in terms of convergence speed and task success rate but also exhibits strong generalizability. The PO-WMFDDPG algorithm effectively integrates the Multi-head Attention mechanism to account for varying influence weights among agents, thereby enhancing its robustness and adaptability to complex task scenarios. This results in more efficient and reliable path planning, making it a promising approach for managing the dynamic and uncertain environments encountered by UAV swarms. The ability to handle partial observability and incorporate weighted mean field calculations further underscores its effectiveness in real-world applications, ensuring consistent and successful task completion.

6. Conclusions

In this paper, a novel Partially Observable Weighted Mean Field Deep Deterministic Policy Gradient (PO-WMFDDPG) is proposed for the problem of task path planning in large-scale UAV swarm operations. The battlefield environment is built to model the motion, detection, and communication of UAVs. Individual UAVs are transformed into agents in RL, combined with the PO-WMFDDPG algorithm, which is used to solve the task path planning problem of UAV swarms. The effectiveness and generalization of the PO-WMFDDPG algorithm are verified in the simulation experiments. In comparison the PO-WMFDDPG algorithm with the DDPG algorithm and the MFDDPG algorithm, the PO-WMFDDPG algorithm outperforms the other two algorithms in terms of task success rate and convergence speed in the context of large-scale UAV swarm.

CRediT authorship contribution statement

Yaozhong ZHANG: Writing – review & editing, Writing – original draft, Project administration, Methodology, Conceptualization. **Meiyan DING:** Methodology. **Yao YUAN:** Software. **Jiandong ZHANG:** Project administration. **Qiming YANG:** Funding acquisition. **Guoqing SHI:** Data curation. **Jianming JIANG:** Visualization, Resources.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

This work was supported by the Aeronautical Science Foundation of China (20220013053005)

References

1. Wang C, Wang JQ, Wei CY, et al. Vision-based deep reinforcement learning of UAV-UGV collaborative landing policy using automatic curriculum. *Drones* 2023;**7**(11):676.
2. Tang J, Liang YG, Li KB. Dynamic scene path planning of UAVs based on deep reinforcement learning. *Drones* 2024;**8**(2):60.
3. H. Gao, Y.H. Jia, L.W. Xu, et al., Anytime algorithm based on adaptive variable-step-size mechanism for path planning of UAVs, Chin J Aeronaut, 2024. https://doi.org/10.1016/j.cja.2024.09.007.
4. Lv H, Chen YD, Li SB, et al. Improve exploration in deep reinforcement learning for UAV path planning using state and action entropy. *Meas Sci Technol* 2024;**35**(5):056206.
5. Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. *Nature* 2015;**518**(7540):529–33.
6. Liu Q, Zhai J, Zhang Z, et al. A review of deep reinforcement learning. *Journal of Computing* 2018;**41**(1):1–27.
7. Zhan G, Zhang XM, Li ZC, et al. Multiple-UAV reinforcement learning algorithm based on improved PPO in ray framework. *Drones* 2022;**6**(7):166.
8. Li SW, Jia YH, Yang F, et al. Collaborative decision-making method for multi-UAV based on multiagent reinforcement learning. *IEEE Access* 2022;**10**:91385–96.
9. Lillicrap TP, Hunt JJ, Pritzel A, et al. Continuous control with deep reinforcement learning. arXiv preprint:1509.02971; 2015.
10. Silver D, Huang A, Maddison CJ, et al. Mastering the game of Go with deep neural networks and tree search. *Nature* 2016;**529**(7587):484–9.
11. Vinyals O, Babuschkin I, Czarnecki WM, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 2019;**575**(7782):350–4.
12. Nowé A, Vrancx P, De Hauwere Y M. Game theory and multi-agent reinforcement learning. *Reinforcement Learning*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 441–70.
13. Yang YD, Luo R, Li M, et al. Mean field multi-agent reinforcement learning. arXiv preprint: 1802.05438; 2018.
14. uz Zaman MA, Zhang KQ, Miehling E, et al. Reinforcement learning in non-stationary discrete-time linear-quadratic mean-field games. *2020 59th IEEE conference on decision and control (CDC)*. Piscataway: IEEE Press; 2020.
15. Cui K, Tahir A, Sinzger M, et al. Discrete-time mean field control with environment states. *2021 60th IEEE Conference on decision and control (CDC)*. Piscataway: IEEE Press; 2021.
16. Wu CX, Ju BB, Wu Y, et al. UAV autonomous target search based on deep reinforcement learning in complex disaster scene. *IEEE Access* 2019;**7**:117227–45.
17. Zhang YS, Zu W, Gao Y, et al. Research on autonomous maneuvering decision of UCAV based on deep reinforcement learning. *2018 Chinese control and decision conference (CCDC)*. Piscataway: IEEE Press; 2018.
18. Qiu C, Hu Y, Chen Y, et al. Deep deterministic policy gradient (DDPG)-based energy harvesting wireless communications. *IEEE Internet Things J* 2019;**6**(5):8577–88.
19. Demkowicz L, Gopalakrishnan J, Nagaraj S, et al. A spacetime DPG method for the Schrodinger equation. *SIAM J Numer Anal* 2017;**55**(4):1740–59.
20. Liu L, Gao F, Luo XL. A review of deep reinforcement learning based on value function and policy gradient. *J Comput* 2019;**42**(6):1406–38.
21. Zhang YZ, Xu JJ, Yao K, et al. UAV swarm pursuit task based on DDPG algorithm. *J Aeronaut* 2020;**41**(10):314–26.
22. LX Xu, WB Chen, X Liu, et al., MADDPG, *Multi-agent deep deterministic policy gradient algorithm for formation elliptical encirclement and collision avoidance*. *Proceedings of 2021 5th Chinese conference on swarm intelligence and cooperative control*, 2022, Springer Nature Singapore; Singapore.