



Research Article

UAV Swarm Confrontation Using Hierarchical Multiagent Reinforcement Learning

Baolai Wang ¹, Shengang Li ¹, Xianzhong Gao ², and Tao Xie ¹

¹College of Computer, National University of Defense Technology, Changsha 410073, China

²College of Aerospace Science, National University of Defense Technology, Changsha 410073, China

Correspondence should be addressed to Tao Xie; hamishxie@vip.sina.com

Received 30 September 2021; Revised 12 November 2021; Accepted 18 November 2021; Published 21 December 2021

Academic Editor: Xingling Shao

Copyright © 2021 Baolai Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the development of unmanned aerial vehicle (UAV) technology, UAV swarm confrontation has attracted many researchers' attention. However, the situation faced by the UAV swarm has substantial uncertainty and dynamic variability. The state space and action space increase exponentially with the number of UAVs, so that autonomous decision-making becomes a difficult problem in the confrontation environment. In this paper, a multiagent reinforcement learning method with macro action and human expertise is proposed for autonomous decision-making of UAVs. In the proposed approach, UAV swarm is modeled as a large multiagent system (MAS) with an individual UAV as an agent, and the sequential decision-making problem in swarm confrontation is modeled as a Markov decision process. Agents in the proposed method are trained based on the macro actions, where sparse and delayed rewards, large state space, and action space are effectively overcome. The key to the success of this method is the generation of the macro actions that allow the high-level policy to find a near-optimal solution. In this paper, we further leverage human expertise to design a set of good macro actions. Extensive empirical experiments in our constructed swarm confrontation environment show that our method performs better than the other algorithms.

1. Introduction

UAVs have the advantages of low cost, flexible maneuverability, strong concealment, and ability to perform in harsh environments. Therefore, they are widely used in the military field and have played an essential role in many local wars. Nevertheless, a single UAV is limited by its detection range and weapon load, making it difficult to complete complex tasks. The use of multiple UAVs, which is in coordination through communication between UAVs, can expand the perception of the environmental situation, achieve coordinated task assignment, collaborative reconnaissance, and attack, and effectively improve the survivability and overall combat effectiveness [1]. This environment, where multiple UAVs cooperate with allies and compete with the enemies, naturally constitutes a multiagent system. In such a dynamically changing environment, training UAVs to complete complex tasks together has essential research significance.

The existing UAV swarm confrontation methods include the differential game-based methods, the expert system-based methods, and the guidance law-based methods. These methods have good performance in a simple, small-scale scenario, but they are not suitable for large-scale, complex, and unknown scenarios. In recent years, as a way to solve decision-making problems, reinforcement learning is widely used in single-agent confrontation environments, such as Atari games [2] and Go [3]. Compared with traditional algorithms such as differential games and expert systems, reinforcement learning does not rely on environmental models. Instead, it mainly optimizes its policy through the reward function, which has more significant advantages in large-scale and complex environments. Therefore, using reinforcement learning-related theories to solve UAV swarm confrontation is a promising method. However, single-agent reinforcement learning (RL) methods, such as DQN [2] and DDPG [4], are not suitable for MAS since the state-action

space increase exponentially with the number of UAVs. Moreover, single RL may suffer from nonstationary due to the concurrency of multiagent policy updates, resulting in ineffective cooperation between multiple agents.

To address these problems, researchers extend single-agent RL algorithms to multiagent settings. For example, Lowe et al. [5] propose a multiagent deep deterministic policy gradient (MADDPG) algorithm, an extension of the DDPG algorithm. Following the centralized learning and decentralized execution paradigm, each agent in MADDPG needs to input all agents' local observations and actions during training, while only the local observation of each agent needs to be input during execution. Besides, other algorithms such as DIAL [6] and ATOC [7] have been proposed to learn communication to promote cooperation between agents.

However, it is challenging to design immediate rewards for evaluating the quality of actions in the UAV swarm confrontation environment. Therefore, only sparse rewards are used to guide the training of policies. As a result, most MARL algorithms are hard to learn effective cooperation policies. In addition, as the joint action space increases exponentially with the number of UAVs, the space that the MARL algorithm needs to explore is increased dramatically, leading to a longer algorithm convergence time. To solve these problems, we propose a *hierarchical multiagent deep deterministic policy gradient* (h-MADDPG) method for UAV swarm confrontation. With temporal abstraction, the difficulties of learning with sparse reward are reduced by decomposing the original problem into multiple levels of abstractions. Furthermore, to reduce the action space that needs to be explored by agents, we leverage human expertise to abstract the primitive action spaces into a set of macro actions. Experimental results show that cooperative strategies are easier to learn based on macro actions at a high level.

This paper proposes a new MARL algorithm in a UAV swarm confrontation scenario with sparse reward and large action space. We show that temporal abstraction and macro action can facilitate MARL policy learning. Our contributions can be summarized as follows:

- (i) We design a hierarchical MARL framework for UAV swarm confrontation, and its high-level policy makes decisions at a relatively low frequency while the low-level policy makes decisions at every time step, which can effectively solve the sparse and delayed reward problem in the UAV swarm confrontation environment
- (ii) A set of macro actions is designed through human expertise so that the action space explored by agents is reduced, which also helps RL agent learn effective cooperation strategies at a higher level and improves the interpretability of selected actions
- (iii) We compared our method with several other RL algorithms in our constructed UAV swarm confrontation environment. The experimental results show that our algorithm can greatly improve the winning rate of UAV confrontation, learn effective cooperation strategies, and increase the convergent speed.

The rest of this paper is organized as follows. In the next section, we present the related work. Then, preliminaries are provided in Section 3, and Section 4 explains our proposed h-MADDPG in detail. Next, experiment and results analysis are presented in Section 5. Finally, we conclude this paper in Section 6.

2. Related Work

UAV swarm confrontation is mainly studied with heuristic algorithms and nonlearning strategies [8], but these algorithms have certain limitations. For example, the guidance law method is an algorithm that guides the UAV to the designated target point; although it is simple and easy to implement, it needs to know the enemy's control strategy or dynamic model in advance, which is impractical in actual confrontation tasks. The differential game method can learn how to act without prior knowledge of the agent, but this method has many problems such as too many state variables, complex differential equations, and challenging to solve the analytical equations, so it is difficult to be applied to complex multiagent environments. Expert system method is based on the prior knowledge of human experts in related fields to establish its system model, and according to the current state of UAV, through fuzzy matching method to select the predefined action strategy in the knowledge base, it depends on the rules made by a large number of human experts; in the complex environment of UAV cluster confrontation, it is impossible to make rules and ensure the optimality of decision.

Reinforcement learning is a method that does not rely on any model or prior knowledge and optimizes its behavior by exploring to maximize the reward function. In recent years, MARL has attracted wide attention from researchers, which provides a new idea for swarm confrontation [9]. The UAV swarm can be modeled as a multiagent system. Each agent needs to cooperate with teammates to defeat the enemy to be regarded as a cooperative multiagent system. The natural way to solve the multiagent decision-making problem is to directly use the single-agent RL algorithm in the multiagent system. Tampuu et al. [10] trained each agent with an independent Q-network in the ALE [11] environment, where each agent maximizes its own expected return. Since the state transition and reward of the environment depend on all agents, once an agent's policy changes, this will result in a nonstationary environment. Besides, this method cannot promote practical cooperation between agents.

Researchers have proposed two development routes to overcome these problems: value decomposition (VD) and actor-critic (AC) methods. Representative VD methods include VDN [12] and QMIX [13]. The basic idea of the VDN method is to centrally train a joint Q-network, which is obtained by adding the local Q-networks of all agents. In this way, the nonstationary environment can be handled through centralized training, and the complex interrelationships between agents can be decoupled. The QMIX algorithm is an improvement of the VDN. Instead of sum, QMIX uses a mixing network to combine the local Q-values,

representing the complex interrelationships between agents. Jiang et al. [14] combines the VD and the group communication method to ensure cooperation between the various groups. The AC algorithm uses the framework of centralized learning and decentralized execution (CTDE), which addresses the nonstationary environment and makes the system scalable; representative algorithms are MADDPG [5] and MAAC [15]. MADDPG assumes that each agent can access all other agents' local observations and actions during training so that it can deal with a nonstationary environment. For MADDPG, the critic network corresponding to each agent inputs the other agents' local observations and actions indiscriminately. Nevertheless, in real-world applications, the agent pays extra attention to the other agents, for which MAAC introduces the attention mechanism into the construction of the Q function. Tang et al. [16] propose hierarchical deep MARL with temporal abstraction in a cooperative environment, in which agents can learn effective cooperation strategies under different time scales. Inspired by the feudal RL [17] architecture, Ahilan and Dayan [18] propose feudal multiagent hierarchies (FMH) to promote cooperation between agents. In this research, a "manager" agent is responsible for setting goals for multiple "workers" to maximize extrinsic rewards, and "workers" are rewarded for completing the set goals.

There are also two technical methods to apply reinforcement learning to UAV swarm confrontation. One method is to regard each UAV as an agent, where each UAV can only obtain environmental information through local observations, and UAVs are independent of each other. For example, Yang et al. [19] propose an UAV air combat autonomous maneuver decision algorithm based on DDPG. However, this method ignores the interactions between UAVs, which greatly reduces the cooperation between UAVs, so it does not perform well in complex scenes. Another method is to adopt CTDE, where UAVs can access the action and state information of all other allies during training, while only local observations are required during policy execution, which can not only improve the cooperation between UAVs but also effectively reduce the communication burden. Xuan and Ke [20] realize the autonomous decision-making control of UAV without communication based on the MADDPG algorithm in confrontation environment. Afterwards, they extend this framework to the MAPPO algorithm [21]. Unfortunately, this method is only suitable for small-scale UAV clusters. When the cluster scale is large, the model will suffer from the curse of dimensionality. At the same time, sparse reward is also a challenge to apply reinforcement learning to UAV swarm confrontation. Compared with the above methods, our algorithm can not only improve the cooperation ability between UAVs but also deal with the problems of curse of dimensionality and sparse reward.

3. Preliminary

In this section, we first define the problem to be solved by MARL and the flight constraints of UAVs. Then, we provide a multiagent model permitting macro actions, a particular

case of semi-Markov decision processes (semi-MDP). Finally, we define macro actions and explain MADDPG in detail.

3.1. Problem Definition. In this work, we construct a UAV swarm confrontation environment based on the multiagent particle environment [22] used in MADDPG [5]. In this confrontation scenario, the red and blue UAV swarms compete against each other, and the one that destroys more enemy UAVs within the specified time wins. For simplicity, the UAV swarm of the red side and the blue side is homogeneous. The UAV follows the following constraints in the confrontation process.

3.1.1. Initial Coordinates. In the initial state, the two groups of UAVs are located on both sides of the target area and they are in a line, respectively. The position of UAV i of a group is $(x_{\min}, y_{\min} + i \times \Delta_y)$, and the position of UAV j of the another group is $(x_{\max}, y_{\min} + j \times \Delta_y)$, where Δ_y is the interval between the vertical coordinates of adjacent UAVs.

3.1.2. Height and Boundary Constraints. Since the simulation environment is two-dimensional, the height of UAV is limited to the same plane. In addition, the target area is bounded, and the UAV cannot exceed its range, that is the coordinates of UAV i need to meet $-4000\text{m} < x_i, y_i < 4000\text{m}$.

3.1.3. Velocity and Acceleration Constraints. Limited by the performance of UAV, the speed and acceleration of UAV i should also meet certain restrictions $20\text{ m/s} < v_i < 40\text{ m/s}$, $-5\text{ m/s}^2 < a_i < 5\text{ m/s}^2$.

3.1.4. Obstacle Constraint. In order to be closer to the real environment, we randomly set two obstacles in the initial state. During flight, the UAV cannot collide with obstacles; otherwise, it will crash. The distance d_{ou} between UAV and obstacle should meet $d_{ou} > R_o + R_u$, where R_o is the radius of the obstacle and R_u is the radius of the UAV. The algorithm evaluation in Section 5.2 will be carried out in the environment with and without obstacles, respectively.

The following four properties can describe each UAV (agent) in this scenario: position (x, y) , speed v , heading φ , and attacking zone (R, β) , where the attacking zone is a sector of (R, β) in front of the UAV. The heading and speed determine the motion state of the UAV in the environment. Therefore, the update of UAV status follows the following rules:

$$\varphi_{t+1} = \varphi_t + p_t \times \varphi_{\max}, \quad (1a)$$

$$v_{t+1} = v_t + a_t \times dt, \quad (1b)$$

$$x_{t+1} = x_t + v_t \times \cos(\varphi_t) \times dt, \quad (1c)$$

$$y_{t+1} = y_t + v_t \times \sin(\varphi_t) \times dt, \quad (1d)$$

where p and a represent the heading change and acceleration of the UAV, respectively, which are output by the UAV control module, φ_{\max} is the maximum angle of UAV heading change at each time step, defined as 30

degrees in this paper, and dt is the time interval of the simulation, defined as 1 second.

At any time t , the relationship between the UAV and the target within its detection range can be represented by $(\delta_{ij}(t), d_{ij}(t), \theta_{ij}(t))$, and t is omitted for clarity when there is no ambiguity. As illustrated in Figure 1, the coordinate of the red UAV i is (x_i, y_i) , and the blue one is (x_j, y_j) . The relative position between the two UAVs is δ_{ij} , and the Euclidean distance between them is d_{ij} . θ_{ij} is the attacking angle of red UAV i relative to blue UAV j . They are calculated by the following equation.

$$\begin{aligned} \delta_{ij} &= (x_j - x_i, y_j - y_i), \\ d_{ij} &= \|\delta_{ij}\|_2, \\ \theta_{ij} &= \arctan \frac{\delta_{ij}^y}{\delta_{ij}^x}. \end{aligned} \quad (2)$$

Only when the distance and attacking angle between the attacker i and the target j satisfy the following conditions, the attacker i has a certain probability of destroying another UAV:

$$\begin{aligned} d_{ij} &< R, \\ \theta_{ij} &< \frac{\beta}{2}. \end{aligned} \quad (3)$$

3.2. Markov Games. In this paper, UAV swarm confrontation is modeled as a partially observable Markov games [23] augmented with a set of macro actions. Formally, the Markov games can be represented as a tuple $\langle \mathcal{F}, \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{X}, \Omega, \mathcal{T}, \mathcal{R} \rangle$, where \mathcal{F} is a finite set of N agents, \mathcal{S} is a set of global states, $\mathcal{A} = \{\mathcal{A}_i\}_{i=0}^N$ is a set of joint primitive actions, $\mathcal{O} = \{\mathcal{O}_i\}_{i=0}^N$ is a set of joint local observations, $\mathcal{X} = \{\mathcal{X}_i\}_{i=0}^N$ is a set of joint intrinsic observations derived from local observations, and Ω is a set of macro actions abstracted with human expertise. Each agent i receives local observation $o_i \in \mathcal{O}_i$ from the environment. The policy of agent i depends on the local observation such that $\pi_i^h : \mathcal{O}_i \times \Omega \rightarrow [0, 1]$, then conditioned on the macro action $\pi_i^l : \mathcal{X}_i \times \Omega \rightarrow [0, 1]$. According to the state transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A}_1 \times \dots \times \mathcal{A}_N \times \mathcal{S} \rightarrow [0, 1]$, the environment transitions to the next state. $\mathcal{R} : \mathcal{S} \times \mathcal{A}_i \rightarrow \mathbb{R}$ is a reward function assigning a extrinsic reward r_i for taking primitive action. Each agent is aimed at maximizing its own discounted sum of future rewards $\sum_{t=0}^{\infty} \gamma^t r_{i,t}$ where γ is a discount factor.

3.3. Macro Actions. A macro action is defined as a finite sequence of primitive actions. Once the agent selects a macro action, it selects the primitive actions following the macro action until the termination condition of the macro action is satisfied. For example, chasing an enemy UAV could be a macro action. This macro action enables the

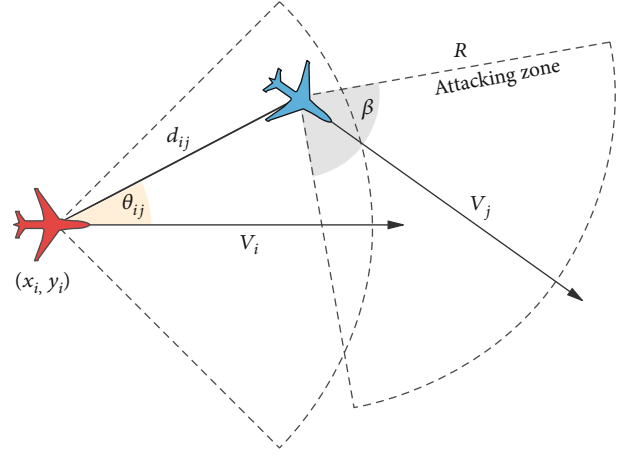


FIGURE 1: Red and blue UAV situation relationship.

UAV to skip planning at the level of velocity or direction. If bad weather is encountered along the way, the UAV can safely change the primitive actions of the flight to avoid a crash while still executing the macro action of chasing the enemy UAV.

The design of macro action needs to follow the following principles: (i) keep a certain distance from allies, and neither fall behind nor collide with each other; (ii) support teammates in time when they are in danger; and (iii) in the process of confrontation, allies cooperate with each other to attack the enemy, so as to maximize damage and reduce their own losses. Based on the above principles, we design the macro action space as $\{\text{Cruise}, \text{Chase}, \text{Escape}, \text{Fire}, \text{Support}\}$. At the beginning of the confrontation, the UAV needs to follow the cluster and *Cruise* the combat area. When an enemy UAV is detected and our UAV is in an advantageous situation, it will execute the *Chase* action until the enemy target enters the attack range, and then execute the *Fire* action to destroy the enemy target. If our UAV is in a disadvantage situation, it will perform an *Escape* action to avoid casualties; then, neighboring allies will perform *Support* actions to help it.

3.4. MADDPG. MADDPG extends the single-agent RL algorithm DDPG to multiagent settings. Each agent contains actor networks and critic networks. The actor network is responsible for the action selection of the agent based on local observations. The critic network is responsible for evaluating the action selected by the actor network. To solve the nonstationarity in the multiagent environment, each agent's critic network needs to access the observations and actions of teammates during training. Through this implicit information exchange, agents are effectively coordinated. In this way, MADDPG is trained with a global state (centralized training), but only local observations are required for execution (decentralized execution).

We parameterize the policy $\pi = \{\pi_1, \dots, \pi_N\}$ of each agent as $\phi = \{\phi_1, \dots, \phi_N\}$. For agent i , the policy gradient of its discounted sum of future rewards is computed by the following equation.

$$\nabla_{\phi_i} J(\pi_i) = E_{s,a \sim D} \left[\nabla_{\phi_i} \pi_i(a_i | o_i) \nabla_{a_i} Q_i^{\pi} s, a_i, \dots, a_N |_{a_i = \pi_i(o_i)} \right], \quad (4)$$

where s is the global state, D is an experience replay buffer, and Q_i^π is the value function modeled by the critic network. Sampling transitions from the replay buffer and the critic network can be updated by the Equation (5). \bar{Q} and $\bar{\pi}$ are the target network.

$$L(\phi_i) = E_{s,a,r,s'} \left[(Q_i^\pi(s, a_1, \dots, a_N) - y)^2 \right], \quad (5)$$

$$\text{where } y = r_i + \gamma \bar{Q}^\pi(s', a'_1, \dots, a'_N) \Big|_{a'_j = \bar{\pi}_j(o_j)}.$$

4. Methods

This section proposes a novel hierarchical MARL framework consisting of a two-level hierarchy. At the top level, the policy searches macro actions in a smaller macro action space and at a lower temporal resolution. At the low level, conditioned on the choices of macro actions, the low-level policy of each agent independently executes primitive actions at every low-level time step. Furthermore, the macro actions are abstracted from primitive actions with prior knowledge, which can effectively integrate the advantages of human expertise and RL agents to facilitate UAV swarm confrontation.

4.1. Hierarchies. The high level of h-MADDPG is composed of a set of discrete macro actions abstracted from human expertise. It collectively divides the primitive action space into several smaller subspaces that can be efficiently explored. The objective of the high-level policy is to maximize the discounted sum of future rewards. We can use a conventional Q-network for the high-level policy whose input is the local observation and output is Q-values for each macro action. Nevertheless, agents select macro actions concurrently, which may lead to inconsistent behavior. For example, some UAVs choose to attack a specific enemy UAV, while some teammates choose to escape, which will result in unnecessary casualties. To better coordinate the choice of macro action, we use MADDPG as the high-level policy to train agents. The actor network and critic network of the high-level policy are updated by the following equations, respectively.

$$\nabla_{\phi_i^h} J(\pi_i^h) = E_{s,\omega \sim D} \left[\nabla_{\phi_i^h} \pi_i^h(\omega_i | o_i) \nabla_{\omega_i} Q_i^{\pi^h}(s, \omega_1, \dots, \omega_N) \Big|_{\omega_j = \pi_j^h(o_j)} \right], \quad (6)$$

$$L(\phi_i^h) = E_D \left[(Q_i^{\pi^h}(s_t, \omega_1, \dots, \omega_N) - y)^2 \right],$$

$$\text{where } y = \sum_t^{t=0} \gamma^t r_{t+t} + \gamma \bar{Q}^{\pi^h}(s_{t+n}, \omega'_1, \dots, \omega'_N) \Big|_{\omega'_j = \bar{\pi}_j^h(o_j)}. \quad (7)$$

In order to be able to output discrete macro actions, the reparameterization trick is used in this paper. As Figure 2 shows, the high level can be regarded as a semi-MDP since macro actions will last for n time steps. Each agent i receives local observation $o_{i,t}$, and the high-level policy selects a macro action $\omega_{i,t} \in \Omega$, the next macro action $\omega_{i,t+n}$ will not

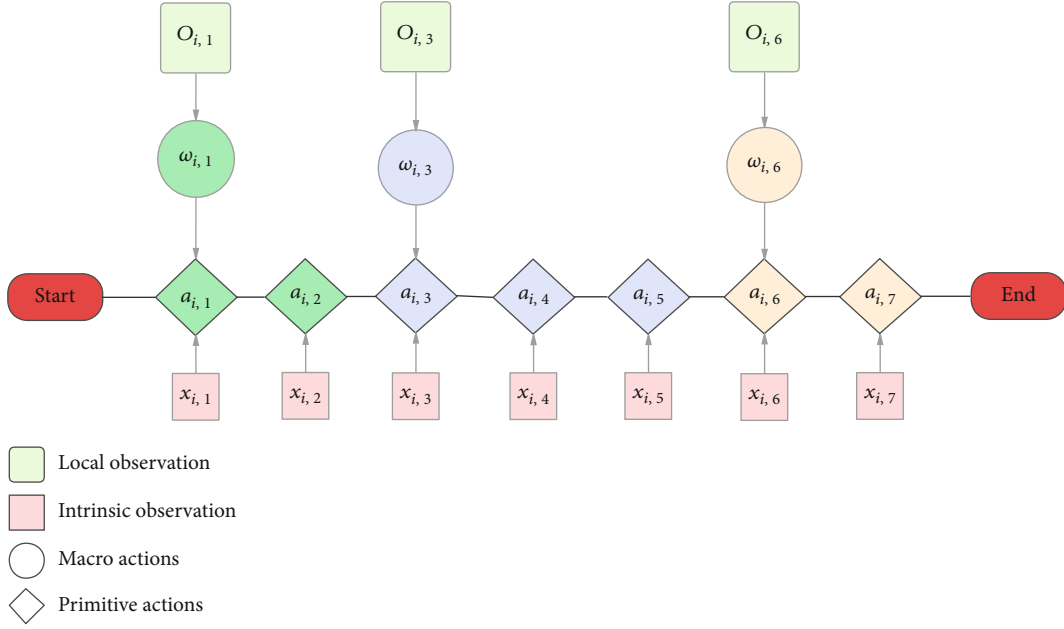
be selected until this macro action is terminated after n steps of primitive actions. The advantages of human expertise and agents can complement each other.

After a macro action is selected, an agent follows it for the next n time steps, during which it can only take actions related to the macro action. Unlike the high-level hierarchy, the low-level policy searches in continuous primitive actions, and it can be modeled as MDPs. Each agent i receives the intrinsic observation $x_{i,t}$ derived from local observation $o_{i,t}$ and then chooses the primitive actions $a_{i,t} \in \mathcal{A}_\omega$ based on the intrinsic observation and the macro action from the high level where $\mathcal{A}_\omega \subseteq \mathcal{A}_i$ is the set of all available primitive actions under the current macro actions. The objective of the low-level policy is to maximize the cumulative intrinsic reward.

The training of the high-level policy is described in Algorithm 1. To facilitate cooperation between agents, h-MADDPG adopts a centralized learning and decentralized execution paradigm during the training of high-level hierarchy. Following this paradigm, the critic of each agent can access the state and action information of all teammates during training so that the environment is stable even if the policies of other agents change. After the training, the agent only uses an independent actor for distributed execution.

4.2. Macro Actions and Pretraining. The key to our approach is to abstract macro actions from primitive actions. The macro action space in our method is abstracted based on human expertise. In a large-scale UAV swarm confrontation, the actions of individual UAVs are selected independently without global control. Therefore, it is necessary to design a set of macro actions for all UAVs so that individual UAVs can choose the optimal decision in the macro action set based on their respective real-time air combat situation. The macro actions of the UAV are defined as follows:

- (i) *Cruise.* If no enemy target is found during the flight, the UAV's behavior at this time is set to fly toward the preset destination. This macro action's intrinsic reward is designed as a negative distance from the preset destination
- (ii) *Chase.* Suppose the UAV detects an enemy target (UAV) during the cruise, but the target is still beyond its attack range. In that case, the UAV needs to estimate the situation of the target enemy aircraft. If the situation assessment result is advantageous, the high-level policy makes a decision close to the target. This macro action's intrinsic reward is designed as a negative distance from the enemy target
- (iii) *Escape.* Suppose the UAV detects an enemy target (UAV) during the cruise, and the result of the situation estimation is inferior, whether or not the enemy target has entered the UAV's attack range. In that case, it should decide to stay away from the target. The intrinsic reward of this macro action is designed to be the distance from the enemy target
- (iv) *Fire.* Suppose the target is within the attacking zone of the UAV, and the result of the situation

FIGURE 2: Hierarchical decision process of agent i .

Input: Pretrained low-level policies $\pi^l = \{\pi_1^l, \dots, \pi_N^l\}$ for all agents

Output: model

- 1: Randomly initialize the high-level networks $\pi^h = \{\pi_1^h, \dots, \pi_N^h\}$ and critic networks $Q^h = \{Q_1^h, \dots, Q_N^h\}$
- 2: **for** each episode **do**
- 3: Get local observation $\mathbf{o}_0 = \{o_{1,0}, \dots, o_{N,0}\}$ and global state s_0
- 4: $t = 0$
- 5: **while** $t < T$ **do**
- 6: Select macro actions $\boldsymbol{\omega}_t = \{\omega_{1,t}, \dots, \omega_{N,t}\}$ for all agents, where $\boldsymbol{\omega}_t = \pi^h(\mathbf{o}_t)$
- 7: **for** $t' = 1, n$ **do**
- 8: Select primitive actions $\mathbf{a}_{t+t'}' = \{a_{1,t+t'}', \dots, a_{N,t+t'}'\}$ conditioned on the macro actions $\mathbf{a}_{t+t'}' = \pi^l(\mathbf{x}_{t+t'}', \boldsymbol{\omega}_t)$, where $\mathbf{x}_{t+t'}' = \{x_{1,t+t'}', \dots, x_{N,t+t'}'\}$ are the intrinsic observations
- 9: Execute primitive actions $\mathbf{a}_{t+t'}'$
- 10: Observe new intrinsic observations $\mathbf{x}_{t+t'}'^{+1}$ and receive extrinsic rewards $\mathbf{r}_{t+t'}' = \{r_{1,t+t'}', \dots, r_{N,t+t'}'\}$
- 11: **end for**
- 12: Get new local observation \mathbf{o}_{t+n} and new global state s_{t+n}
- 13: $\mathbf{R}_t = \{\sum_{j=0}^n r_{1,t+j}, \dots, \sum_{j=0}^n r_{N,t+j}\}$
- 14: Store transition $(s_t, \boldsymbol{\omega}_t, \mathbf{R}_t, \mathbf{o}_t, s_{t+n}, \mathbf{o}_{t+n})$ in D
- 15: Sample a random minibatch of M transitions $(s_i, \boldsymbol{\omega}_i, \mathbf{R}_i, \mathbf{o}_i, s_{i+1}, \mathbf{o}_{i+1})$ from D
- 16: Update the parameters of π^h and Q^h according to Equation (6) and (7)
- 17: $t = t + n$
- 18: **end while**
- 19: **end for**

ALGORITHM 1: The high-level policy training in h-MADDPG.

assessment between the UAV and the target enemy UAV is an advantage. In that case, it should decide to attack the target. When the UAV makes an effective attack action, it receives an intrinsic reward value of +1

- (v) *Support*. When the UAV does not find the target and the ally is at a disadvantage, the UAV needs to be close to the ally to support teammates. This

macro action's intrinsic reward is designed as a negative distance from the ally who needs support.

If both high-level and low-level policies are trained simultaneously, then the low level may not perform macro actions well. Thus, it would be beneficial for the low-level policy to grasp how to perform macro actions. We leverage a bottom-up pretraining procedure to address this problem,

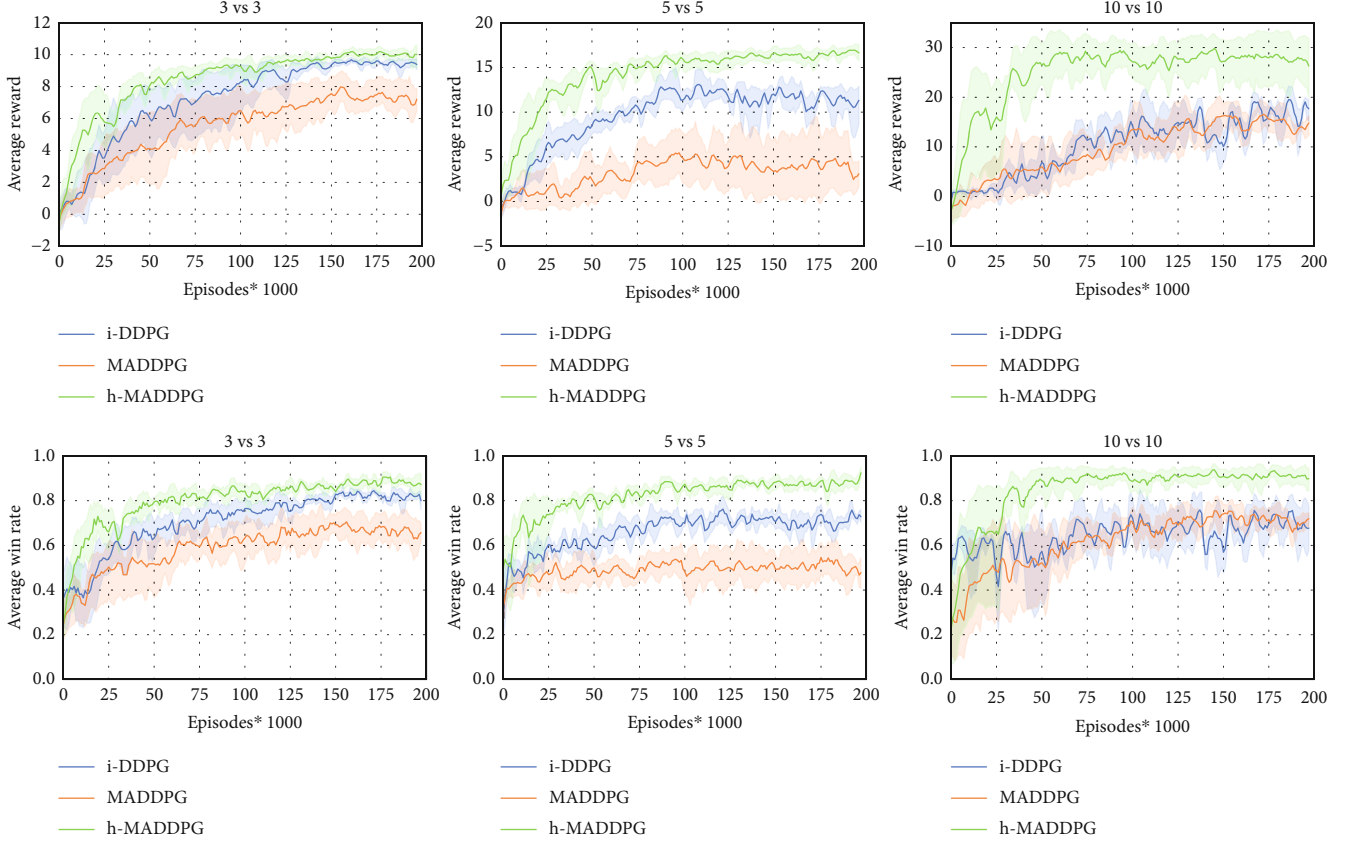


FIGURE 3: Comparison of our method against baseline algorithms in different UAV scales.

in which the low-level policies are pretrained before high-level policies.

4.3. Parameter Sharing. There is a large number of UAVs with identical properties in our scenario. For convenience, we implement parameter sharing among the same macro actions of agents with similar input dimensions, which also improves training efficiency. The training results are very similar to that obtained by policy networks without parameter sharing.

4.4. Action Masking and Death Masking. In the UAV confrontation environment, some actions should not be executed due to the limitation of the scene. For example, a UAV cannot perform an attack action when there is no target in its attack range; it cannot execute chase action when there is no target in its detection range. Therefore, to make the probability of selecting unavailable actions 0, we mask out the unavailable actions during the forward and backward pass.

In centralized training, the critic network needs to input the states of all our agents. However, as the policy network has not been trained, our UAVs may die during the training process. Moreover, the state of the dead agent input to the critic network amplifies the bias of the learned value function, which will hinder the policy learning of other alive agents. Consequently, we make the state of the dead agent as a zero vector, which is supported by the dropout tech-

nique [24]; replacing some states with a certain probability can make the training robust.

4.5. Agent-Specific Global State. In MADDPG, the input to the critic network is the concatenation of all agents' local observations, i.e., (o_1, \dots, o_N) . However, when the number of agents N is too large or the local observation dimension o_i is too high, the input dimension is too high for the critic network to converge. What is more serious is that incorrect value function will further hurt actor network updating.

A practical solution is to use the global state provided by the environment. The global state includes information about UAVs that are out of sight, which reduces a partially observable MDP to an MDP and makes the critic network learning much more effortless. However, it lacks some information contained in the local observation, such as relative position, distance from all other UAVs, and available actions. As described in research [25], the loss of the essential information will result in a bad result. Therefore, the agent-specific global state is used in our method, containing the agent-specific features and other global information.

5. Experiments and Results

We construct a UAV swarm confrontation environment described in Section 3.1. This section shows the effectiveness of our proposed h-MADDPG algorithm, and then we

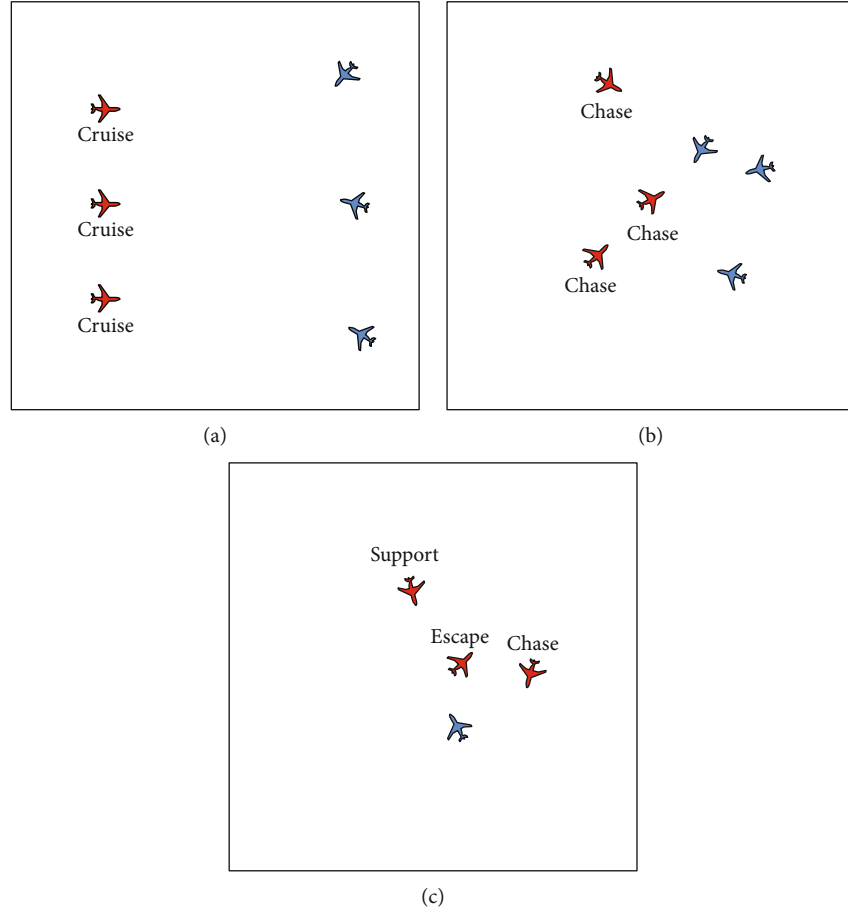


FIGURE 4: Visualization of macro actions selected in one episode in 3 vs. 3. (a) $t = 7$. (b) $t = 21$. (c) $t = 35$.

compare it with the independent DDPG (i-DDPG) and MADDPG.

5.1. Experiment Setup. In this section, we compare h-MADDPG, MADDPG, and i-DDPG in our constructed environment, where the allied UAVs are controlled by agents we trained, and the enemy UAVs are controlled by a heuristic rule to attack the nearest enemy within its attack range. Therefore, a proper strategy is essential to maximize the damage to enemy UAVs while minimizing the damage to itself, which needs a range of collaborative skills.

The agent-specific global state required for MARL training is illustrated in Section 4.5, including each UAV's heading, distance, relative position, and attacking angle to the specific UAV, while the local observation only includes the information of UAVs in the detection range. The UAV destroying an enemy unit will be given a reward of +1, and a reward of -1 will be given when it is destroyed. An extra bonus of +3 will be given to each team member who won the final victory. The UAV in the environment has 6 available primitive actions, i.e., *no-op*, *turn-left*, *turn-right*, *accelerate*, *decelerate*, and *fire*. The high level of h-MADDPG is trained over macro actions, and each macro action is executed for up to 5-time steps. All agents choose their macro actions concurrently. For MADDPG and i-DDPG, the agents are trained based on primitive actions.

For comparison, the actor networks of the three algorithms all consist of 2 hidden layers with 128/128 units, and the critic network has the same structure. The learning rates are 0.0004/0.001 for the actor and critic network. We set the discount factor to 0.95, the batch size to 256, and the replay buffer size to 500000. The neural networks are updated every 100 steps. For h-MADDPG, we share the parameters for high-level and low-level policies, and the low-level policies are pretrained before the high-level policies. Parameter sharing, action masking, death masking, and agent-specific global state are used in all three methods. All experiments are carried out with five different random seeds for evaluation, and results are shown with a 95% confidence interval and moving averaged with a window of 3.

5.2. Evaluations. We evaluate our algorithms in three scenarios with different UAV scales, i.e., 3 vs. 3, 5 vs. 5, and 10 vs. 10. Figure 3 shows the experimental results of different methods in three scenarios. In the 3 vs. 3 scenario, our h-MADDPG can effectively defeat the enemy and achieve a winning rate of 87% while the i-DDPG and MADDPG can reach about 82% and 66%, respectively, indicating that our algorithm results are roughly the same as other algorithms in the 3 vs. 3 scenario. This is because the action space in the 3 vs. 3 scenario is not large enough to take full advantage of our algorithm. However, the winning rates of i-DDPG

and MADDPG in the 5 vs. 5 scenario are 70% and 50%, respectively; and in the 10 vs. 10 scenario, they are 57% and 59%, respectively, indicating that they may fail to learn a good policy, while h-MADDPG has a winning rate of 88% in 5 vs. 5 and 91% in 10 vs. 10 scenarios, respectively. This suggests that our proposed method outperforms the i-DDPG and MADDPG methods. Moreover, as the scale of UAV swarm becomes larger, the winning rate of h-MADDPG becomes higher. We argue that this is due to the fact that when the scale of UAV swarm is larger, the heuristic algorithm of the enemy is no longer applicable. Another interesting result is that the single-agent algorithm DDPG performed slightly better than the multiagent algorithm MADDPG in the experiment, which shows that MADDPG did not learn an effective cooperation strategy from the sparse and delayed reward.

As we expected, the experimental results verify the superiority of the algorithm over the baseline algorithms. In the 3 vs. 3 scenario, both input dimension and action space can be effectively processed by i-DDPG and MADDPG. Consequently, the winning rate of h-MADDPG is close to these two algorithms. When the cluster size increases to 10 vs. 10, the joint primitive action space becomes C^{10} , where C represents the continuous action space of each UAV and 10 is the number of UAVs, while the joint macro action space is 5^{10} , where 5 is the number of the macro actions of each UAV. Since the continuous action space C is much greater than 5, the joint primitive action space is much larger than the joint macro action space. Therefore, h-MADDPG based on macro action can fully explore the action space and make the performance of the algorithm more excellent. In addition, because the upper strategy makes decisions every n steps, it can effectively alleviate the problem of sparse reward in the environment and make the algorithm converge faster. In a word, the advantages of h-MADDPG algorithm mainly owe to small joint action space and sparse decision-making.

Figure 4 shows the decision-making process of our UAVs during the confrontation. We see that all UAVs select *Cruise* in the beginning (Figure 4(a)) to search for the target. Then, when they observe the target, they chase the target and attack (Figure 4(b)). When the situation is unfavorable for our UAV, it executes an escape strategy and attracts enemy UAV into the attack range of its teammate (Figure 4(c)). It indicates that h-MADDPG has learned an effective cooperation strategy.

In order to verify the performance of the algorithms in a more realistic environment, we add obstacles to the environment. The initial state of 3 vs. 3 confrontation scene with obstacles is shown in Figure 5. Two obstacles are randomly placed in the environment, and the UAV too close to them will be destroyed and receive a reward of -1. As can be seen from Figure 6, i-DDPG and MADDPG have both had a lower winning rate in obstacle-free environments than in obstacle environments. Except that in the 10 vs. 10 scenario, the winning rate of h-MADDPG in the environment with obstacles decreases, and the winning rate remains basically unchanged in the 3 vs. 3 and 5 vs. 5 scenarios. We argue that this is because when the cluster size becomes larger, the

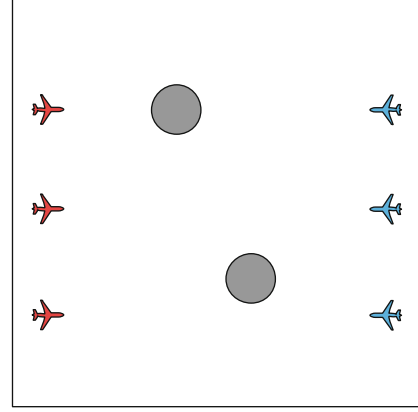


FIGURE 5: The initial state of 3 vs. 3 confrontation scene with obstacles.

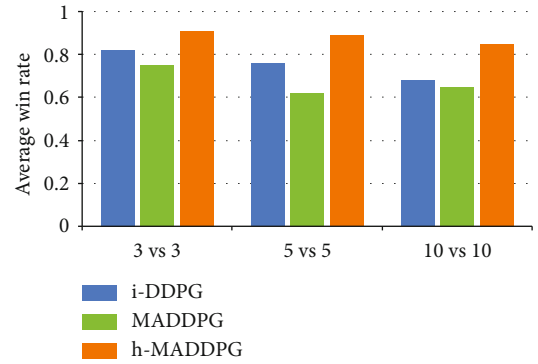


FIGURE 6: Performance comparisons of different cluster sizes with obstacles.

UAVs in the environment are too crowded, which makes some UAVs inevitably collide with obstacles and causes them to crash.

5.3. Ablation Studies. This section investigates the effect of three improvements of the algorithm (action masking, death masking, and agent-specific global state) on policy performance. The following experiments are conducted in an obstacle-free environment.

First, we evaluate the effect of using action masking and death masking in Figure 7. We observe that using an action masking or death masking benefits h-MADDPG's performance. In particular, in the 10 vs. 10 scenario, using an action masking and death masking improves the winning rate of UAV confrontation and speeds up the learning rate.

Then, we compare the agent-specific global state with the original global state representation (the concatenation of local observations). Figure 8 suggests that it is tough for h-MADDPG using the concatenated local observation to converge, particularly in the scenario with many UAVs, which is caused by the much higher state dimension compared with the agent-specific global state. In the 10 vs. 10 scenario, the policy using the concatenation of local observations did not learn an effective cooperation strategy. First, this is because the input dimension of the critic network is too

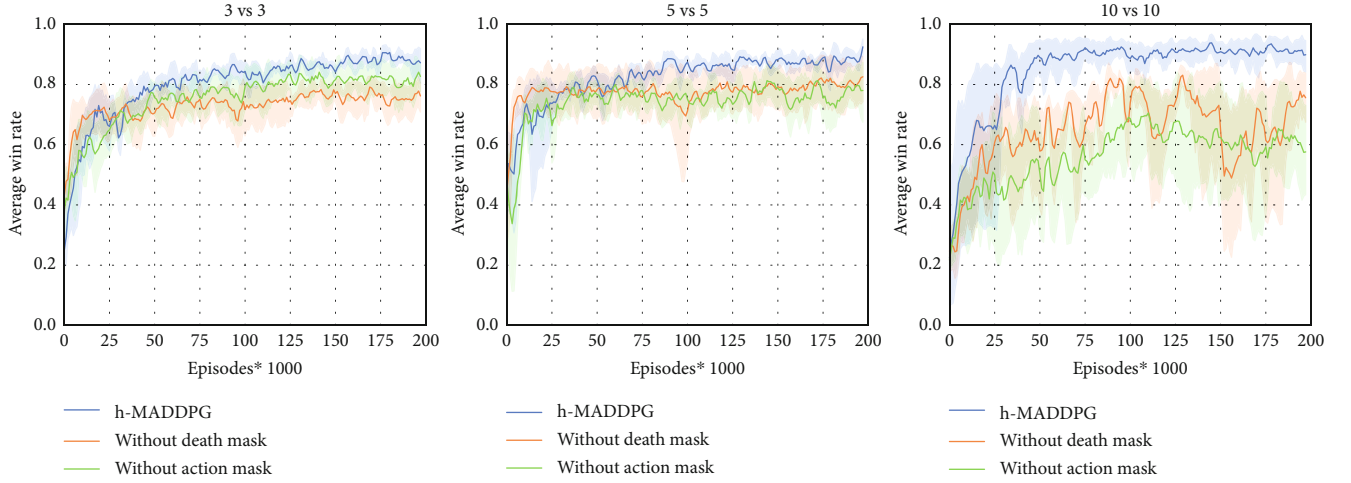


FIGURE 7: Ablation studies demonstrating the effect of action masking and death masking on h-MADDPG's performance.

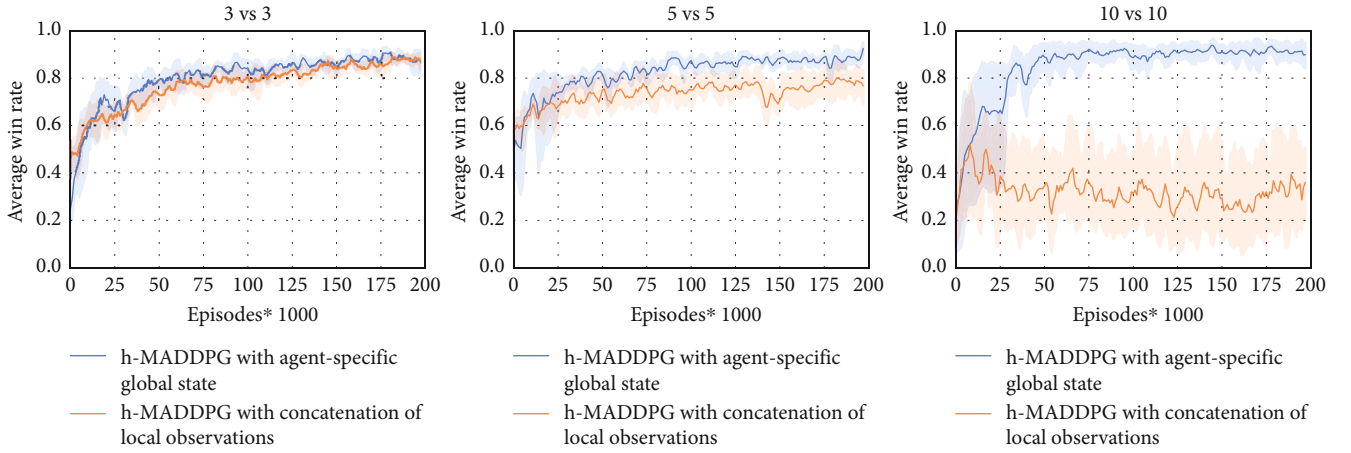


FIGURE 8: Ablation studies demonstrating the effect of global state on h-MADDPG's performance.

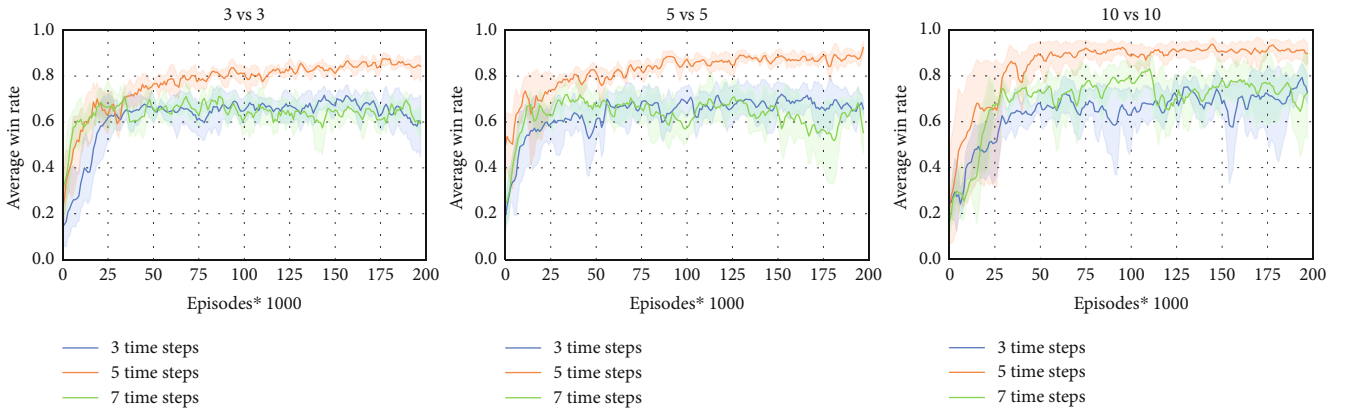


FIGURE 9: Influence of the macro action interval (the number of time steps between two consecutive macro action selections).

high, which makes it difficult for the neural network to converge. Second, the lack of information about UAVs that are out of range results in a nonstationary environment.

These experimental results show that our proposed improvement measures can improve the policy's performance and speed up the learning process.

5.4. Macro Action Interval. In our framework, the high-level policy chooses macro actions to each agent every n time steps, where n is the macro action interval. The macro action interval determines the selection frequency of macro action in the low-level policy and may seriously affect the performance of h-MADDPG. In order to have a clearer understanding of this effect, we test the performance of the algorithm under different macro action intervals in the above three scenarios. From Figure 9, we can see that the macro action interval plays a vital role in performance; when the macro action interval is 5 time steps, the experimental results are the best in all three scenarios.

6. Conclusions

In this paper, we study the UAV swarm confrontation problem with hierarchical multiagent reinforcement learning methods. Compared with the baseline algorithms i-DDPG and MADDPG, the winning rate of our proposed h-MADDPG algorithm is improved by 34% and 32% in the scenario of 10 vs. 10, respectively. In addition, the convergence speed of the model training is also accelerated. This proves that our algorithm can effectively solve the problems of dimensionality curse and sparse reward in large-scale swarm confrontation. Moreover, three efficient training techniques are leveraged in our method, where action masking can prevent UAV from invalid attack and reduce ammunition waste, and death masking can not only reduce the invalid detection of UAV but also reduce the estimation deviation of the value function of the model, and the agent-specific global state ensures the necessary information required for model training while reducing the input dimension of neural network. Empirically, we verify the effectiveness of the proposed algorithm and the training techniques. In future work, we aim to test the performance of h-MADDPG on a more realistic environment with heterogeneous UAVs.

Data Availability

The code developed in this research is openly available at GitHub (<https://github.com/SJTUwbl/multi-UAV.git>).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was funded by the National Science Foundation of China (grant No. 61472476) and Postgraduate Scientific Research Innovation Project of Hunan Province (grant No. CX20210030).

References

- [1] D. Luo, X. Yang, and J. Zhang, "New progresses on uav swarm confrontation," *Science & Technology Review*, vol. 35, no. 7, p. 26, 2017.
- [2] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [3] D. Silver, J. Schrittwieser, K. Simonyan et al., "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.
- [4] T. Lillicrap, J. Hunt, A. Pritzel et al., *Continuous control with deep reinforcement learning*, CoRR, 2015.
- [5] R. Lowe, A. T. YiWu, J. Harb, P. Abbeel, and I. Mordatch, *Multi-agent actor-critic for mixed cooperative-competitive environments*, Neural Information Processing Systems (NIPS), 2017.
- [6] J. N. Foerster and Y. M. Assael, *Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning*, CoRR, 2016, <http://arxiv.org/abs/1605.06676>.
- [7] J. Jiang and L. Zongqing, "Learning attentional communication for multi-agent cooperation," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS'18*, pp. 7265–7275, Red Hook, NY, USA, 2018.
- [8] D. Xing, Z. Zhen, and H. Gong, "Offense-defense confrontation decision making for dynamic uav swarm versus uav swarm," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 233, no. 15, pp. 5689–5702, 2019.
- [9] G. Zhang, Y. Li, X. Xu, and H. Dai, "Efficient training techniques for Multi-Agent reinforcement learning in combat tasks," *IEEE Access*, vol. 7, pp. 109301–109310, 2019.
- [10] A. Tampuu, T. Matiisen, D. Kodelja et al., "Multiagent cooperation and competition with deep reinforcement learning," *PLoS One*, vol. 12, no. 4, pp. 1–15, 2017.
- [11] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: an evaluation platform for general agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, 2013.
- [12] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, Eds., "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '18*, pp. 2085–2087, Richland, SC, USA, 2018.
- [13] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "Qmix: monotonic value function factorisation for deep multi-agent reinforcement learning," *International Conference on Machine Learning*, vol. 80, pp. 4295–4304, 2018.
- [14] H. Jiang, D. Shi, C. Xue, Y. Wang, G. Wang, and Y. Zhang, Eds., "Ghgc: Goal-based hierarchical group communication in multi-agent reinforcement learning," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 3507–3514, Toronto, ON, Canada, October 2020.
- [15] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," *Proceedings of the 36th International Conference on Machine Learning Research, volume 97 of Proceedings of Machine Learning Research*, K.-m. Chaudhuri and R. Salakhutdinov, Eds., , pp. 2961–2970, PMLR, 2019.
- [16] H. Tang, J. Hao, T. Lv, Y. Chen, Z. Zhang, H. Jia, C. Ren, Y. Zheng, C. Fan, and L. Wang, Eds., *Hierarchical Deep Multiagent Reinforcement Learning*, CoRR, 2018, <http://arxiv.org/abs/1809.09332>.

- [17] P. Dayan and G. Hinton, "Feudal reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 5, p. 9, 2000.
- [18] S. Ahilan and P. Dayan, "Feudal multi-agent hierarchies for cooperative reinforcement learning," in *Annual Conference of the American Library Association*, 2019, <http://arxiv.org/abs/1901.08492>.
- [19] Q. Yang, Y. Zhu, J. Zhang, S. Qiao, and J. Liu, Eds., "UAV air combat autonomous maneuver decision based on ddpq algorithm," in *2019 IEEE 15th International Conference on Control and Automation (ICCA)*, pp. 37–42, Edinburgh, UK, July 2019.
- [20] S. Xuan and L. Ke, "Study on attack-defense countermeasure of UAV swarms based on multi-agent reinforcement learning," *Radio Engineering*, vol. 51, pp. 360–366, 2021.
- [21] S. Xuan and L. Ke, "UAV swarm attack-defense confrontation based on multi-agent reinforcement learning," in *Advances in Guidance, Navigation and Control. Lecture Notes in Electrical Engineering*, vol. 644, L. Yan, H. Duan, and X. Yu, Eds., Springer, Singapore, 2020.
- [22] I. Mordatch and P. Abbeel, "Emergence of grounded compositional language in multi-agent populations," 2017, <http://arxiv.org/abs/1703.04908>.
- [23] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proceedings of the Eleventh International Conference, Rutgers University*, pp. 157–163, New Brunswick, NJ, USA, July 1994.
- [24] W. Kim, M. Cho, and Y. Sung, "Message-dropout: an efficient training method for multi-agent deep reinforcement learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 6079–6086, 2019.
- [25] C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of mapo in cooperative, multi-agent games," 2021, <http://arxiv.org/abs/2103.01955>.