The Aeronautical Journal (2022), **126**, pp. 932–951
doi:10.1017/aer.2021.112

REGULAR PAPER

Autonomous and cooperative control of UAV cluster with multi-agent reinforcement learning

D. Xu[1] and G. Chen[2]

[1]State Key Laboratory for Strength and Vibration of Mechanical Structures, Xi'an Jiaotong University, Xi'an, 710049, China and [2]Shaanxi Province Key Laboratory for Service Environment and Control of Advanced Aircraft, Xi'an Jiaotong University, Xi'an, 710049, China
E-mail: aachengang@xjtu.edu.cn

Received: 3 June 2021; **Revised:** 3 September 2021; **Accepted:** 9 November 2021

Keywords: Cluster; Artificial intelligence; Autonomous learning; Cooperative control; Reinforcement learning; Improved multi-agent deep deterministic policy gradient

Abstract

In this paper, we expolore Multi-Agent Reinforcement Learning (MARL) methods for unmanned aerial vehicle (UAV) cluster. Considering that the current UAV cluster is still in the program control stage, the fully autonomous and intelligent cooperative combat has not been realised. In order to realise the autonomous planning of the UAV cluster according to the changing environment and cooperate with each other to complete the combat goal, we propose a new MARL framework. It adopts the policy of centralised training with decentralised execution, and uses Actor-Critic network to select the execution action and then to make the corresponding evaluation. The new algorithm makes three key improvements on the basis of Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm. The first is to improve learning framework; it makes the calculated Q value more accurate. The second is to add collision avoidance setting, which can increase the operational safety factor. And the third is to adjust reward mechanism; it can effectively improve the cluster's cooperative ability. Then the improved MADDPG algorithm is tested by performing two conventional combat missions. The simulation results show that the learning efficiency is obviously improved, and the operational safety factor is further increased compared with the previous algorithm.

Nomenclature

a	actions selected according to the policy at current state s
r	reward
s	current observation states
s'	next states after actions
$Q(s, a)$	Q-value function: Critic for select action a at state s
$R(s, a)$	reward function: reward for choosing action a at state s
$P(s'\|s,a)$	probability function: tendency to select action a at state s to get s'
J	objective function
L	loss function
N	minibatch size
π	reinforcement learning policy
θ	policy parameters
θ^{μ}	Actor function distribution parameters
θ^{Q}	Critic function distribution parameters
μ	Actor function distribution
Q	Critic function distribution
γ	reward decay coefficient
α	learning rate

τ	target update factor
N_t	random process noise

1.0 Introduction

In the United States, Russia and other western developed countries, the use of unmanned aerial vehicles (UAVs) in the battlefield for reconnaissance, interference and strike missions has become the norm. In recent years, with the continuous development of artificial intelligence technology, the combat mode of UAV has gradually developed from 'single combat' to 'cluster intelligence', and has become a key research topic in the military field of various countries [1–5].

UAV cluster combat generally refers to the simultaneous deployment of hundreds of small, fast and high-performance UAVs in the battlefield airspace. UAVs build bionic formations by simulating the clustering behaviour of swarms, schools of fish and ant colonies, carry out interactive communication through various channels, such as combat data link system, tactical radio system and communication relay network, and finally realise operational coordination based on cloud computing, big data and artificial intelligence and other advanced technologies. UAV cluster can carry out strategic deterrence, campaign confrontation and tactical operations in military operations and make the capability of a single UAV be expanded and the overall combat effectiveness of the multi-UAV system can be improved [6–8]. With the continuous integration and development of unmanned systems and intelligent technologies, the actual combat degree of UAV cluster warfare is constantly improving, and it is gradually stepping into the battlefield and opening the curtain of intelligent warfare. In 2015, the US Naval Research Office announced the LOCUST program. Next, DARPA issued the Gremlins program. Unmanned aerial vehicles (UAVs) with electronic payloads and collaboration capabilities are launched outside the defensive area for offshore reconnaissance and electronic attacks. In 2017, the US Perdix micro-UAV high-speed launch demonstration project made a new breakthrough. The successful launch of 103 micro-UAVs from three F/A-18F fighters at Mach 0.6 demonstrated advanced group behaviour and mutual coordination ability. Following, China has again set a world record for swarm flights of fixed-wing UAVs. One hundred nineteen fixed-wing drones successfully demonstrated the actions such as dense catapult take-off, aerial assembly, multi-target grouping, formation and encirclement. But these research achievements just stay at the cluster operation concept demonstration stage and fail to realise the autonomous decision and intelligent control. Considering that the future battlefield will have higher and higher requirements for real-time and intelligence, it is very important to truly realise autonomous and intelligent cooperative operation of UAV cluster by studying the application of intelligent control algorithm in different cluster combat tasks, for seizing the initiative of future air combat [9–17].

UAV cluster as a Multi-Agent System (MAS) [18–21], the environment model and dynamic model are both complex. Each agent in the cluster needs to consider the influence from the actions of other agents, and the dynamics of the environment when learning behaviour strategies. However, traditional Reinforcement Learning (RL) [22–30] algorithms are mainly for a single agent, so they are not applicable for cluster control. For Deep Q-Network (DQN) [26,31] algorithm based on Q-learning [32,33], a single agent in MAS will be affected by the state of other agents, resulting in different state transitions and making the method of experience replay no longer applicable. For Policy Gradient (PG) [29,34] method, the continuous changes in the environment and the increase in the number of agents will lead the learning variance to increase further. Therefore, we propose to use Multi-Agent Reinforcement Learning (MARL) [35–39] algorithm to realise the cooperative control of UAV cluster.

The research on the intelligent collaborative control of UAV cluster is mainly to construct efficient learning framework and rigorous reward mechanism. Considering the instability of the environment in MAS and the interaction among agents, and refering to the Lowe's research [40], we made some key improvements on the basis of the weaknesses of the original Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm. The improved new learning framework and more strict reward mechanism both make the algorithm be able to obtain more accurate evaluation values, making the learned behaviour strategy more optimised. Successfully realise the autonomous cooperative control of UAV cluster.

The paper is organised as follows. Section 2 introduces the background and development of the Reinforcement Learning algorithms. Section 3 explains in detail the improved MADDPG algorithm and the reasons for improvement. Section 4 shows the experimental setup of two conventional combat missions for UAV cluster. Section 5 shows the simulation results of the improved MADDPG algorithm on two conventional combat missions, and the comparison with the MADDPG before improved. The conclusions and future work are presented in Section 6.

2.0 Background

RL is an online learning technique that is different from supervised and unsupervised learning methods. It regards learning as a process of attempt and evaluation. First, the reinforcement learning system perceives the state of the environment and then takes a certain action and act on the environment. After the environment accepts the action, the environment state changes and gives a reinforcement signal (reward or punishment) at the same time as the feedback to the reinforcement learning system. The reinforcement learning system selects the next action based on the reinforcement signal and current state of the environment. The principle of the selection is to increase the probability of being rewarded. The selected action not only affects the state of the environment at the next moment, but also the final reinforcement value. The agent in reinforcement learning system can continuously accumulate experience just by many times' attempts, and finally get the best behaviour strategy [5,26–30]. Traditional RL algorithms are mainly for the single agent system (SAS), so later we also call them Single-Agent Reinforcement Learning (SARL) algorithm. Traditional RL algorithms can be divided into value-based reinforcement learning, the reinforcement learning based on policy gradients and Actor-Critic network.

As the most basic value-based reinforcement learning algorithm, Q-Learning algorithm [31,33] is used to guide the agent's actions by calculating the Q-value of each state and action pair and storing it in Q-table. But this algorithm is only applicable to the environment where the state space and the action space are both small and discrete. In actual control model, state space and action space are large. If the Q-table is used to record the state and action pair of each computation, the table will be very large. It will cost much time when inquiring about the maximum Q-value. In order to solve the problem that the state and action space in real control model are too large to lookup in Q-table, it is proposed to use Deep Neural Network (DNN) to predict the Q-value and learn the optimal strategy. The reason is that DNN has a good effect on the extraction of complex features. Therefore, Deep Q-Network (DQN) [26] combining deep neural network and Q-Learning algorithm was proposed. It is a model-free deep reinforcement learning algorithm and has three key technical improvements: The first one is to approximate the value function using deep convolutional neural network; the second one is using the target network to update the target Q-value to ensure parameter convergence; and the third one is to use an experience replay buffer to store the labeled data samples, which can break the correlation of random sampling in neural network and improve the update efficiency. Although the DQN algorithm has been able to solve the problem of high-dimensional state or action space, this value-based reinforcement learning algorithm is only applicable to discrete space. It has insufficient processing capacity for high-dimensional continuous space and cannot solve the problem of random strategy. Therefore, the reinforcement learning algorithm based on policy gradient was proposed as another kind of RL algorithm.

Policy Gradient (PG) [29] algorithm omits the intermediate steps and directly selects actions based on the current state. It can enhance and weaken the possibility of choosing actions by computing reward, so there is no need to compute Q-values. For good behaviours, it will increase the probability being selected next time, and for the bad ones, the probability being selected next time will be weakened. After the random strategy is obtained through PG learning, the optimal strategy probability distribution needs to be sampled at each step of the behaviour selection to obtain the specific value of the action. The action space is usually high-dimensional, so there is no doubt that frequent sampling is a waste of computing power. Besides, the strategy usually converges to a local optimal rather than a global optimal. For the action space, it may be continuous values or very high-dimensional discrete values, so that the spatial dimension of the action is extremely large. If we use a stochastic strategy, we need to study the probabilities of all possible actions, and the number of samples required is very large. So the deterministic strategy is came up with to simplify the problem. It makes the determined value of the action at each step can be obtained directly by calculating the selected action. It is Deterministic Policy Gradient (DPG) [30] algorithm.

Combining the value-based reinforcement learning method and the reinforcement learning method based on policy gradient, a new algorithm called Actor-Critic (AC) [41–44] network is generated. It can not only select the appropriate action in the continuous action space, but also evaluate the selected action by calculating the reward. For AC algorithm, the model involves two neural networks: Actor network and Critic network. Actor network has the same function with PG algorithm, which is responsible for generating actions and interacting with the environment. It selects the appropriate actions in the continuous action space with policy functions and round update, so the learning efficiency is relatively slow. Critic network apply a value-based algorithm to achieve single-step update, such as Q-Learning. The value function in algorithm is responsible for evaluating the selected actions and directing the next actions. Each time a parameters update is performed in a continuous space, and there is a correlation between the parameters before and after updating. That is to say, the training data of the model is no longer an independent and identical distribution, which leads the neural network to take a one-sided view of the problem.

In order to solve the above problems, an improved reinforcement learning algorithm called Deep Deterministic Policy Gradient (DDPG) [45–48] was proposed. It not only draws on the key technologies of the experience replay mechanism and target network from DQN, but also solves the problem of difficult convergence. DDPG adopts Convolutional Neural Network (CNN) as the simulation of policy function μ and Q^{μ}. The paramater μ comes from the policy network and Q^{μ} is the parameter of Q network. The roles of these two networks are similar to Actor and Critic networks, respectively. We update Q-value by minimising the loss function:

$$L = \frac{1}{N}\sum\nolimits_i \left[r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'}) - Q(s_i, a_i|\theta^{Q})\right]^2 \tag{1}$$

And use the sampled gradient to update the policy:

$$\nabla_{\theta^{\mu}\mu|s_i} \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^{Q})|_{s=s_i,a=\mu(s_i)} \nabla_{\theta^{\mu}}\mu(s|\theta^{\mu})|s_i \tag{2}$$

The target networks adopted in DDPG are updated by $\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}$ and $\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}$. One advantage of DDPG algorithm is that it can use only low-dimensional observations to learn competitive policies. And another advantage is that it can do off-policy exploration. For continuous action space, DDPG algorithm converges faster than DQN algorithm.

In recent years, with the introduction of the concept of cluster, the emphasis of reinforcement learning has gradually shifted to MARL. MARL means that multiple agents in cluster interact with the environment through ‘trial and error’, and each agent has an impact on the environment. The agents in cluster have complex relationships, and their interests may be aligned, not entirely aligned, or completely opposite. And the cluster can be cooperative, competitive, or both. In MAS, each agent needs to learn its own optimal strategy to maximise its utility. The environment of SAS is stable, but in MAS, each agent constantly learns and improves its strategy by interacting with the environment and calculating the reward value. Therefore, from the perspective of each agent, the environment is complex and dynamically unstable, which no longer meets the convergence conditions of traditional reinforcement learning algorithms. It is precisely because of the instability of MAS environment that the behaviour strategies of each agent will not be completely consistent. In MAS, the behaviour of any agent will affect the system environment and the behaviour choices of other agents. When one agent’s strategy changes, other agent's optimal strategy may also change, which will affect the convergence of the algorithm and make the design of the reward function more complicated. For DQN algorithm, the agent's own state transfer will be different under the influence of other agents' states, resulting in the experience replay method is no longer applicable. For PG algorithm, the constant change of environment and the increase of agent number will lead to the further increase of learning variance. At the same time, the dimensions of the connected actions combined by the current actions of each agent will also increase exponentially with the increase of the number of agents. As a result, strategy learning becomes complex and time consuming. Comprehensively considering the above difficulties, after getting familiar with the SARL algorithm, especially the implementation principle of DDPG algorithm, Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm [40] is proposed. It is a MARL algorithm combining AC network and DQN algorithm, and has the following three characteristics: (1) The optimal strategy obtained after learning can output the optimal actions using only local information; (2) No need to obtain the dynamic model of the environment in advance; (3) Not only suitable for cooperative environment, but also suitable for competitive environment.

3.0 Methods

Although he MADDPG algorithm adopted in Ref. [40] has been able to solve the problem of collaboration among multiple agents, there are still serious problems such as low learning efficiency, long time consuming and many internal collisions when testing in the simulation scene set in Section 4 of this paper. Therefore, we made key improvements to the learning network framework and the setting of reward function on the original algorithm in view of these deficiencies, and demonstrated the superiority of the improved algorithm through simulation comparison.

Improved MADDPG algorithm is similar to the original MADDPG algorithm in principle. It's essentially an Actor-Critic network, but makes a series of improvements to the Actor-Critic algorithm. These improvements makes a new algorithm more suitable for complex multi-agent scenarios. First, the improved MADDPG algorithm adopts the strategy of centralised training with decentralised execution. During training, the Critic network and Actor network are trained using centralised learning. Actor network selects an action to execute according to the current state. The Q-value of the current state-action pair is calculated by the Critic network and fed back to the Actor network. Critic network trains based on the estimated Q-value and the actual Q-value, and Actor network updates its strategy based on the feedback. In the test phase, you only need the Actor network to make actions and no longer need the feedback from the Critic network. Therefore, during training, we can add some additional information to the Critic network to get a more accurate Q-value, such as the states and actions of other agents. That is the meaning of concentrated training. Agent evaluates the current action not only according to its own situation, but also the information of other agents. The decentralised execution means that after each agent is fully trained, the agent can quickly complete the selection of the optimal action only according to its own state and no longer needs the states and actions of other agents. It can greatly reduce the amount of data required for calculation. Second, the improved MADDPG algorithm also adjusts the observation data recorded in the experience replay buffer. We change the state and action of a single agent to the joint state and action of all agents. And we also use the strategy set effect optimisation method to make each agent learn their own different strategies, and finally use the overall effect of all strategies to determine the optimal strategy. The main purpose of this improvement is to improve the stability and robustness of MADDPG algorithm. Actually, the improved MADDPG algorithm is essentially also a DPG algorithm. For each agent, a Critic network requiring global information and an Actor network needing local information can be obtianed through training, and each agent has its own reward function. So, the new improved MADDPG algorithm can be used not only for cooperative tasks but also for adversarial tasks, and the action space can also be continuous.

The structure of the improved MADDPG algorithm mainly includes Q network and P network, as shown in Fig. 1, and these two networks are trained at the same time. The function of Q network is

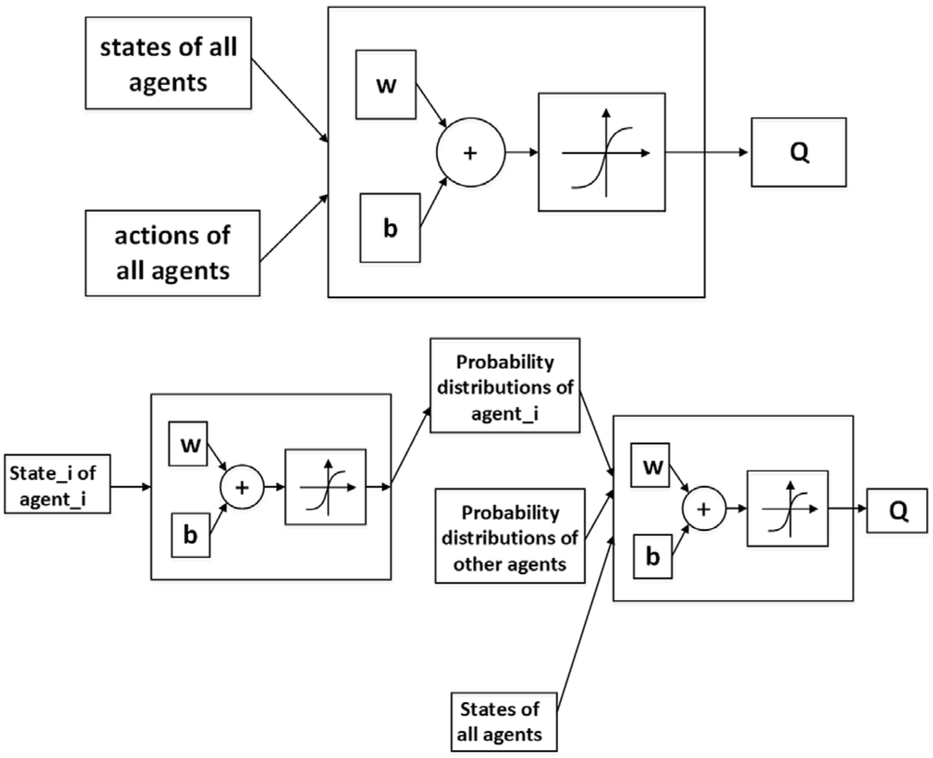

Figure 1. *MADDPG algorithm structure. (a) Q network, and (b) P network.*

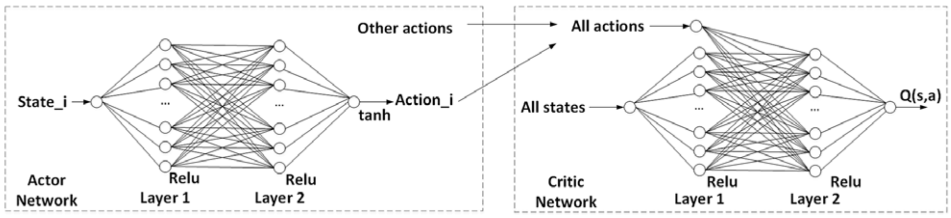

Figure 2. *Improved P network framework.*

consistent with the function of DQN algorithm. The difference is that the input data is expanded to the states and actions of all agents in the MAS environment. Therefore, it can be considered that Q network is the overall evaluation of cluster behaviour. However, P network is to conduct behaviour selection and optimisation learning for a single agent in the cluster. The first half of P network computes a probability distribution of all possible actions for the agent's current state, which is equivalent to Actor network. The latter part works like a Critic network. Since the change of the behaviour of a single agent in the cluster will lead to the instability of the whole system, the states and actions of other agents must be taken into account when evaluating the behaviour of this agent. The final training results are the output of the optimal actions. For the improved algorithm, the innovation of the network structure is that the structure of P network is different from that of traditional Actor-Critic network. It treats the Actor network and Critic network as a whole for training and determines the optimal action by maximising the Q-value.

The specific structure of the improved P network is shown in Fig. 2. By comprehensively considering the state variables and action variables of all agents, the calculated Q-value is more accurate and the behaviour selection is more optimised.

The specific steps of the improved MADDPG algorithm can refer to the followings:

```
Improved MADDPG algorithm for $N$ agents
for episode = 1: M
(a) Initialise a random process $N$ for action exploration
(b) Receive initial observation state s
(c) for t = 1: T
    i. For each agent $i$, select action $a_i = \mu_{\theta_i}(o_i) + N_t$ according to the current policy and exploration
       noise
    ii. Execute actions $a = (a_1, \cdots, a_N)$ and observe reward r and observe new state s′
    iii. Store transition $(s, a, r, s')$ in R
    iv. $s \leftarrow s'$
    v. for agent $i$=1:$N$
        (a) Sample a random minibatch of S transitions $\left(s^j, a^j, r^j, s'^j\right)$ from R
        (b) Set $y^j = r_i^j + \gamma Q_i^{\mu'}(s'^j, a'_1, \cdots, a'_N)|_{a'_k=\mu'_k(o_k^j)}$
        (c) Update Critic by minimising the loss:
            $L(\theta_i) = \frac{1}{S}\sum_j \left(y^j - Q_i^{\mu}(s^j, a_1^j, \cdots, a_N^j)\right)^2$
        (d) Update the Actor policy using the sampled gradient:
            $\nabla_{\theta_i} J \approx \frac{1}{S}\sum_j \nabla_{\theta_i}\mu_i(o_i^j)\nabla_{a_i} Q_i^{\mu}(s^j, a_1^j, \cdots, a_i, \cdots, a_N^j)|_{a_i=\mu_i(o_i^j)}$
            end for
    vi. Update the target network parrmeters for each agent $i$:
        $\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$
        end for
```

4.0 Mission Scenarios

The improved MADDPG algorithm is used to construct a multi-agent reinforcement learning model for two conventional combat missions: navigation and location regional reconnaissance and round-up confrontation. After the task instruction is determined, the cluster can use the corresponding reinforcement learning model to learn autonomously and complete the selection of the optimal strategy, so as to achieve multiple UAVs' task distribution and formation cooperative control, and finally complete the target task quickly and efficiently.

4.1 Regional reconnaissance

Regional reconnaissance can be divided into two parts: The first part is to complete location determination, and the second is to navigate to the corresponding coordinate points. Location determination requires the UAV cluster to independently complete the location distribution according to the number of cluster and the area to be detected. At the same time, it requires to ensure that the cluster can cover the largest area without missing any information. According to the detection perspective of each UAV, the reinforcement learning algorithm is used to perform autonomous calculations to achieve cluster allocation and formation control [49,50]. Because UAVs with the same detection angle have different detection areas at different heights, the higher the height, the larger the area, the more information, but the details are not clear enough. In order to ensure the full coverage of the area when the number of clusters is determined, the algorithm should be able to quickly calculate according to the area to be investigated, and independently complete the optimal allocation of each UAV in the cluster. The detailed implementation method can refer to the Refs [51] and [52]. The second part is to make the UAV cluster to formulate optimal navigation schemes based on recent principles by observing the relative position of each UAV.

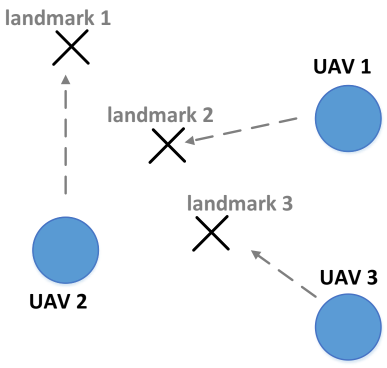

Figure 3. *Cooperative navigation.*

The goal is to make the corresponding UAVs be able to reach the designated location as soon as possible without internal collisions. The specific implementation plan can refer to Fig. 3.

In this paper, we mainly study how the UAVs in cluster can complete mission assignment and reach the corresponding positions through cooperative communication without internal collisions. We call it cooperative navigation in later study.

In the environment of cooperative navigation, N UAVs must cooperate through physical actions to reach a set of N landmarks. Each UAV makes the task arrangement by observing the relative positions of other UAVs and all the landmarks and considering all UAVs' distances from arbitrary landmarks. In other words, the UAV cluster has to cover all of the landmarks using the least time. Further, these UAVs will be penalised when colliding with each other. The goal of cooperative navigation is to enable the cluster to assign targets autonomously and enable UAVs to reach the corresponding location as quickly as possible without colliding with each other.

4.2 Round-up confrontation

When carrying out tracking, encircling, and enemy-to-self confrontation, we must first complete the assignment of mulitple targets. That is assigning different targets to the corresponding UAVs to simultaneously complete multiple targets' tracking and striking. Typically, it is the least expensive arrangement to allocate three UAVs round up a target. The specific scheme design can refer to Fig. 4. It shows the combat plan that multiple UAVs cooperate to round up and attact one target. And the attaction mode adopts the suicidal destruction combat mode. It means that arranging any one UAV to hit the target. It requires that several UAVs with the same combat target and combat mission can communicate with each other and cooperate to complete the target mission while avoiding internal collisions. This is a problem of cooperative control in the MAS, so we plan to use MARL algorithms to complete the autonomous perception and learning of multiple agents in order to obtain the optimal cooperation strategy.

In the environment of round-up confrontation, N slower cooperative UAVs must chase a faster target UAV around a randomly generated environment with L large obstacles impeding the way. One UAV chooses actions by observing the relative positions and velocities of the other UAVs, and the positions of the obstacles. Each time any one UAV in the cluster collides with the target UAV, the cluster will be rewarded while the target UAV is penalised. If there is one collision in cluster, the cluster will receive one punishment. The more collisions, the more punishments. In addition, when the cluster don't capture the target, the reward function of cluster is related to the relative distance between the target UAV and each UAV in cluster. The reward value will decrease as the distance from the target increases. To the contrary, the target's reward will increase with the distance increasing.

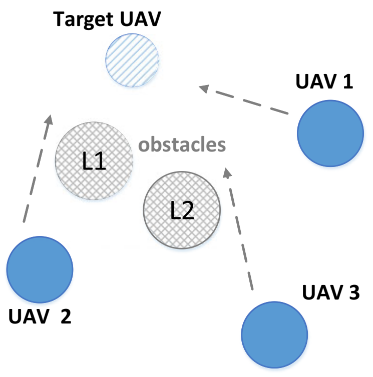

Figure 4. *Cooperative round-up confrontation.*

5.0 Simulation Results

5.1 Simulation tests on cooperative navigation

We compiled the environmental document of cooperative navigation mission and constructed the learning framework of the improved MADDPG algorithm. In the environmental document, we assigned three UAVs to reach three designated locations in the shortest possible time and made sure these UAVs didn't collide with each other. When we set up the algorithm network, we designed Q network and P network separately. Wherein, the structure of Q network and P network are similar to the structure of Actor-Critic network. It uses a fully connected neural network with two hidden layers, shown in Fig. 2. The number of nodes of each layer is 128. In both hidden layers, the activation functions are rectified linear units (Relu) functions. In order to make the choice of action more optimised and the evaluation of action more accurate, the tangent hyperbolic (tanh) function is applied in the output layer of the Actor network, and the action variables output by the Actor network are introduced in the input layer of the Critic network. These changes are the improvements for the original algorithm, which can increase the learning efficiency and optimise the learned policy.

Besides, we also improved the reward mechanisms because policy learning is based on reward values. The states available to the algorithm are the current position of each agent and the location of each landmark. To compile simply, we test the improved MADDPG algorithm in two-dimensional environment, and make the agents do uniform motion. We initialise the initial coordinate position of each agent and the specified target positions, respectively, making coordinates x and y to be any value in the range of -1 to 1, respectively. The reward value for each agent is initialised to 0 before the iteration training begins. During training, the UAVs in the cluster select actions randomly and compute the rewards according to the reward mechanisms, then evaluate the current actions to optimise the action selection. After several iterations, the optimal policy can be obtained. At this point, UAV cluster can autonomously select actions according to the optimal policy to carry out the cooperative navigation task as soon as possible.

The reward function of each UAV is calculated with its actual position and the designated locations. Because the UAV cluster is fully cooperative relationship in cooperative navigation mission, we must consider the whole cluster when calculating the reward value. That is to consider all the distances from each UAV to each landmark. Since the algorithm must first ensure that each landmark must have one UAV to reach, we hope that the UAVs can complete collaborative allocation and adjust the scheme at any time as the situation changes, so that the nearest UAV to the landmark to cover. Therefore, when the reward function is set, the reward value should increase as the relative distance between the UAV and the landmark decreases. To achieve collaborative allocation and avoid multiple UAVs covering the same landmark, we need to add collision avoidance settings. When two UAVs have the same distance from the same landmark and are both closest, they can coordinate one of them to the second nearest landmark according to the principle of avoiding collision, eventually realising the full coverage of all landmarks. Since it is a problem of full cooperation, we need to calculate the total reward value of the whole cluster

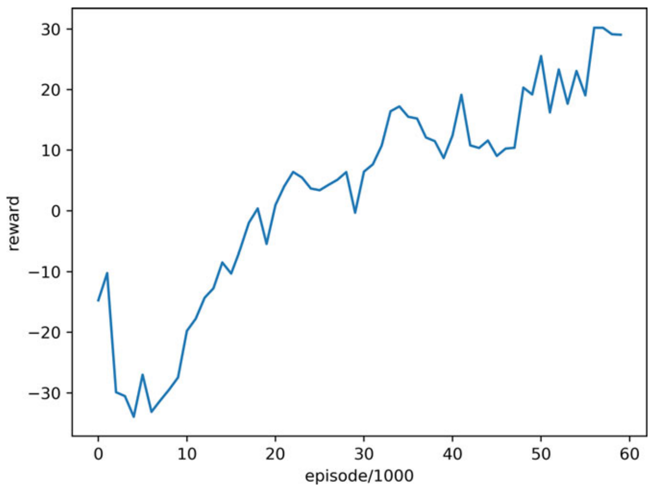

Figure 5. *The total reward per episode for the cluster.*

under the current actions and learn the policy accordingly. For one agent (the agent in the text is UAV), we calculate the distances of all landmarks to it and take the minimum distance to compute the reward. In order to occupy the designated locations, the agent's reward has a negative correlation with the minimum distance. The shorter the distance, the closer to the landmark, the larger the reward. Considering the size of agents and landmarks, we think the agent occupys the landmark when the distance is less than the sum of their radii (the sum is 0.1). As a reward, we make the reward value add 10. So the reward function of each agent is made as the following equation.

$$reward = \begin{cases} reward - 0.1 \times \min(dists), & if \ \min(dists) \geq 0.1 \\ reward + 10, & else \end{cases} \tag{3}$$

$$dists = \left| s_{agent} - s_{landmark_i} \right|_{i=1,\cdots,n} \tag{4}$$

Wherein, s_{agent} refers to the position of an agent waiting to be calculated the reward, and $s_{landmark}$ is the position of any one designated locations. In order to avoid internal collision in the process and multiple agents covering the same landmark, we make the following settings. If one agent collides with another one, the agent should be punished, and the reward should be subtracted 10 for one collision. The total reward value per episode of the cluster are shown in Fig. 5.

In Fig. 5, the total reward generally increases with episodes. During the first 10,000 episodes, it is a tentative learning phase, the cluster is exploring the rules through trial and error. At the same time, we also add disturbances to increase the uncertainty of the environment. So the reward value is decreasing fast. But in the next episodes, the reward continues to increase rapidly. It indicates that the cluster learns and trains through the improved MADDPG algorithm and the learning efficiency is high. According to the reward mechanism we set, if the UAV cluster could cover all landmarks, the reward value should be close to 30. From Fig. 5, it's clear that the total reward value of cluster can reach 30 near to 60,000 episodes, which indicates that the three UAVs can cover three different landmarks without internal collisions.

To better observe the simulation results, we output a frame of image every 30,000 time steps to display the training process in an animated manner. Figure 6 shows the training results under two scenarios chosen from 60,000 scenarios. Figure 6(a) is the common case where the assigned landmarks are dispersed. In this case, the cooperative navigation mission is completed when the UAV cluster can cover or approach all the designated landmarks separately. Figure 6(b) shows a special case where two designated landmarks are very close. Under this circumstance, the policy learned through the improved MADDPG

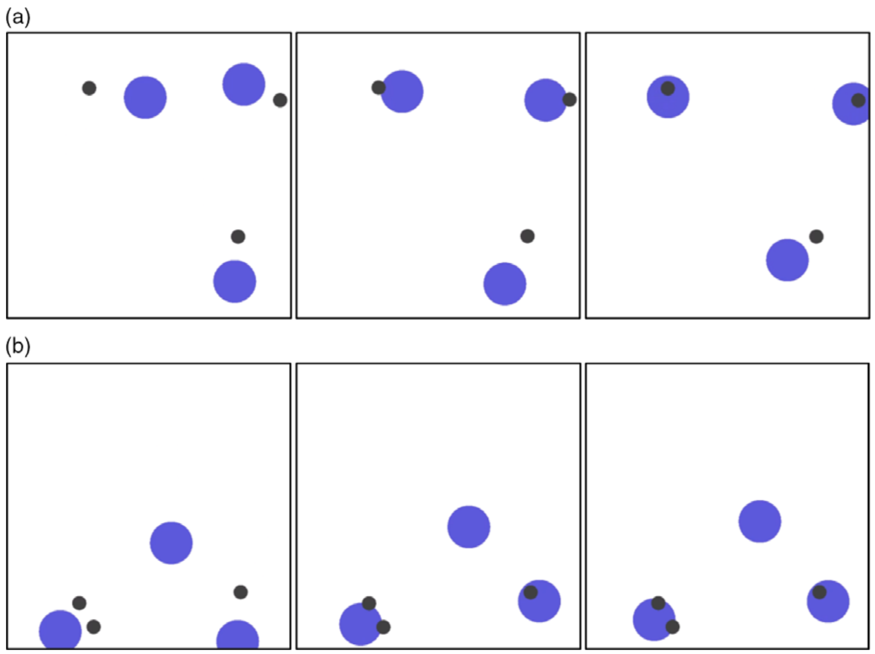

Figure 6. *The training process for cooperative navigation. (a) The common scenario where the assigned landmarks are dispersed, and (b) the special scenario where two designated landmarks are very close.*

algorithm will automatically arrange a nearest UAV to cover the two landmarks at the same time. It can avoid damage caused by collision between UAVs.

5.2 Simulation tests on round-up confrontation

In the environmental document of round-up confrontation, we depart the agents into good agents and adversary agents, and command adversaries to chase and attack good agents. The states available to the improved MADDPG algorithm are the number of agents, including good agents and adversaries, and the current position of each agent. If there are obstacles, the states available still include the number of obstacles and their locations. In order to compile simply, we test the improved MADDPG algorithm in two-dimensional environment and set three adversaries to round up and attack one good agent while avoiding two obstacles; the velocity of all agents are uniform. We first initialise the x and y coordinates of all agents both to the range $(-1, 1)$, the locations (x and y coordinates) of designated obstacles to the same range $(-1, 1)$ and the initial reward values to 0. Then we select actions randomly for adversaries and good agent separately, and compute their reward values according to corresponding reward functions. Finally we learn the optimal policy autonomously on the base of cumulative reward. The goal is to choose actions according to the optimal policy to make UAV cluster be able to round up and attack the free moving target.

For the good agent, the goal is to escape the adversaries' pursuit. So the reward has a positive correlation with the distances between the good agent and all the adversaries. The longer the distances, the larger the reward. The reward function for good agents is shown in Equation (5).

$$reward = reward + 0.1 \times \sqrt{\left(s_{agent} - s_{adversary_i}\right)^2}\,\Big|_{i=1,\cdots,n} \tag{5}$$

Wherein, s_{agent} refers to the position of a good agent waiting to be calculated the reward, and $s_{adversary}$ is the position of any one adversary. When the good agent collides with any one adversary, the reward of good agent should be subtracted 10 as one punishment. While, for adversaries, the reward has a negative correlation with the distance from the good agent. The shorter the distance, the larger the reward. The reward function for adversaries is shown in Equation (6).

$$reward = reward - 0.1 \times \sqrt{\left(s_{agent} - s_{good\_agent_i}\right)^2}\,\Big|_{i=1,\cdots,n} \tag{6}$$

Wherein, s_{agent} refers to the position of an adversary waiting for calculating the reward, and s_{good_agent} is the position of any one good agent. Because a good roundup requires the cooperation of three UAVs to approach the target from all directions, we need to calculate the reward value of each UAV separately and hope that each UAV is constantly approaching the target. When an UAV collides with the target, the reward value of this UAV should be added 10 as a reward. In order to avoid internal collisions among the cluster, the reward value will be subtracted 10 as a punishment when this UAV collides with another one. The simulation results are shown in Fig. 7.

In Fig. 7, the rewards of three adversaries are on the rising trend. Although the reward values of No. 1 adversary and No. 3 adversary fall after 50,000 episodes, the No. 2 adversary keeps rising. While the reward value of good agent continues to decrease before 50,000 episodes and has dropped below 0 after 5,000 episodes. Considering that the number of good agent is only one, there won't be punishments for internal collisions. According to the reward function of the good agent, we can conclude that the reward of good agent always increases except when it collides with the adversaries. Therefore, the good agent can be chased and attacked by the adversaries after 5,000 episodes, and the adversaries can keep closer and closer distances when rounding up the good agent. The reward value of good agent has dropped below -10 after about 35,000 episodes and drops faster than before, which shows that the adversaries after training can effectively prevent the good agent from escaping and the success rate of roundup and confrontation is also increasing. Reference to the reward function of adversaries, we know that the reward always decreases except when the adversary collides with the good agent. Beacuse the good agent can be attacked by the adversaries after 5,000 episodes, the maximum reward of adversaries should be greater than 0. Based on this, Fig. 7(a) shows that there is a collision in adversaries. In order to compare the reward function curves of adversaries and good agent more clearly, we put them in the same picture, shown in Fig. 7(c). The later the training, the greater the difference between the reward values of adversaries and good agent, the better the cluster policy of adversaries.

In our research, we found that the learning efficiency of MARL algorithm will be different when the reward function changes. From the training results in Fig. 7, we can see that the reward values of adversaries is generally small, which means that there are internal collisions. So we made some changes in the reward function setting of adversaries. We modify Equation (6) and get the new equation.

$$reward = reward + 1.0/\sqrt{\left(s_{agent} - s_{good\_agent_i}\right)^2}\,\Big|_{i=1,\cdots,n} \tag{7}$$

The reward value is increased, and the shorter the distance, the greater the amount of increase. In order to avoid the problem that the reward value will increase sharply even be infinite, we set the calculation conditions for Equation (8). When there is a collision between this adversary and the good agent, the reward value is directly added 10 as a reward. That is, when the distance is small enough, the equation no longer applies. The results are shown in Fig. 8.

Referring to the reward function set for good agent, we know that the reward of good agent should be bigger than 0 if it escaped the attack of these adversaries. In Fig. 8(b), the rewards of good agent remain below 0, so the good agent must be attacked by adversaries. If the reward of good agent is less than -10, it means the good agent is attacked twice or more. In Fig. 8(a), the reward values of all adversaries can reach above 20 when there is one collision (before 13,000 episodes). For adversaries, the bonus value for each collision is increased by 10. Therefore, the extra values are calculated by Equation (5). When two or more collisions occur (after 13,000 episodes), the reward value stay above 30. The more collisions, the higher the reward value. From this analysis, even if the adversaries can't catch up with the good agent, it can maintain a close distance to achieve the combat mission for tracking and rounding. When there is one internal collision, the reward value of these two adversaries will be reduced by 10 as a penalty. Since the reward for good agent stays below -20 after 13,000 episodes, that means it has suffered at

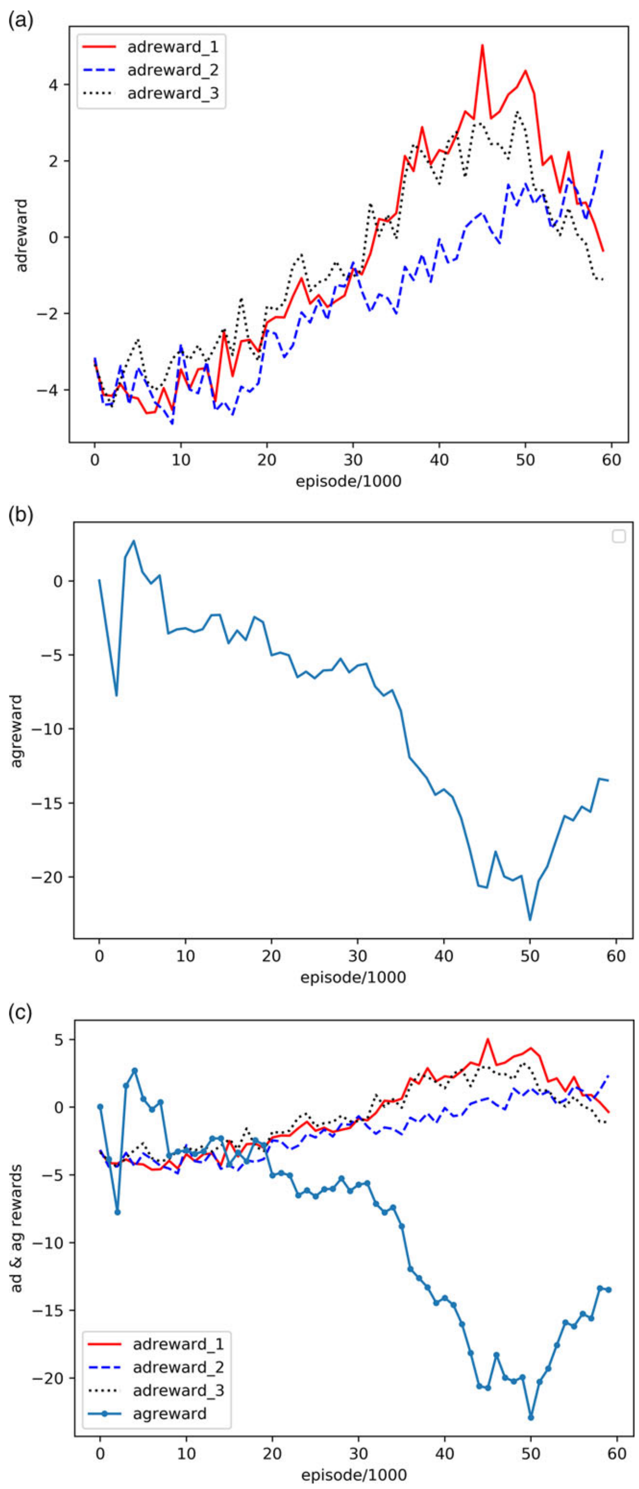

Figure 7. *The accumulative reward per episode. (a) The reward curves for three cooperative adversaries, (b) the reward curve for good agent, and (c) the reward curve for good agent versus adversaries.*

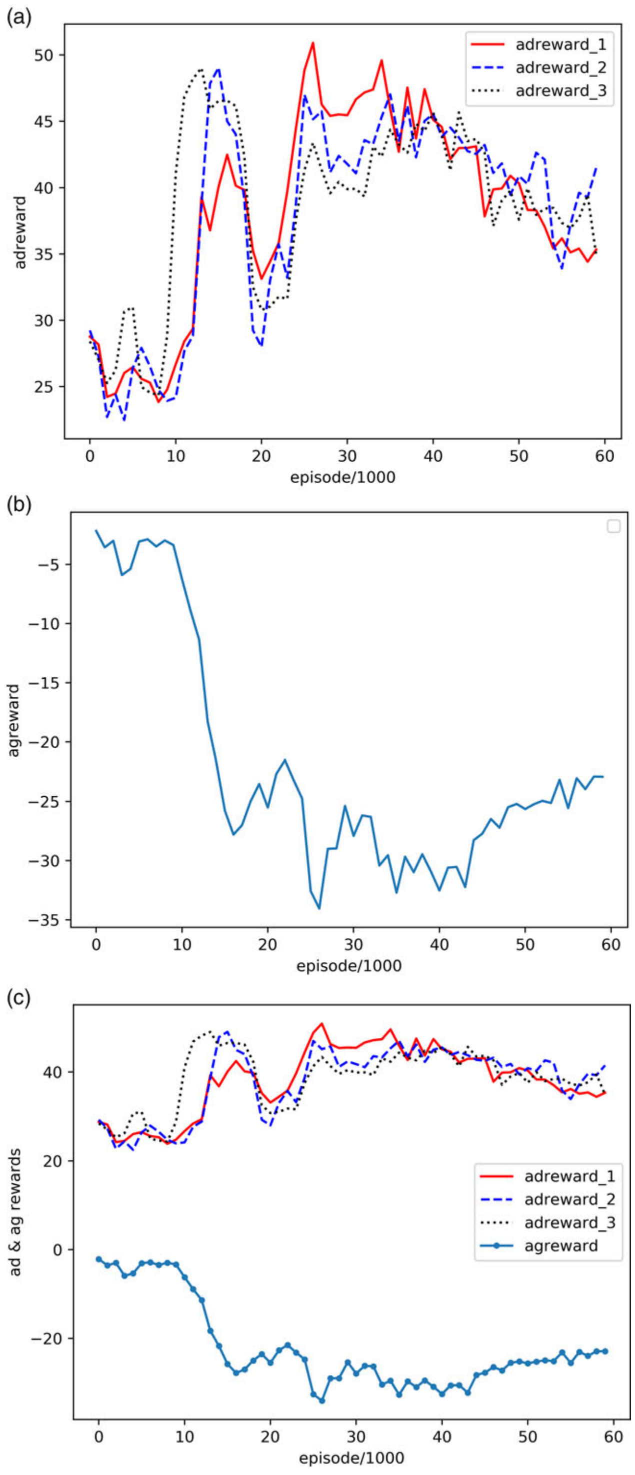

Figure 8. *The accumulative reward per episode. (a) The reward curves for three cooperative adversaries, (b) the reward curve for good agent, and (c) the reward curve for good agent versus adversaries.*

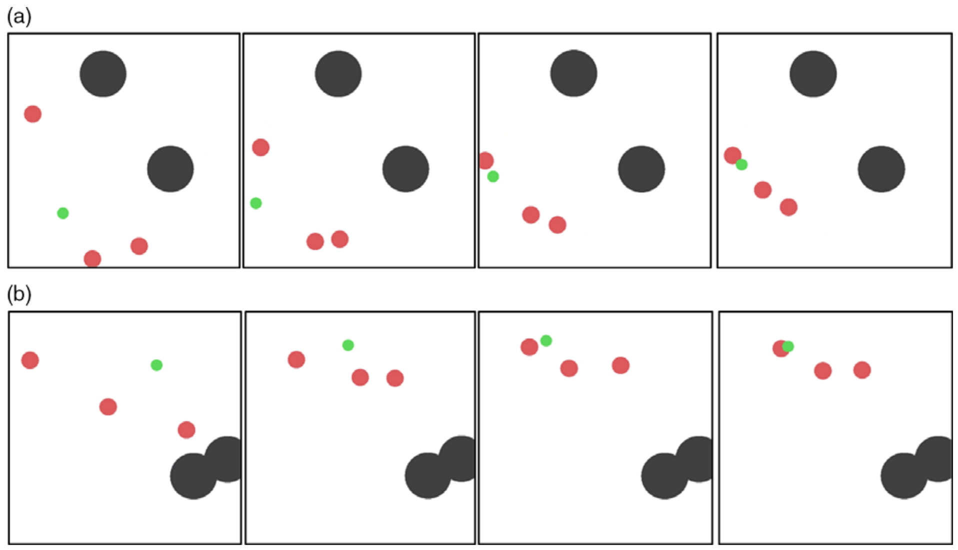

Figure 9. *The training process for round-up and confrontation with improved MADDPG algorithm. (a) Randomly generated scenario 1, and (b) scenario 2.*

least three collision attacks. At this point, the reward value of each adversary is usually greater than 40, which means that collisions within the cluster are mostly avoided. Therefore, we can conclude that the adversaries can learn the optimal policy by the improved reward mechanisms, and realise the roundup and confrontation mission without internal collisions.

The training process is presented in animated manner as follows. Two scenarios are randomly selected from 60,000 different scenarios, and one frame is output every 20,000 time steps, shown in Fig. 9(a) and (b). It can be seen that three adversaries can quickly catch up with the randomly moving agent from their original positions without internal collisions and successfully achieve the confrontation task.

5.3 Simulation comparison for MADDPG vs improved MADDPG

In order to prove that the improved MADDPG algorithm in this paper is superior to the MADDPG algorithm in Ref. [40], we carried out simulation tests with two algorithms respectively in the same application scenario.

When building the learning framework, the original MADDPG algorithm adopts MLP model. It is a fully connected three-layer neural network structure. In order to compare the advantages and disadvantages of learning framework fairly, the nodes number in hidden layers and activation functions of neural network are chosen the same. When calculating the Q-value, the original algorithm only considers the current state and environmental observation variables of all agents, while the improved algorithm adds the action variables of all agents. Therefore, the evaluation of the action will be more accurate and the learned strategy is significantly improved.

When setting the reward function, both algorithms are calculated based on the relative distance between the UAVs and the target. In original MADDPG algorithm, the cluster is considered as a whole, and a global reward function is set. But from the results of the simulation tests, this idea has a big flaw. There is no guarantee that the reward of each UAV in the cluster is increasing. This is reflected in actual flights, where there is no guarantee that every drone is trying to get close to the target. Besides, internal collision avoidance settings are also inadequate. In view of these shortcomings, we improve the reward function. On the one hand, the reward function is set for each agent in the cluster according to its relative distance from the target, which can ensure the strategy optimisation of the individual agent. From a global perspective, the reward of all agents is added up to control the overall strategy optimisation of

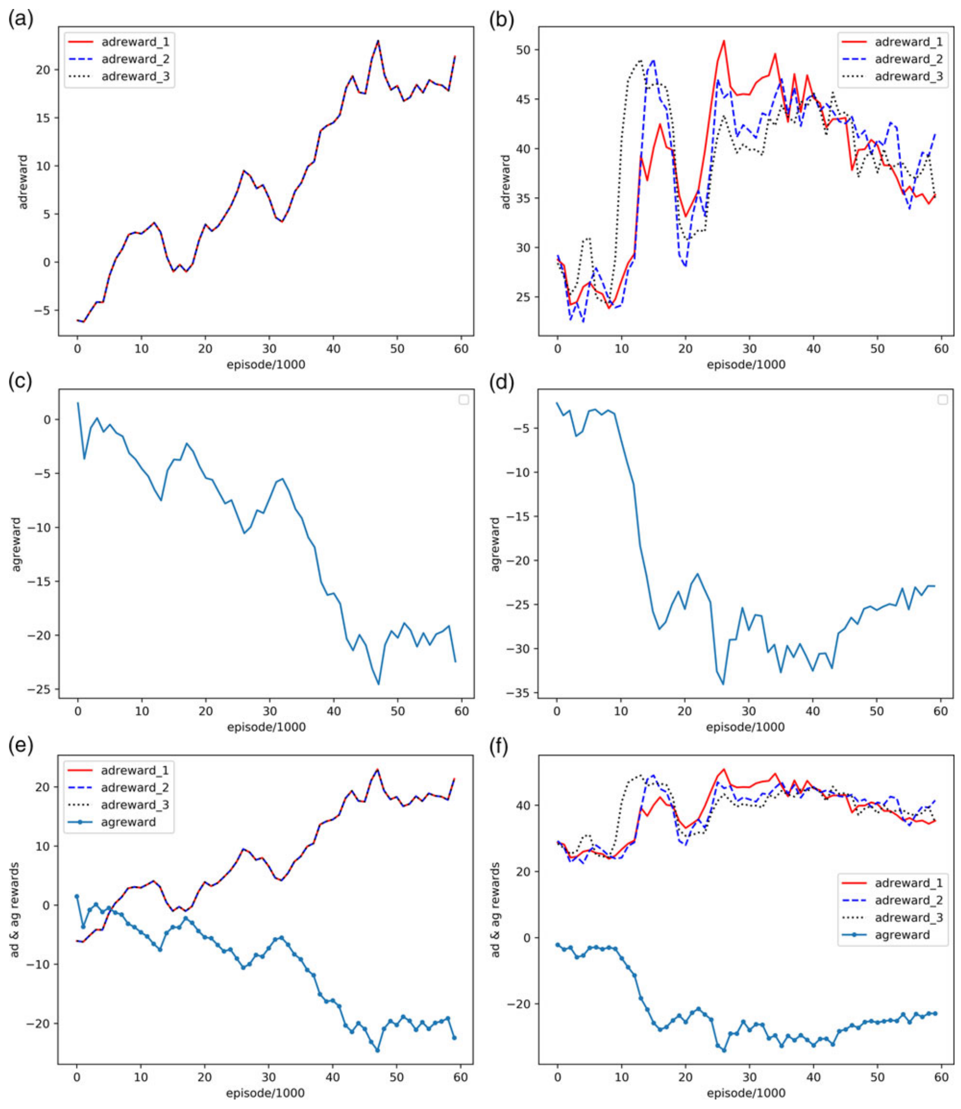

Figure 10. *Simulation comparison of reward curve for MADDPG algorithm with improved MADDPG algorithm. (a) The reward curves of three cooperative adversaries trained by MADDPG algorithm, (b) the reward curves of three cooperative adversaries trained by improved MADDPG algorithm, (c) the reward curve of good agent trained by MADDPG algorithm, (d) the reward curve of good agent trained by improved MADDPG algorithm, (e) the reward curve for good agent versus adversaries trained by MADDPG algorithm, and (f) the reward curve for good agent versus adversaries trained by improved MADDPG algorithm.*

the cluster. On the other hand, an additional collision penalty is added to the reward function of each agent. It is clear from the results that the number of internal collisions has been greatly reduced. So the optimal strategy learned by the improved MADDPG algorithm is obviously better than that obtained by the original algorithm. Under the same round-up confrontation task, the training results of the two algorithms are shown in Fig. 10.

As can be seen from the above results, the reward values of adversaries calculated with original MADDPG are obviously less than the values got with improved MADDPG. The difference between the reward value of good agent and adversaries is also significantly reduced. Through analysis, the reason

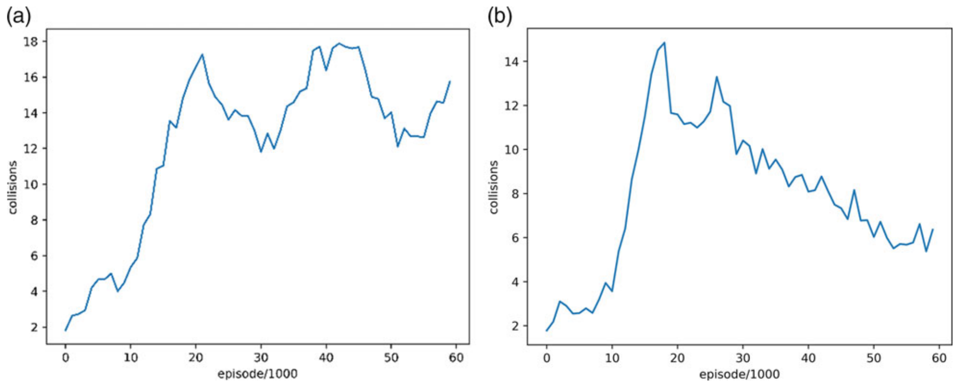

Figure 11. *Comparison of collision number with different algorithms. (a) The collision number occurred trained with MADDPG algorithm, and (b) the collision number occurred trained with improved MADDPG algorithm.*

leading to above results should be that more damage collisions occurred. Therefore, we compare the number of collisions with the two algorithms.

As can be seen from Fig. 11, the number of collisions trained with the new algorithm reduces more than double. This indicates that both obstacle avoidance and internal collision avoidance are controlled, which greatly improves the security of cluster collaboration. All of the above comparisions can demonstrate that the improved MADDPG algorithm has obvious advantages.

In order to compare with the improved MADDPG algorithm more clearly, we also present the training process in animation form and intercepte the training process in several scenarios as illustration. From Fig. 12(a), (b) and (c), we can see that the cluster trained by the original MADDPG algorithm does not have a good global concept. The distant UAV is often ignored, which is likely to result in a decline in mission completion rate. In addition, the completion of obstacle avoidance and collision avoidance is not good. The situations similar to Fig. 12(b), (c) and (d) often occur during training. This will also cause significant losses.

6.0 Conclusions and Future Work

This paper proposed a new improved Multi-Agent Reinforcement Learning algorithm, which mainly improved the learning framework and reward mechanism based on the principle of MADDPG algorithm. The action variables are introduced into Q network and P network, and used for calculation of Q value together with the state variables. The structure of Q network and P network in the new framework are no longer the same. The number of network nodes can be adjusted according to the training results. In addition, the reward mechanism is also improved. The obstacle avoidance settings are added to effectively improve the survival rate of UAV cluster. At the same time, the control of each UAV in the cluster is added, which greatly improves the ability of cooperative combat. Integrating the above improvements, the improved MADDPG algorithm is able to calculate Q values more precisely, which can also benefit the learning of optimal policy. In order to verify the feasibility of the improved MADDPG algorithm, we constantly adjust through a large number of simulation tests, and finally achieved good training results.

One disadvantage of the algorithm is that its real-time performance can not meet the actual combat requirements. And with the increase of the number of clusters, the amount of computing information increases exponentially, which makes the real-time performance of the calculation worse. We preliminarily assume that Deep Learning and Reinforcement Learning can be combined to solve the real-time problem. Through a large number of simulation experiments to accumulate data, and introduce deep learning module to learn these prior knowledge. Then use reinforcement learning to conduct online decision-making training, which is helpful to improve their independent decision-making ability.

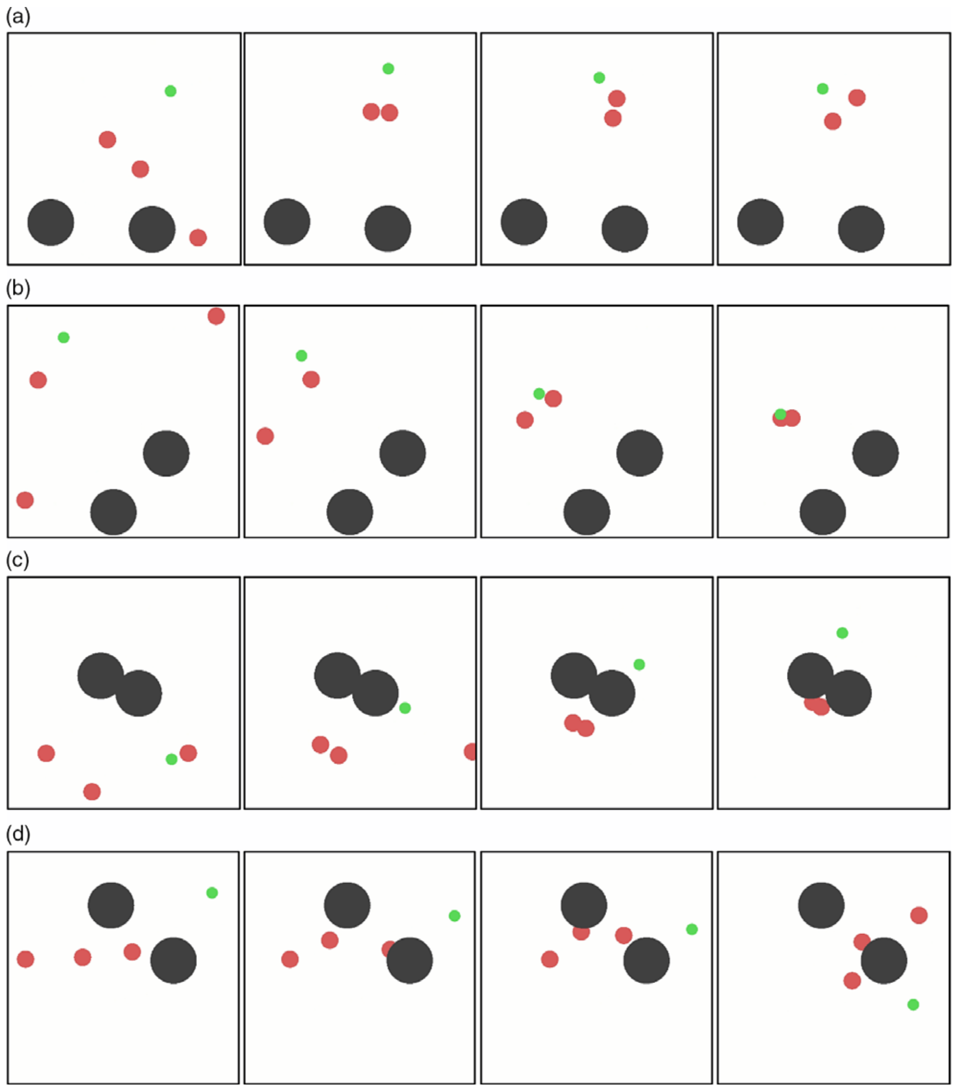

Figure 12. *The training process for round-up and confrontation with the original MADDPG algorithm. (a) and (b) are two randomly generated scenarios, in which one agent is lost, which fails to better reflect the collaborative characteristics of the cluster. (c) and (d) are two randomly generated scenarios, in which the agents frequently collides with obstacles, so the operation safety cannot be well guaranteed.*

Another weakness is that the algorithm can only apply to the problem of a fixed number of agents. In actual combat, it is very likely that the loss of the UAVs in the cluster will always occur, so the development of a variable number of agents algorithm will be the next problem to be solved. We leave these two problems to future work.

Acknowledgement. This work was partially supported by the National Natural Science Foundation of China (Nos. 11872293, 11672225), and the Program of Introducing Talents and Innovation of Disciplines (No. B18040).

Supplementary material

To view supplementary material for this article, please visit https://doi.org/10.1017/aer.2021.112

References

[1] Xing, D.J., Zhen, Z.Y. and Gong, H.J. Offense-defense confrontation decision making for dynamic UAV swarm versus UAV swarm, *Proc. Inst. Mech. Eng. G J. Aerosp. Eng.*, 2019, **233**, (15), pp 5689–5702. https://doi.org/10.1177/0954410019853982

[2] Zhang, J. and Xing, J.H. Cooperative task assignment of multi-UAV system, *Chin. J. Aeronaut.*, 2020. https://doi.org/10.1016/j.cja.2020.02.009

[3] Wang, C., Wu, L.Z., Yan, C., et al. Coactive design of explainable agent-based task planning and deep reinforcement learning for human-UAVs teamwork, *Chin. J. Aeronaut.*, 2020. https://doi.org/10.1016/j.cja.2020.05.001

[4] Imanberdiyev, N., Fu, C., Kayacan, E., et al. Autonomous navigation of UAV by using real-time model-based reinforcement learning, *14th International Conference on Control, Automation, Robotics and Vision (ICARCV 2016).* https://doi.org/10.1109/ICARCV.2016.7838739

[5] Wu, Y.H., Yu, Z.C., Li, C.Y., et al. Reinforcement learning in dual-arm trajectory planning for a free-floating space robot, *Aerosp. Sci. Technol.*, 2020, **98**. https://doi.org/10.1016/j.ast.2019.105657

[6] Dong, Y.Q., Ai, J.L. and Liu, J.Q. Guidance and control for own aircraft in the autonomous air combat: A historical review and future prospects, *Proc. Inst. Mech. Eng. G J. Aerosp. Eng.*, 2019, **233**, (16), pp 5943–5991. https://doi.org/10.1177/0954410019889447

[7] Sun, Z., Chao, T., Wang, S., et al. Ascent trajectory tracking method using time-varying quadratic adaptive dynamic programming, *Proc. Inst. Mech. Eng. G J. Aerosp. Eng.*, 2018, **233**, (11), pp 4154–4165. https://doi.org/10.1177/0954410018817613

[8] Xu, G.T., Long, T., Wang, Z., et al. Target-bundled genetic algorithm for multi-unmanned aerial vehicle cooperative task assignment considering precedence constraints, *Proc. Inst. Mech. Eng. G J. Aerosp. Eng.* 2019, **234**, (3), pp 760–773. https://doi.org/10.1177/0954410019883106

[9] Zhao, E.J., Chao, T., Wang, S.Y., et al. Multiple flight vehicles cooperative guidance law based on extended state observer and finite time consensus theory, *Proc. Inst. Mech. Eng. G J. Aerosp. Eng.*, 2016, **232**, (2), pp 270–279. https://doi.org/10.1177/0954410016683734

[10] Lowe, R., Wu, Y., Tamar, A., et al. Multi-agent Actor-Critic for mixed cooperative-competitive environments. arXiv:1706.02275v3, 2018.

[11] Liu, Y.X., Liu, H., Tian, Y.L., et al. Reinforcement learning based two-level control framework of UAV swarm for cooperative persistent surveillance in an unknown urban area, *Aerosp. Sci. Technol.*, 2020, **98**, p 105671. https://doi.org/10.1016/j.ast.2019.105671

[12] Zhen, Z.Y., Xing, D.J. and Gao, C. Cooperative search-attack mission planning for multi-UAV based on intelligent self-organized algorithm, *Aerosp. Sci. Technol.*, 2018, **76**, pp 402–411. https://doi.org/10.1016/j.ast.2018.01.035

[13] Yao, P., Wang, H.L. and Su, Z.K. Cooperative path planning with applications to target tracking and obstacle avoidance for multi-UAVs, *Aerosp. Sci. Technol.*, 2016, **54**, pp 10–22. https://doi.org/10.1016/j.ast.2016.04.002

[14] Wang, C., Li, J., Jing, N., et al. A distributed cooperative dynamic task planning algorithm for multiple satellites based on multi-agent hybrid learning, *Chin. J. Aeronaut.* 2011, **24**, (4), pp 493–505. https://doi.org/10.1016/S1000-9361(11)60057-5

[15] Sun, G.B., Zhou, R., Xu, K., et al. Cooperative formation control of multiple aerial vehicles based on guidance route in a complex task environment, *Chin. J. Aeronaut.* 2020, **33**, (2), pp 701–720. https://doi.org/10.1016/j.cja.2019.08.009

[16] Fu, X.W., Pan, J., Wang, H.X., et al. A formation maintenance and reconstruction method of UAV swarm based on distributed control, Aerosp. Sci. Technol., 2020, 104, p. 105981. https://doi.org/10.1016/j.ast.2020.105981

[17] Fu, X.W., Pan, J., Wang, H.X., et al. A formation maintenance and reconstruction method of UAV swarm based on distributed control with obstacle avoidance, *Australian and New Zealand Control Conference (ANZCC)*, 2019. https://doi.org/10.1109/ANZCC47194.2019.8945601

[18] La, H.M., Nguyen, T., Le, T.D., et al. Formation control and obstacle avoidance of multiple rectangular agents with limited communication ranges, *IEEE Trans. Control Network Syst.,* 2017, **4**, (4), pp 680–691. https://doi.org/10.1109/TCNS.2016.2542978

[19] La, H.M. and Sheng, W. Dynamic target tracking and observing in a mobile sensor network, *Robot. Autonom. Syst.*, 2012, **60**, (7), pp 996–1009. https://doi.org/10.1016/j.robot.2012.03.006

[20] Degas, A., Rantrua, A., Kaddoum, E., et al. Dynamic collision avoidance using local cooperative airplanes decisions, *CEAS Aeronaut. J.*, 2020, **11**, pp 309–320. https://doi.org/10.1007/s13272-019-00400-6

[21] Busoniu, L., Babuska, R. and Schutter, B.D. Multi-agent reinforcement learning: An overview, Srinivasan, D., & Jain, L.C. (eds). *Innovations in Multi-Agent Systems and Applications – 1*, vol. **310**. Studies in Computational Intelligence, Springer, Berlin, Heidelberg, 2010, pp 183–221. https://doi.org/10.1007/978-3-642-14435-6_7

[22] Musavi, N., Onural, D., Gunes, K., et al. Unmanned aircraft systems airspace integration: a game theoretical framework for concept evaluations, *J. Guid. Control Dyn.* 2017, **40**, (1), pp 96–109. https://doi.org/10.2514/1.G000426

[23] Petar, K., Sylvain, C. and Darwin, C. Reinforcement learning in robotics: applications and real-world challenges, *Robotics*, 2013, **2**, (3), pp 122–148. https://doi.org/10.3390/robotics2030122

[24] Das-Stuart, A., Howell, K.C., and Folta, D. Rapid trajectory design in complex environments enabled by reinforcement learning and graph search strategies, *Acta Astronaut.*, 2019, **171**, pp 172–195. https://doi.org/10.1016/j.actaastro.2019.04.037

[25] Jiang, J.X., Zeng, X.Y., Guzzetti, D., et al. Path planning for asteroid hopping rovers with pre-trained deep reinforcement learning architectures, *Acta Astronaut.*, **2020**, **171**, pp 265–279. https://doi.org/10.1016/j.actaastro.2020.03.007

[26] Mnih, V., Kavukcuoglu, K., Silver, D., et al. Human-level control through deep reinforcement learning, *Nature*, **518**, 2015, pp 529–533. https://doi.org/10.1038/nature14236.

[27] Wang, Z.Y., Freitas, N.D. and Lanctot, M. Dueling network architectures for deep reinforcement learning, *Proceedings of the International Conference on Machine Learning*, New York, USA, April 2016: 1995-2003. arXiv: 1511.06581v3.

[28] Hausknecht, M. and Stone, P. Deep recurrent q-learning for partially observable MDPs, *Association for the Advancement of Artificial Intelligence (AAAI 2015)*, 2017. arXiv: 1507.06527v4.

[29] Yang, X.X. and Wei, P. UAV navigation in high dynamic environments: A deep reinforcement learning approach, Chin. J. Aeronaut. 2020. https://doi.org/10.1016/j.cja.2020.05.011

[30] Silver, D., Lever, G., Heess, N., et al. Deterministic policy gradient algorithms, *Proceedings of the International Conference on Machine Learning*, vol. 32, 2014, pp 387–395.

[31] Duryea, E., Ganger, M. and Hu, W. Exploring deep reinforcement learning with multi q-learning, *Intell. Cont. Automat.*, 2016, **7**, (4) pp 129–144. https://doi.org/10.4236/ica.2016.74012

[32] Littman, M.L. Markov games as a framework for multi-agent reinforcement learning, *Proceedings of the 11th International Conference on Machine Learning (ICML 1994)*, Rutgers University, New Brunswick, NJ, July 1994, pp 157–163. https://doi.org/10.1016/B978-1-55860-335-6.50027-1

[33] Gong, L.G., Wang, Q., Hu, C.H., et al. Switching control of morphing aircraft based on Q-learning, *Chin. J. Aeronaut.*, 2020, **33**, (2), pp 672–687. https://doi.org/10.1016/j.cja.2019.10.005

[34] Peters, J. and Schaal, S. Policy gradient methods for robotics, *International Conference on Intelligent Robots and Systems*, 2007. https://doi.org/10.1109/IROS.2006.282564

[35] Babuska, R., Busoniu, L., and Schutter, B.D. Reinforcement learning for multi-agent systems, *Proceedings of the11th International Conference on Emerging Technologies and Factory Automation (ETFA 2006)*, IEEE, Prague, Czech Republic, 2006. http://www.dcsc.tudelft.nl

[36] Nguyen, T.T., Nguyen, N.D., and Nahavandi, S. Deep reinforcement learning for multi-agent systems: A review of challenges, solutions and applications, 2019. arXiv: 1812.11794v2.

[37] Li, C.G., Wang, M. and Yuan, Q.N. A mulit-agent reinforcement learning using actor-critic methods, *Proceedings of the 7th International Conference on Machine Learning and Cybernetics* 2008. https://doi.org/10.1109/ICMLC.2008.4620528

[38] Gupta, J.K., Egorov, M. and Kochenderfer, M. Cooperative multi-agent control using deep reinforcement learning. In Sukthankar, G. and Rodriguez-Aguilar, J. (eds.) *International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017)*, Lecture Notes in Computer Science, 106(42), Springer, Cham, 2017, pp 66–83. https://doi.org/10.1007/978-3-319-71682-4_5

[39] Guo, H.L. and Meng, Y. Distributed reinforcement learning for coordinate multi-robot foraging, *J. Intell. Robot Syst.*, 2010, **60**, pp 531–551. https://doi.org/10.1007/s10846-010-9429-4

[40] Lowe, R., Wu, Y., Tamar, A., et al. Multi-agent actor-critic for mixed cooperative-competitive environments, *Proceedings of the Neural Information Processing Systems (NIPS 2017)*. arXiv:1706.02275v3.

[41] Lillicrap, T.P., Hunt, J.J., Pritzel, A., et al. Continuous control with deep reinforcement learning, *International Conference on Learning Representations*, 2015, pp 1–14. https://doi.org/10.1016/S1098-3015(10)67722-4

[42] Nagabandi, A., Kahn, G., Fearing, R.S., et al. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning, 2017. arXiv: 1708.02596v2.

[43] Yang, Z., Merrick, K., Abbass, H., et al. Multi-task deep reinforcement learning for continuous action control, *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017, pp 3301–3307. https://doi.org/10.24963/ijcai.2017/461

[44] Baker, B., Gupta, O., Naik, N., et al. Designing neural network architectures using reinforcement learning, *International Conference on Learning Representations*, 2017. arXiv: 1611.02167v2

[45] Liu, Q.H., Liu, X.F. and Cai, G.P. Control with distributed deep reinforcement learning: Learn a better policy, 2018. arXiv: 1811.10264v2.

[46] Goecks, V.G., Leal, P.B., White, T., et al. Control of morphing wing shapes with deep reinforcement learning, *2018 AIAA Information Systems-AIAA Infotech @ Aerospace*, Kissimmee, Florida, January 2018. https://doi.org/10.2514/6.2018-2139

[47] Wen, N., Liu, Z.H., Zhu, L.P., et al. Deep reinforcement learning and its application on autonomous shape optimization for morphing aircrafts, *J. Astronaut.*, 2017, **38**, pp 1153–1159. https://doi.org/10.3873/j.issn.1000-1328.2017.11.003

[48] Xu, D., Hui, Z., Liu, Y.Q., et al. Morphing control of a new bionic morphing UAV with deep reinforcement learning, *Aerosp. Sci. Technol.*, 2019, **92**, pp 232–243. https://doi.org/10.1016/j.ast.2019.05.058

[49] La, H.M. Multi-robot swarm for cooperative scalar field mapping, *Handbook of Research on Design, Control, and Modeling of Swarm Robotics*, 2015. https://doi.org/10.4018/978-1-4666-9572-6.ch014

[50] La, H.M., Sheng, W. and Chen, J. Cooperative and active sensing in mobile sensor networks for scalar field mapping, *IEEE Trans. Syst. Man Cybern. Syst.*, 2015, **45**(1), pp 1–12. https://doi.org/10.1109/TSMC.2014.2318282

[51] Adepegba, A.A., Miah, S. and Spinello, D. Multi-agent area coverage control using reinforcement learning, *Proceedings of the 29th International Florida Artificial Intelligence Research Society Conference*, 2016, pp 368–373. http://dx.doi.org/10.20381/ruor-5715

[52] Pham, H.X., La, H.M., Feil-Seifer, D., et al. Cooperative and distributed reinforcement learning of drones for field coverage, 2018. arXiv: 1803.07250v1.

Cite this article: Xu D. and Chen G. (2022). Autonomous and cooperative control of UAV cluster with multi-agent reinforcement learning. *The Aeronautical Journal*, **126**, 932–951. https://doi.org/10.1017/aer.2021.112