symmetry

MDPI

Article

Building a Connected Communication Network for UAV Clusters Using DE-MADDPG

Zixiong Zhu [1,2], Nianhao Xie [1,2], Kang Zong [3] and Lei Chen [3,*]

[1] Department of Applied Mechanics, Institute of Aerospace Science, National University of Defense Technology, Changsha 410073, China; zixiongzhu524@nudt.edu.cn (Z.Z.); xienianhao@nudt.edu.cn (N.X.)
[2] Hunan Key Laboratory of Intelligent Planning and Simulation for Aerospace Missions, Changsha 410073, China
[3] Defense Innovation Institute, Chinese Academy of Military Science, Beijing 100071, China; zongkang12@nudt.edu.cn
* Correspondence: chenl@nudt.edu.cn

Abstract: Clusters of unmanned aerial vehicles (UAVs) are often used to perform complex tasks. In such clusters, the reliability of the communication network connecting the UAVs is an essential factor in their collective efficiency. Due to the complex wireless environment, however, communication malfunctions within the cluster are likely during the flight of UAVs. In such cases, it is important to control the cluster and rebuild the connected network. The asymmetry of the cluster topology also increases the complexity of the control mechanisms. The traditional control methods based on cluster consistency often rely on the motion information of the neighboring UAVs. The motion information, however, may become unavailable because of the interrupted communications. UAV control algorithms based on deep reinforcement learning have achieved outstanding results in many fields. Here, we propose a cluster control method based on the Decomposed Multi-Agent Deep Deterministic Policy Gradient (DE-MADDPG) to rebuild a communication network for UAV clusters. The DE-MADDPG improves the framework of the traditional multi-agent deep deterministic policy gradient (MADDPG) algorithm by decomposing the reward function. We further introduce the reward reshaping function to facilitate the convergence of the algorithm in sparse reward environments. To address the instability of the state-space in the reinforcement learning framework, we also propose the notion of the virtual leader–follower model. Extensive simulations show that the success rate of the DE-MADDPG is higher than that of the MADDPG algorithm, confirming the effectiveness of the proposed method.

Keywords: DE-MADDPG; UAV cluster; connected communication network; reward reshaping

Citation: Zhu, Z.; Xie, N.; Zong, K.; Chen, L. Building a Connected Communication Network for UAV Clusters Using DE-MADDPG. *Symmetry* **2021**, *13*, 1537. https://doi.org/10.3390/sym13081537

Academic Editor: Saeid Nahavandi

Received: 29 July 2021
Accepted: 19 August 2021
Published: 20 August 2021

1. Introduction

The increasing application of unmanned aerial vehicles (UAVs) has resulted in complex scenarios. Apart from military operations such as target destruction and cooperative investigation [1], civilian applications can also benefit from UAV technology, such as environmental monitoring, precision agriculture [2,3], disaster relief [4], and traffic monitoring [5]. As it becomes increasingly difficult for a single UAV to meet the needs of complex tasks [6], the recent development of technologies such as cluster intelligence has enabled applications with small UAV cluster systems. Using multiple UAVs in a system is challenging due the communication problems between them, such as the damage of some UAVs and the interruption of communication links, which may affect the data transmission within the cluster. Therefore, rebuilding a connected communication network for a cluster with communication problems is essential for the cooperation of UAVs [7].

Recently, deep reinforcement learning (DRL) methods have been proposed to solve extremely high complexity problems which are not often solvable using traditional control

methods [8]. The DRL provides new ideas for solving the problems of large-scale time-varying systems [9]. To control the movement of the UAV clusters, however, the use of the traditional DRL method is challenged by the state-space dimension and the difficulty of convergence [10].Using one agent to control all UAVs does not reflect the relationship between UAVs [11]. In the multi-agent scenario, other agents update their strategies independently. This may make the environment unstable from the perspective of any agent. A stable Markov process is the prerequisite for the convergence of the algorithm [10].

To ensure as stable a Markov process as possible, we propose to use a MADDPG algorithm [12] and the improved MADDPG algorithm. The MADDPG algorithm uses a centralized training and distributed execution framework. Each UAV is regarded as an agent and outputs actions based on its observations and the value of its actor network. In the centralized training, the critic network of each agent is updated by the rewards from the environment and the states and actions of all agents, and the actor network of each agent is updated by the score from the critic network. From a global perspective, this is a stable Markov process, which can alleviate the instability of the multi-agent environment to a certain extent. More importantly, this algorithm separates training and execution. After training the model in an offline simulation environment, it is uploaded to the UAV control module. During the online execution, each UAV outputs control commands based on the trained model, which effectively saves time for making decisions on the fly. In contrast to the single-agent algorithm, MADDPG can effectively solve the problem of cooperation between UAVs. The main contributions of this work are as follows.

1. We construct a partially observable Markov decision process (POMDP) framework for a UAV cluster to build a connected network. To the best of our knowledge, this is the first use of multi-agent reinforcement learning to build a connected network of UAV clusters. In response to this problem, we make improvements to the MADDPG framework and propose the DE-MADDPG algorithm.
2. We rationally apply the theory of mobile consistency to set the reward reshaping function, which guides the agent to learn an effective policy faster and solves the problem of difficult convergence caused by the sparse reward.
3. To the best of our knowledge, our work is the first to control a cluster based on RL in open space. We introduce the idea of the virtual leader–follower model to solve the problem of an unstable state space and achieve remarkable results.

The rest of the paper is organized as follows. In Section 2, we present work related to our study. Section 3 presents the model of our work. In Section 4, we present the MADDPG and the DE-MADDPG, which is followed by the simulation results and discussions in Section 5. Finally, the conclusion of this paper and the future work are presented in Section 6.

2. Related Work

There is a large amount of research on how to build a connected communication network for UAV clusters. Ramanatha et al. [13] proposed a method based on heuristic algorithms and controlled the UAV transmission power to enable a dynamic network topology and achieve network connectivity. Nevertheless, due to the limited transmit power, it is almost impossible for UAVs to communicate with each other if the distance between them becomes larger than their corresponding communication range. Kashyap et al. [14] enhanced the connectivity of the cluster network by adding relay nodes. Adding additional relay nodes, however, further increases the number of UAVs.

Models such as those presented by Reynolds et al. [15], Vicsek et al. [16], and Conzin et al. [17] are based on the theory of cluster mobility and control the movement of UAVs to construct a connected network. In a distributed situation, however, UAVs that are too marginal are not able to obtain neighbor information and are thus unable to connect with others. According to the estimation result of algebraic connectivity, Yang et al. [18] use the gradient method to derive the agent's motion control amount, so that the multi-agent can maintain a certain communication network during the movement process. Hao et al. [19]

designed a distributed agent cluster control protocol based on the control strategy of the potential function gradient and also realized the maintenance of the communication network of the multi-agent system. Dai et al. [20] discussed the optimal trajectory planning of multiple spacecraft from a disconnected state to build a connected network, which has certain reference significance for how to control the movement of multiple UAVs to construct a connected network.

With the development of reinforcement learning (RL), some scholars have applied RL to cluster control and achieved remarkable results. To control the UAV to search for an unknown area, Zanol et al. [21] used the Q-learning algorithm to train a single agent that controls a drone searching for an unknown area. A similar approach was taken by Klaine et al. [22], and they controlled the UAVs to build an emergency network for a disaster area. In [23], to maximize the coverage of the UAV cluster, Liu et al. proposed a DRL-based energy-efficient control method for coverage and connectivity (DRL-EC3). Wang et al. [9] proposed a trajectory control algorithm of UAVs based on MADDPG, guiding the movement of ground users. In [24], to detect targets and avoid obstacles, Han et al. proposed the simultaneous target assignment and path planning (STAPP) approach based on MADDPG.

Similar to these studies, we propose a distributed UAV control method based on DE-MADDPG to rebuild a connected communication network for a cluster with communication problems.

3. Model

With a GPS receiver and a set of radio transceivers, each UAV can obtain its real-time position and communicate with other UAVs if their physical distance is small enough [25]. When the cluster communication network is intact, each UAV can communicate with all other UAVs through the communication link. To improve the stability of the communication network between the UAV clusters, it is necessary to ensure that each UAV can be connected to at least one other UAV. Otherwise, some individual UAVs in the clusters may move apart from the clusters. Some malfunctions, such as damage to some UAVs and the interruption of communication links, may change the topology of the cluster and affect the communication of other UAVs. The cluster must rebuild the connected network after malfunctions.

3.1. Coordinate System Establishment

We consider the launching point as the origin of the north–east–down coordinate system (denoted as O_{XYZ}), as shown in Figure 1.

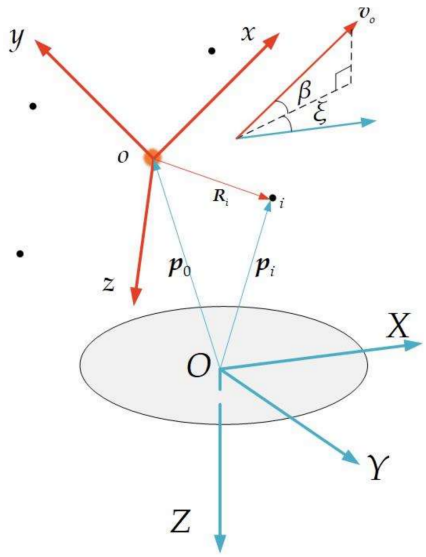

Figure 1. The positional relationships of two coordinate systems.

We index the UAV in the cluster as i, $i \in \{1, 2, \cdots, n\}$. The position vector of i is p_i. The origin of cluster coordinate system (denoted as o_{xyz}) coincides with the initial barycenter o of the cluster. The position vector p_o and the velocity vector v_o of the barycenter of the cluster in the O_{XYZ} system are given by Equation (1). The x axis coincides with the cluster velocity v_o, and the z axis lies in the vertical plane and is perpendicular to the x axis downward; the y axis direction is determined by the right-hand rule, as shown in Figure 1.

$$\begin{cases} p_o = \frac{1}{n}\sum_{i=1}^{n} p_i \\ v_o = \dot{p_o} \end{cases} \tag{1}$$

The coordinate conversion matrix $M_{O\to o}$ from the O_{XYZ} system to the o_{xyz} system is given by Equations (2)–(4):

$$M_{O\to o} = M_2(\beta)M_3(\xi) \tag{2}$$

$$M_2(\beta) = \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix} \tag{3}$$

$$M_3(\xi) = \begin{bmatrix} \cos\xi & \sin\xi & 0 \\ -\sin\xi & \cos\xi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

where β is the flight path angle of the barycenter of the cluster, and ξ is the heading angle.

In the O_{XYZ} system, vector $\boldsymbol{R}_i = \boldsymbol{p}_i - \boldsymbol{p}_o$.

In the o_{xyz} system, the position vector $\boldsymbol{R'}_i$ of UAV i is given by Equation (5). We model the motion of UAVs in the o_{xyz} system.

$$\boldsymbol{R'}_i = (\boldsymbol{p}_i - \boldsymbol{p}_o)M_{O\to o} \tag{5}$$

3.2. Virtual Leader–Follower Model

Using the traditional method to construct the state space, the RL algorithm is not likely to converge because of the instability of the state space. To address this issue, we introduce a virtual navigator.

When the cluster communication network is intact, UAVs can exchange position information with each other and calculate the real-time position $\boldsymbol{p}_o$ of the barycenter. We assume that at time t_0, the cluster communication fails. However, the position vector $\boldsymbol{p}_o(t_0)$ and the velocity vector $\boldsymbol{v}_o(t_0)$ of the barycenter in the O_{XYZ} system are obtained by each UAV. The virtual navigator takes the position vector $\boldsymbol{p}_o(t_0)$ as the initial position and moves in a uniform straight line at the velocity $\boldsymbol{v}_o(t_0)$. The movement of the virtual navigator is not affected by other UAVs.

Each UAV can easily calculate the real-time position of the virtual navigator, although the position information of other UAVs cannot be obtained. To better describe the cluster movement, we take the virtual navigator as the origin of the o_{xyz} system. The velocity $\boldsymbol{v}_o(t_0)$ direction of the virtual navigator is in the x axis direction, and the definitions of y and z axes are the same.

In the o_{xyz} system, based on Equation (6), the velocity vector $\boldsymbol{v}'_i$ of UAV i is given by

$$v'_i = (v_i - v_o(t_0))M_{O\to o} \tag{6}$$

where v_i is the velocity vector of UAV i in the O_{XYZ} system.

The virtual navigator model reduces the state space of the agent by taking the position of the UAV relative to the virtual navigator as the state of RL and makes it possible to control the UAV cluster in the open space based on RL. Although the virtual navigator is introduced to assist the agent in training, the algorithm remains a decentralized algorithm.

3.3. Network Connectivity Model

The transmission of wireless signals in the space results in the damping of the received power. Based on the Friis transmission equation, the transmitting power P_t^i of node f_i and the receiving power P_r^j of receiving node f_j satisfies Equation (7):

$$\frac{P_r^j}{P_t^i} = \left(\frac{\lambda}{4\pi d_{ij}}\right)^2 G_t G_r \tag{7}$$

where G_t is the antenna gain of the transmitting node, G_r is the antenna gain of the receiving node, d_{ij} is the distance between the transmitting node f_i and the receiving node f_j, λ is the wavelength of the radio frequency (RF) signal, and $\frac{\lambda}{4\pi d_{ij}}$ is called the free-space loss factor. Usually, G_t, G_r are constant values.

The signal-to-noise ratio of the receiving node f_j in decibels (dB) is given by

$$SNR = 10\lg\left(\frac{P_r^j}{P_n}\right) \tag{8}$$

where P_n is the power of noise and interference. The signal-to-noise ratio threshold is SNR_{th}. If the signal-to-noise ratio threshold satisfies Equation (9), the received signal is detectable.

$$SNR \geq SNR_{th} \tag{9}$$

Based on Equations (7)–(9), the maximum transmission distance $d_{\max}$ between the transmitter f_i and receiver f_j can be determined by Equation (10), which indicates the communication range of two UAV nodes based on their distance.

$$d_{\max} = \frac{\lambda}{4\pi}\sqrt{\frac{P_t^i G_t G_r}{10^{SNR_{th}/10} P_n}} \tag{10}$$

Assuming that the UAV cluster network is homogeneous, each UAV can be represented as a node, and the two-way communication links between UAVs are represented by the edges. Therefore, the UAV nodes $\boldsymbol{f} = \{f_1, f_2, \cdots, f_n\}$ and their corresponding communication links $\boldsymbol{e} = \{e_1, e_2, \cdots e_m\}$ form a three-dimensional network topology, expressed as a graph $G(\boldsymbol{f}, \boldsymbol{e})$, and m is the number of edges in the graph. In Figure 2, we show a cluster with a communication malfunction, and the UAV on the left cannot communicate with the UAV on the right.

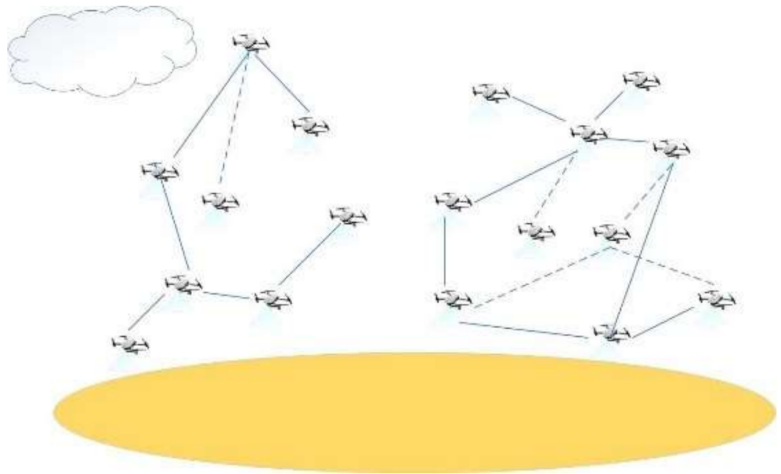

Figure 2. A schematic of the network topology.

The connectivity of this network can be modeled using an adjacency matrix of $G(\boldsymbol{v}, \boldsymbol{e})$, which is defined as $A = (a_{ij})_{n\times n}$.

$$a_{ij} = \begin{cases} 1 & \text{if } d_{ij} \le d_{\max}, i \ne j \\ 0 & \text{otherwise} \end{cases}, \ i, j \in \{1, \cdots, n\} \tag{11}$$

where $a_{ij} = 1$ means that f_i, f_j are connected. If all the entries of the matrix $P(A) = I + A + A^2 + \cdots + A^{n-1}$ are non-zero, the UAV cluster network is fully connected [18,26].

4. Methods

The multi-agent reinforcement learning (MARL) algorithm [27] is a solution model framework for the collaborative control of UAV clusters. It enables multiple agents to complete complex tasks through collaborative decision-making in a high-dimensional, dynamic environment. The MARL is autonomous, distributed, coordinated, and near to the optimal joint policy based on the core idea of centralized learning and distributed execution. It is based on the actor–critic framework [28] and effectively solves the problems of non-stationarity in the multi-agent environment and the failure of experience playback.

4.1. Markov Game for Multi-UAV Cooperation

We simulate the training process of reinforcement learning based on a locally observable Markov game [29]. The Markov process of n agents is represented by a high-dimensional tuple $< S, A, R, P, \gamma >$, where $S = [s_1, s_2, \cdots, s_n]$ denotes the state space of the Markov decision process, $A = [a_1, a_2, \cdots, a_n]$ is the joint action set of all agents, $R = [r_1, r_2, \cdots, r_n]$, r_i is the reward of the agent i, $P : S \times A \times S \to [0, 1]$ is the state transfer function, and γ is the attenuation coefficient of the cumulative discount reward [30].

In a multi-agent system, all agents execute a joint policy to generate a new state. The reward of each agent depends on the joint policy executed by all of them. According to the relationship between the agents, MARL can be divided into complete cooperation, complete competition, and a competition/cooperation hybrid [31]. Our objective is to train the UAV clusters to learn the policy of constructing connected networks with complete cooperation. In this setting, the objective of each agent is to maximize the common reward.

4.2. Multi-Agent DDPG Approach

The MADDPG algorithm is an extension of the DDPG algorithm considering a multi-agent environment, where each agent has independent actor and critic networks [12]. The MADDPG algorithm has been improved based on the traditional A–C algorithm, allowing the critic network to use the strategies of other agents to learn.

As shown in Figure 3, the MADDPG adopts the framework of centralized training with decentralized execution. Each agent is associated with an actor network and a centralized critic network. In the training, a single agent observes its state, outputs actions based on the actor network, and then obtains the corresponding rewards and new states from the environment. The critic network obtains the action and state information of all agents and outputs the state–action value for the state and action from a global perspective. Then, the actor network is updated by the state–action value. In the execution, without the critic network, the actor network outputs the actions based on the local observation.

The joint policy is denoted by $\pi = [\pi_1, \pi_2, \cdots, \pi_n]$, which is defined by the policy parameter $\theta = [\theta_1, \theta_2, \cdots, \theta_n]$. The cumulative expected reward of the agent i is

$$J(\theta_i) = E_{s\sim p^{\pi}, a_i\sim\pi_i}\left[\sum_{t=0}^{\infty} \gamma^t r_i^t\right] \tag{12}$$

where p^{π} is the state distribution, t is the instant time, θ_i is the probability distributions function of action and is implicit in policy π_i, and the operator E is the expectation operator.

For the deterministic policy gradient algorithm, the policy gradient is given by [32]

$$\nabla J(\theta_i) = E_{s,a\sim D}\left[\nabla_{\theta_i}\pi_i(a_i|o_i)\nabla_{a_i}Q_i^{\pi}(s,a_1,\cdots,a_n)\big|_{a_i=\pi_i(o_i)}\right] \tag{13}$$

where D is the experience replay buffer consisting of a series of tuples $(s,s',a_1,\ldots,a_n,r_1,\ldots,r_n)$ which record the experience of all agents, o_i is the observation of the agent i, $Q_i^{\pi}(s,a_1,\cdots,a_n)$ is the centralized state–action value function of the agent i, $s=[o_1,\cdots,o_n]$ consists of the observations of all agents, and $s'=[o'_1,\cdots,o'_n]$ consists of the new observations of all agents after executing the actions. Note that different agents have different state–action value functions, which are the basis of cooperation and competition amongst multi-agents.

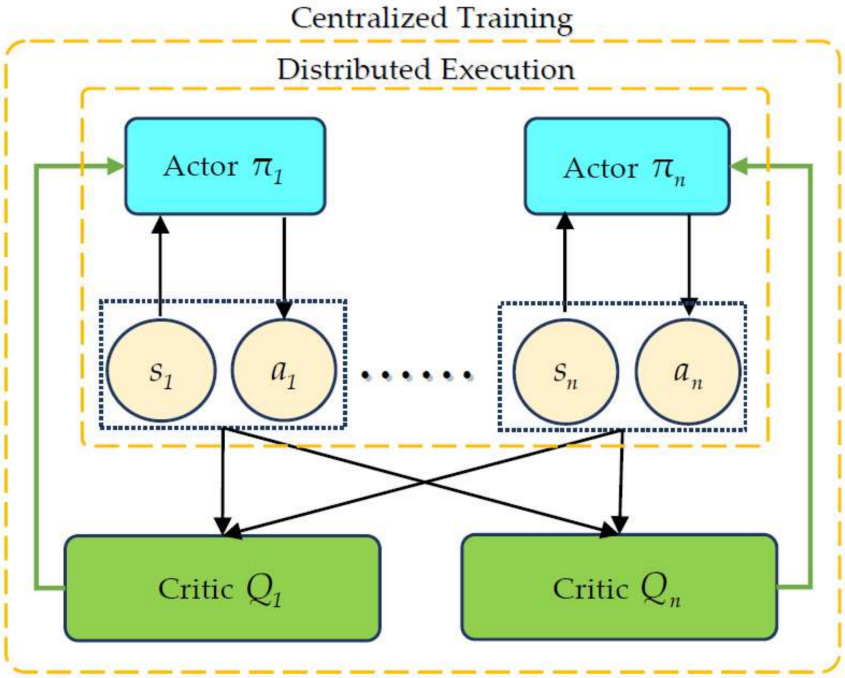

Figure 3. The framework of the MADDPG.

The centralized critic network Q_i of the agent i is updated as

$$\begin{aligned} L(\theta_i) &= E_{s,a,r,s'}\left[(Q_i^{\pi}(s,a_1,\cdots,a_n)-y)^2\right] \\ \text{where } y &= r_i+\gamma Q_i^{\pi'}(s',a'_1,\ldots,a'_n)\big|_{a'_j=\pi'_j(o'_j)} \end{aligned} \tag{14}$$

where $\pi'=[\pi'_1,\cdots,\pi'_n]$ is the target policy vector with the lagging update parameter θ'_i.

In the MADDPG algorithm, the critic network calls the global information update network, whereas the actor network only uses local observation information. For any $\pi_i \neq \pi'_i$, there is $P(s'|s,a_1,\ldots,a_N)=P(s'|s,a_1,\ldots,a_N,\pi'_1,\ldots,\pi'_n)$. Since the actions of the agent are known, the real-time policy is changed and the environment becomes stable [12].

4.3. DE-MADDPG Approach

The DE-MADDPG algorithm is an extended version of the MADDPG algorithm, which improves the network architecture of the MADDPG algorithm [33]. The MADDPG algorithm implements centralized training through a global centralized critic network. As shown in Figure 4, the DE-MADDPG algorithm has a global centralized critic network shared amongst all agents, and each agent has a local critic network [34].

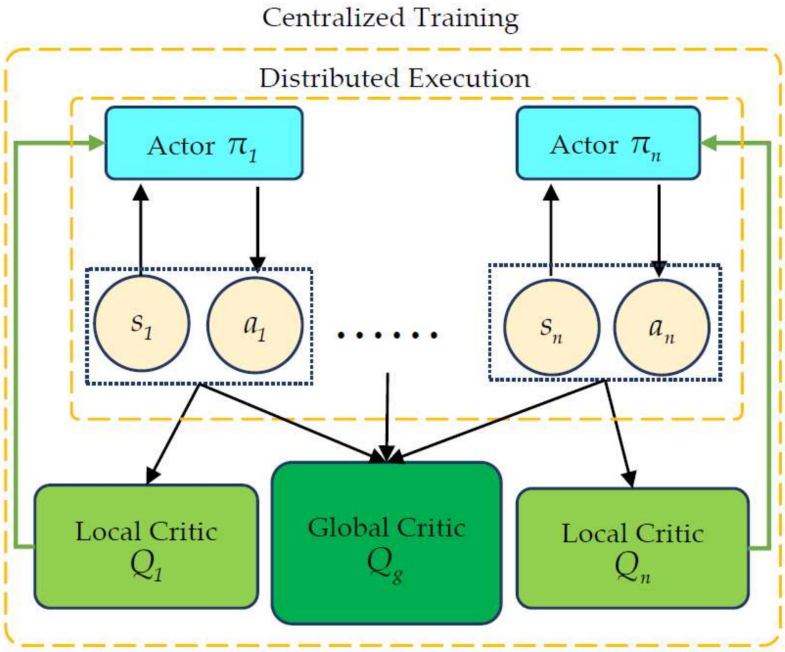

Figure 4. The framework of the DE-MADDPG.

The purpose of the improvement is to maximize local and global rewards through reward decoupling. The role of the global critic network is to maximize the global reward, whereas the role of the local critic network is to maximize the local reward. The DE-MADDPG algorithm is therefore a centralized control and distributed execution architecture. During the training phase, the state and action information of other agents are needed, but it is only necessary to output actions based on the agent's state during execution.

The main idea of the DE-MADDPG algorithm is to combine the MADDPG with a single agent DDPG, where the policy gradient is

$$\begin{aligned}\nabla J(\theta_i) = &\underbrace{E_{s,a\sim D}\left[\nabla_{\theta_i}\pi_i(a_i|o_i)\nabla_{a_i}Q_{\psi}^{\pi}(s,a_1,\ldots,a_n)\Big|_{a_i=\pi_i(o_i)}\right]}_{\text{MADDPG}}\\ &\underbrace{+E_{o_i,a_i\sim D}\left[\nabla_{\theta_i}\pi_i(a_i|o_i)\nabla_{a_i}Q_{\varphi_i}^{\pi}(o_i,a_i)\Big|_{a_i=\pi_i(o_i)}\right]}_{\text{DDPG}}\end{aligned} \tag{15}$$

where D is the experience replay buffer, Q_{φ_i} is the local critic network of the agent i with the parameter φ_i, and Q_{ψ} is the global critic network with the parameter ψ [33].

The two critic networks of the DE-MADDPG algorithm correspond to two loss functions, respectively. The global critic network Q_{ψ} is updated as

$$\begin{aligned}L(\psi) &= E_{s,a,r,s'}\left[\left(Q_{\psi}^{\pi}(s,a_1,\ldots,a_n)-y_g\right)^2\right]\\ \text{where } y_g &= r_g+\gamma Q_{\psi'}^{\pi'}(s',a'_1,\ldots,a'_n)\Big|_{a'_i=\pi'_i(o'_i)}\end{aligned} \tag{16}$$

where $\pi' = (\pi'_1, \pi'_2, \ldots, \pi'_n)$ are the target policies with the parameters $\theta' = (\theta'_1, \theta'_2, \cdots, \theta'_n)$, and $Q_{\psi'}$ is the target global critic network with the parameter ψ'.

The local critic network Q_{φ_i} of an agent i is updated as

$$\begin{aligned} L(\varphi_i) &= E_{o,a,r,o'}\left[\left(Q^{\pi}_{\varphi_i}(o_i, a_i) - y_l\right)^2\right] \\ \text{where } y_l &= r^i_l + \gamma Q^{\pi'}_{\varphi'_i}\left(o'_i, a'_i\right)\Big|_{a'_i=\pi'_i(o'_i)} \end{aligned} \tag{17}$$

where $Q_{\varphi'_i}$ is the target local critic network with the parameter φ'_i.

The pseudo-code of the DE-MADDPG is shown in the following Algorithm 1.

Algorithm 1: Training Process of the DE-MADDPG.

Initialize: main global critic network Q_ψ with the parameter ψ
Initialize: target global critic network $Q_{\psi'}$ with the parameter ψ'
Initialize: critic network Q_{φ_i} with the parameter φ_i of each agent $i = 1, 2, \cdots, n$
Initialize: target critic network $Q_{\varphi'_i}$ with the parameter φ'_i of each agent i
Initialize: actor network π_i with the parameter θ_i of each agent i
Initialize: target actor network π'_i with the parameter θ'_i of each agent i
Initialize: replay buffer D
1: **for** episode = 1 to Max-episodes **do**
2: **for** step t = 1 to Max-step **do**
3: Get initial state $\boldsymbol{s} = [o_1, \cdots, o_n]$
4: Chooses action $a_i = \pi_i(o_i)$
5: Execute actions $a = [a_1, \cdots, a_n]$
6: Receive new state $\boldsymbol{s}' = [o'_1, \cdots, o'_n]$, local rewards $r_l = [r_1, \cdots, r_n]$, and global reward r_g
7: Store $\left(\boldsymbol{s}, a, r_l, r_g, \boldsymbol{s}'\right)$ into D
/* Train global critic */
8: Sample a random minibatch of S samples $\left(\boldsymbol{s}^j, \boldsymbol{a}^j, r^j_g, \boldsymbol{s}'^j\right)$ from D
9: Set $y^j_g = r^j_g + \gamma Q^{\pi'}_{\psi'}\left(\boldsymbol{s}'^j, a'_1, \ldots, a'_n\right)\Big|_{a'_i=\pi'_i(o'^j_i)}$
10: Update global critics Q_ψ by minimizing the loss
$\frac{1}{S}\sum_j\left(Q^{\pi}_{\psi}\left(s^j, a^j_1, \ldots, a^j_n\right) - y^g_j\right)^2$
11: Update target critics $Q_{\psi'}$:
$\psi'_i \leftarrow \tau\psi_i + (1-\tau)\psi'_i$
/* Train local critics and update actor network */
12: **for** agent $i = 1$ to **do**
13: Sample a random minibatch of K samples $\left(\boldsymbol{s}^j, \boldsymbol{a}^j, \boldsymbol{r}^j_l, \boldsymbol{s}'^j\right)$ from D
14: Set $y^j_i = r^j_i + \gamma Q^{\pi'}_{\varphi'_i}\left(o'^j_i, \pi'_i\left(o'^j_i\right)\right)$
15: Update local critic $Q^{\pi}_{\varphi_i}$
$\frac{1}{K}\sum_j\left(Q^{\pi}_{\psi_i}\left(o^j_i, a^j_i\right) - y^j_i\right)^2$
16: Update actor network:
$\theta_i \leftarrow \theta_i \frac{1}{S}\sum_j \nabla_{\theta i}\pi_i\left(a^j_i\middle|o^j_i\right)\nabla_{a_i}Q^{\pi}_{\psi}\left(s^j, a^j_1, \ldots, a^j_n\right) \quad + \nabla_{\theta_i}\pi_i\left(a^j_i\middle|o^j_i\right)\nabla_{a_i}Q^{\pi}_{\varphi_i}\left(o^j_i, a^j_i\right)$
17: **end for**
18: Update target critic network and target actor network for each agent i:
$\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$
$\varphi'_i \leftarrow \tau\varphi_i + (1-\tau)\varphi'_i$
19: **end for**
20: **end for**

4.4. Design of DE-MADDPG

We set up a reward function based on the goal of building a connected network and trained the UAV cluster based on the DE-MADDPG algorithm. In the following, we present the application of the DE-MADDPG algorithm in this task in the four involved elements, including agent, state, action, and reward.

Agent: The n UAVs are regarded as n agents. In reality, each UAV can only obtain its state. However, in the training phase, it is allowed to use the state and action information of all UAVs to train the agents.

State: The state of each agent is the location coordinates of the UAV in the cluster coordinate system, represented by a vector (x, y, y).

Action: The action of each agent is the velocity of the UAV in the north–east–down coordinate system, denoted by a vector (v_i, β^i, ξ^i), where $v_i = \left|v^{'}{}_i\right|$, β^i is the flight path angle of the UAV i and ξ^i is the heading angle.

Reward: Building a connected network of UAV clusters based on reinforcement learning is a sparse reward problem. The feasible solution is to define a reward shaping function that guides the agent to the ultimate goal. In the case of the connectivity problem, the ultimate goal is to achieve a connected network within a limited time. Therefore, we introduce the aggregation tendency rule to construct the reward reshaping function. During the movement of the cluster, each UAV that is moving to the center facilitates communication and interconnection within the cluster. Based on this, we consider a Gaussian shaping reward function. At time t, we denote the distance between the UAV i and the virtual navigator by d_i, and the difference between this distance at the present time instant and at the previous time instant is denoted by $\Delta d_i = d_i(t) - d_i(t-1)$. The shaping reward function r_i of the agent i is therefore defined as

$$r_i = \begin{cases} -\exp(d_i/\sigma_1) & d_i > l_{\max}, \Delta d_i \geq 0 \\ \exp(d_i/\sigma_2) & d_i > l_{\max}, \Delta d_i < 0 \\ 0 & d_i \leq l_{\max} \end{cases} \tag{18}$$

where σ_1 and σ_2 are the reward coefficients that are used to adjust the reward size.

In the DE-MADDPG algorithm, it is necessary to design the local reward and the global reward. The global reward is given by

$$r_g = \begin{cases} R_2 & \text{if the network is totally connected} \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

where R_2 is the terminal reward of the entire cluster if the network is connected.

The local reward of an agent i is given by

$$r_l = \begin{cases} r_i + r_c & s_i{}' \notin \mathbb{S} \\ r_i & s_i{}' \in \mathbb{S} \end{cases} \tag{20}$$

where r_c is the punishment of UAV i if it is out of state space, and $\mathbb{S}$ is the set of state space.

5. Experiment and Results

In this section, we first present the simulation setting and the hyperparameters. Then, the configuration of the algorithm is presented, followed by the results, including the reward function and success rate.

5.1. Environment Settings

The detailed simulation parameters are given in Table 1.

Table 1. The parameters of the UAV cluster network.

Parameter	Description	Value
d	Space dimension	3
n	Number of UAVs	8, 10, 12
$v_{\max}$	Maximum velocity of UAVs	2 m/s
β	Flight path angle of UAVs	$[-\pi/2, \pi/2]$
ξ	Heading angle of UAVs	$[0, 2\pi]$
$l_{\max}$	Maximum communication distance	30 m

1. With the virtual navigator as the center, we considered the initialization of a space area of 100 m × 100 m × 100 m. The UAV clusters were randomly distributed in this space area, and the space area moved with the movement of the virtual navigator. UAVs were homogeneous with the same system parameters.

2. To facilitate the calculation, we simplified the action space. By default, a UAV moved at the maximum speed when adjusting its relative position with an elevation angle of $\beta \in \{-\pi/2, 0, \pi/2\}$ and heading angle of $\xi \in \{0, \pi/2, \pi, 3\pi/2\}$.

3. We considered two environments for training and testing. In the training environment, in cases where the UAV was about to move beyond the boundary, a rebound occurred with a certain penalty but without energy consumption. In the test environment, if the UAV hit the boundary, the episode would fail.

5.2. Training Configuration

The simulations were developed in Python, using Pycharm Community 2020.2 and the Anaconda3 platform. The deep neural network was also based on the Paddlepaddle1.8.5 framework.

The UAV obtained its state by observing the environment and produced the control values according to the set control policy. The UAV then used the feedback of the environment to adjust the control policy and form a closed-loop training process. The training hyperparameters are presented in Table 2.

Table 2. The hyperparameters.

Parameter	Description	Value/Interval
γ	Discount factor	0.99
τ	Soft update rate	0.1
$BatchSize$	Number of episodes to optimize	1024
M	Size of experience pool size	200,000
α_c	Learning rate of the critic model	0.001
α_a	Learning rate of the actor model	0.001
$MaxEpisode$	Number of episodes	20,000
$MaxStep$	Maximum steps per episode	200

The network structure of the DE-MADDPG is shown in Figure 5. The local critic network of each agent is a three-layer fully connected neural network, including an input layer, a hidden layer, and an output layer. The input corresponds to the observation and action information of this agent. The output is a scalar that evaluates the action in the state of this agent. The hidden layer includes 64 neurons with Rectified Linear Unit (ReLU) activation function. The output layer is linear without an activation function.

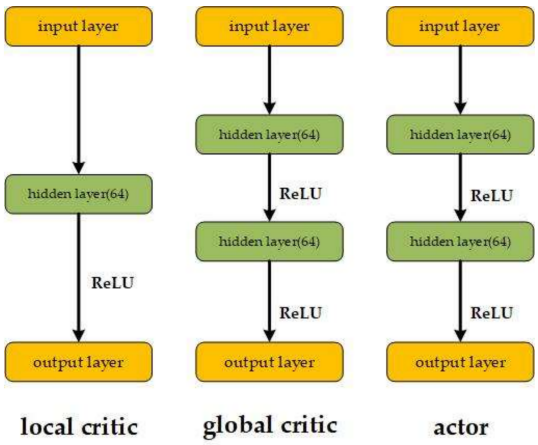

Figure 5. The network structure of the DE-MADDPG.

The global centralized critic network is a four-layer fully connected neural network, including two hidden layers. The input is the information of observations and actions of all agents. The output is a scalar that evaluates the current actions in the current global state.

The actor network is a four-layer fully connected neural network—the same as the global centralized critic network. The input corresponds to the observation information of this agent. The output is the value of all actions in the current state.

5.3. Simulation Results

In Figure 6, we present the total reward curve of the DE-MADDPG for the training of 20,000 episodes. The green line is the reward for each episode, and the red line is the average reward for every 50 episodes. During the first 1000 episodes, the agents gradually learn the policy of moving toward the center and obtain higher intermediate rewards. The agents gradually learn the policy of building a connected network, and the reward reaches a stable value after 17,000 episodes. Note that there are very few episodes that did not achieve full connectivity within 200 steps.

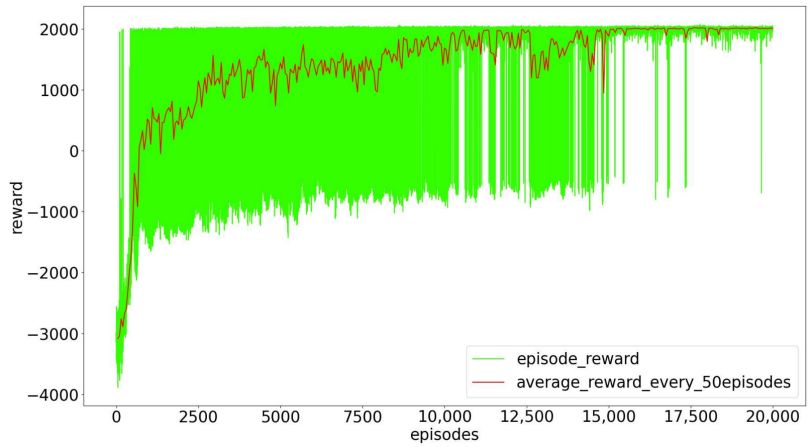

Figure 6. Rewards versus 20,000 episodes.

The training results after different numbers of episodes are shown intuitively in Figure 7. In the subgraphs (a–d), lines with different colors indicate the tracks of different UAVs that built the connected network, and the dot on each line indicates the position of the UAV in the cluster coordinate system after the network was connected. After 500 episodes, the tracks were very messy and redundant, and the UAVs were more likely to collide. The agents learned the policy of moving toward the center after 1000 episodes. However, the turbulent tracks show that the UAVs changed their direction of movement frequently. After 3000 episodes, the tracks were clearer, but the direction of movement of some UAVs changed too many times. After 10,000 episodes, the UAV rarely changed their direction of movement, and the change of direction was relatively smooth. We ran 100 test episodes after every 500 training episodes, and the average steps of successful testing episodes after different numbers of episodes are shown in Figure 8. This illustrates that the cluster built a connected network faster and more directly with an increasing number of training episodes.

In Figure 9, we present the average reward for every 50 episodes and the standard deviation for 10 training sessions of the MADDPG and DE-MADDPG. In each training, the same random seed was used to ensure the same initial state in each training round. Throughout the training phase, the reward of the DE-MADDPG was higher than that of the MADDPG. This reason can be easily found in Figure 10, where we see that the MADDPG was not able to learn the strategy of controlling the UAV not to exceed the state space. Figure 10 also illustrates the rate of exceeding state space of the MADDPG and DE-MADDPG for different numbers of training episodes. Comparing the training results of MADDPG and DE-MADDPG, the policy that was learned by DE-MADDPG was more efficient.

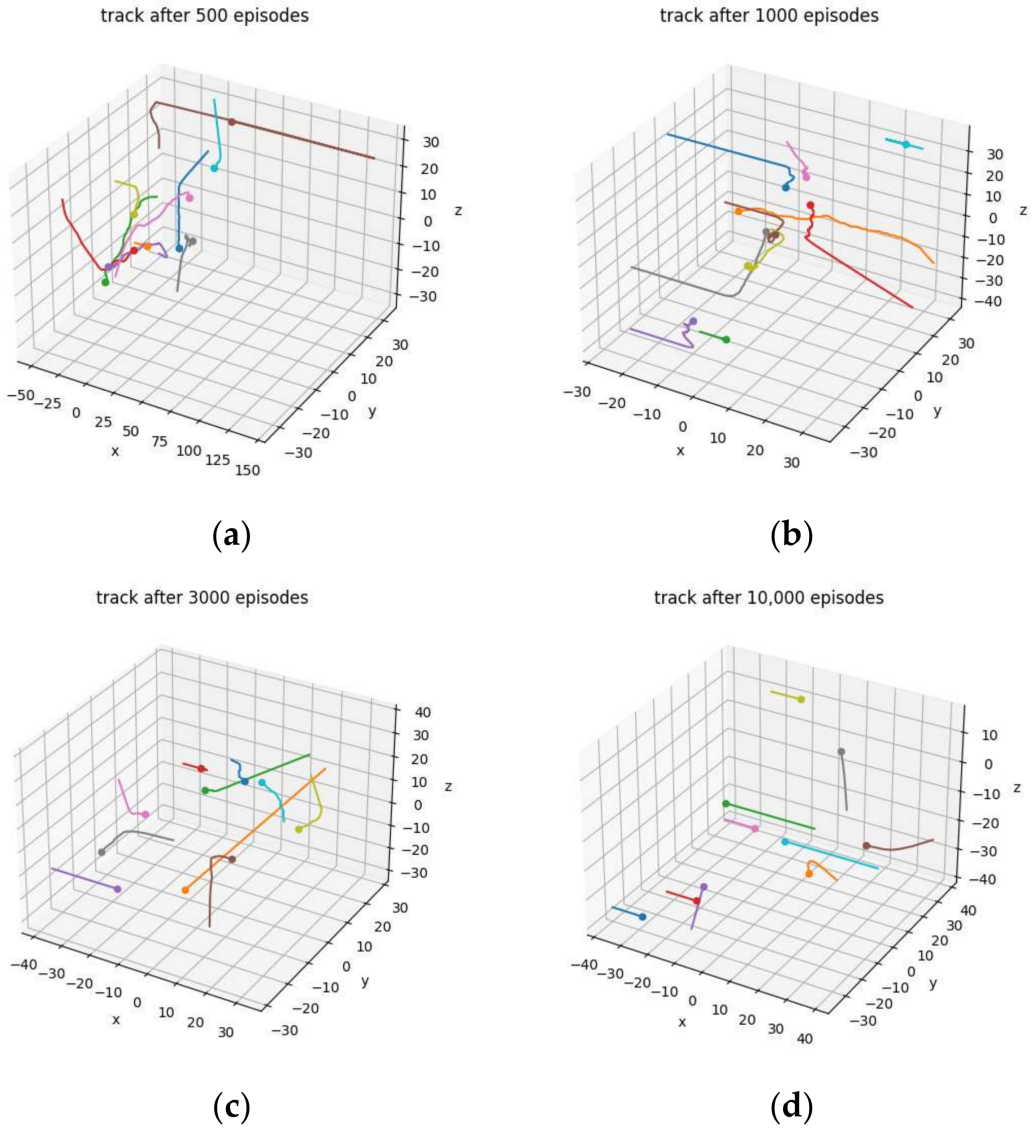

Figure 7. Tracks of the cluster after different numbers of episodes.

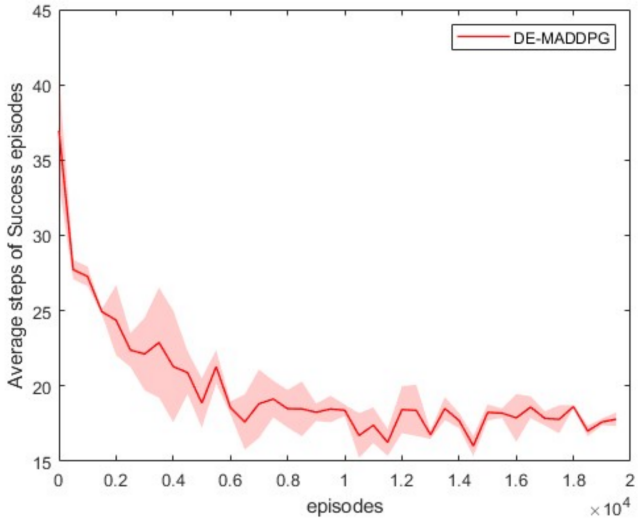

Figure 8. Average steps after different numbers of episodes.

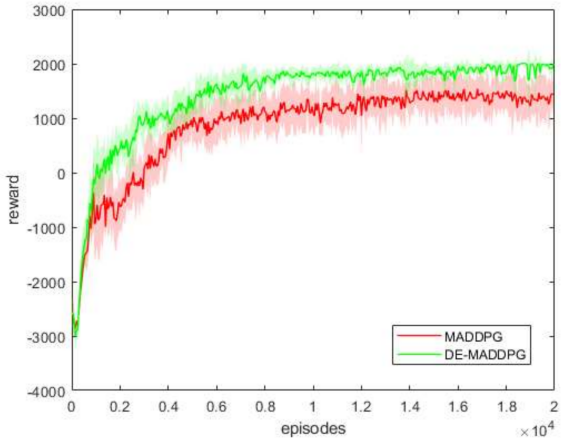

Figure 9. The average rewards versus the episodes for 10 iterations of training.

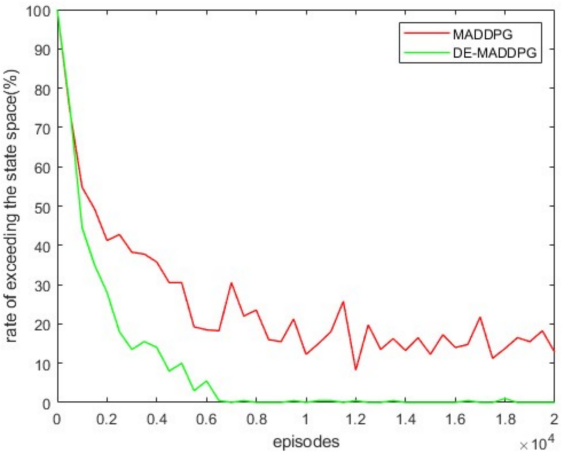

Figure 10. The rate of exceeding the state space.

Figure 11 presents the success rate of the MADDPG and DE-MADDPG for different numbers of training episodes. We conducted 100 tests with 100 episodes of training. During the tests, the agents did not update their policy. Comparing the test results of the MADDPG and DE-MADDPG, the success rate of the DE-MADDPG was higher, indicating that the policy of the DE-MADDPG was better than that of the MADDPG.

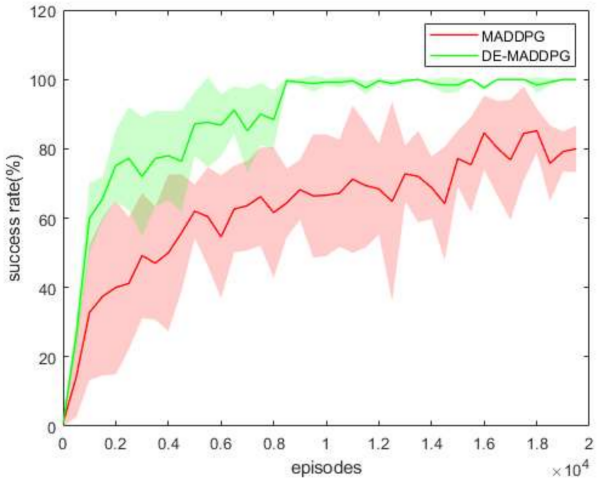

Figure 11. Success rate versus episodes.

Figure 12 illustrates the success rate of clusters with different numbers of UAVs after training. It is seen that by increasing the number of UAVs, the training becomes more difficult to converge. Nevertheless, after training enough episodes, all of them eventually learn excellent policies for constructing a connected network.

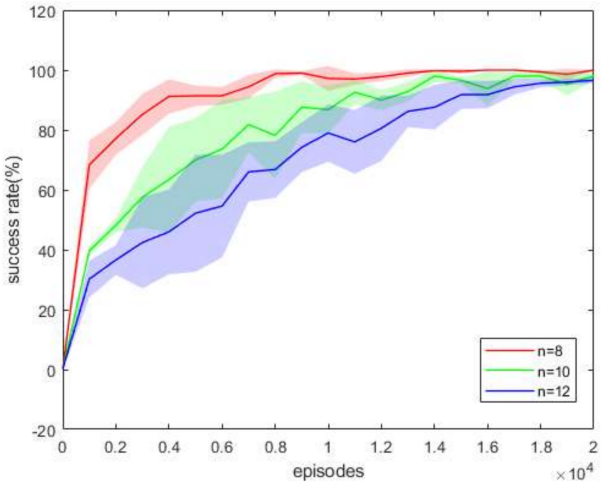

Figure 12. Success rate with different amounts of UAVs.

6. Conclusions

We developed the DE-MADDPG algorithm for UAV clusters rebuilding a connected network, where a virtual navigator was proposed to address the instability issue of the state space. The simulation results confirmed the effectiveness of the proposed algorithm for controlling the UAV clusters and constructing a connected network, where the success rate was much higher than that of the MADDPG algorithm. The results presented in this paper fill the gap in the application of reinforcement learning algorithms to providing cluster network connectivity.

The communication network in our study is a single-connected network, which is easily paralyzed by external interference. Fortunately, a bi-connected network, which is defined as two disjointed communication paths between any pair of UAV nodes in the network, is more robust. In the future, we will study how to build a bi-connected network for UAV clusters based on RL.

Author Contributions: Conceptualization, L.C.; methodology, Z.Z. and K.Z.; software, Z.Z.; validation, N.X. and Z.Z.; data curation, Z.Z.; writing—original draft preparation, Z.Z.; writing—review and editing, N.X., K.Z. and Z.Z.; visualization, Z.Z. and N.X.; supervision, L.C.; funding acquisition, L.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Zhu, J.; Song, B.; Zhang, H.; Wang, C. Positioning accuracy and formation analysis of multi-UAV passive positioning system. *J. Phys. Conf. Ser.* **2021**, *1856*, 012055. [CrossRef]
2. Bacco, M.; Cassarà, P.; Colucci, M.; Gotta, A.; Marchese, M.; Patrone, F.A.; Marchese, M.; Patrone, F. A Survey on Network Architectures and Applications for Nanosat and UAV Swarms. In *Wireless and Satellite Systems, Proceedings of the 9th International Conference on Wireless & Satellite Systems, Oxford, UK, 14–15 September 2017*; Springer Nature Switzerland AG: Cham, Switzerland, 2018; pp. 75–85.

3. Bacco, M.; Chessa, S.; Benedetto, M.; Fabbri, D.; Girolami, M.; Gotta, A.; Moroni, D.; Pascali, M.A.; Pellegrini, V. *UAVs and UAV Swarms for Civilian Applications: Communications and Image Processing in the SCIADRO Project*; Springer: Cham, Switzerland, 2017.
4. Dong, L.; Tong, Z.; Tong, M.; Tang, S. Boundary exploration algorithm of disaster environment of coal mine based on multi-UAVs. In Proceedings of the 2017 7th IEEE International Conference on Electronics Information and Emergency Communication (ICEIEC), Macau, China, 21–23 July 2017.
5. Ke, R.; Li, Z.; Kim, S.; Ash, J.; Cui, Z.; Wang, Y. Real-Time Bidirectional Traffic Flow Parameter Estimation From Aerial Videos. *IEEE Trans. Intell. Transp. Syst.* **2016**, *18*, 890–901. [CrossRef]
6. Dui, H.; Zhang, C.; Bai, G.; Chen, L. Mission reliability modeling of UAV swarm and its structure optimization based on importance measure. *Reliab. Eng. Syst. Saf.* **2021**, *215*, 107879. [CrossRef]
7. Lakew, D.S.; Sa'Ad, U.; Dao, N.-N.; Na, W.; Cho, S. Routing in Flying Ad Hoc Networks: A Comprehensive Survey. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 1071–1120. [CrossRef]
8. Wu, M.; Gao, Y.; Wang, P.; Zhang, F.; Liu, Z. The Multi-Dimensional Actions Control Approach for Obstacle Avoidance Based on Reinforcement Learning. *Symmetry* **2021**, *13*, 1335. [CrossRef]
9. Wang, L.; Wang, K.; Pan, C.; Xu, W.; Hanzo, L. Multi-Agent Deep Reinforcement Learning Based Trajectory Planning for Multi-UAV Assisted Mobile Edge Computing. *IEEE Trans. Cogn. Commun. Netw.* **2020**, *7*, 73–84. [CrossRef]
10. Zhang, Y.; Zhuang, Z.; Gao, F.; Wang, J.; Han, Z. Multi-Agent Deep Reinforcement Learning for Secure UAV Communications. In Proceedings of the 2020 IEEE Wireless Communications and Networking Conference (WCNC), Seoul, Korea, 25–28 May 2020.
11. Hu, C. A Confrontation Decision-Making Method with Deep Reinforcement Learning and Knowledge Transfer for Multi-Agent System. *Symmetry* **2020**, *12*, 631. [CrossRef]
12. Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In Proceedings of the 31st Annual Conference on Neural Information Processing Systems, NIPS 2017, Long Beach, CA, USA, 4–9 December 2017; pp. 6380–6391.
13. Ramanathan, R.; Rosales-Hain, R. Topology control of multihop wireless networks using transmit power adjustment. In Proceedings of the IEEE INFOCOM 2000, Conference on Computer Communications, Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064), Tel Aviv, Israel, 26–30 March 2000.
14. Kashyap, A.; Khuller, S.; Shayman, M. Relay placement for fault tolerance in wireless networks in higher dimensions. *Comput. Geom. Theory Appl.* **2011**, *44*, 206–215. [CrossRef]
15. Reynolds, C.W. Flocks, Herds and Schools: A Distributed Behavioral Model. In Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques; Association for Computing Machinery: New York, NY, USA, 1987; pp. 25–34.
16. Vicsek, T.; Czirok, A.; Ben-Jacob, E.; Cohen, I.; Sochet, O. Novel Type of Phase Transition in a System of Self-Driven Particles. *Phys. Rev. Lett.* **1995**, *75*, 1226. [CrossRef] [PubMed]
17. Couzin, I.D.; Krause, J.; James, R.; Ruxton, G.D.; Franks, N.R. Collective memory and spatial sorting in animal groups. *J. Theor. Biol.* **2002**, *218*, 1–11. [CrossRef]
18. Yang, P.; Freeman, R.A.; Gordon, G.J.; Lynch, K.M.; Srinivasa, S.S. Decentralized estimation and control of graph connectivity for mobile sensor networks. *Automatica* **2010**, *46*, 390–396. [CrossRef]
19. Hao, F.; Yue, W.; Jie, C.; Xin, B. Flocking of Second-Order Multiagent Systems With Connectivity Preservation Based on Algebraic Connectivity Estimation. *IEEE Trans. Cybern.* **2017**, *47*, 1067–1077.
20. Dai, R.; Maximoff, J.; Mesbahi, M. Optimal Trajectory Generation for Establishing Connectivity in Proximity Networks. *IEEE Trans. Aerosp. Electron. Syst.* **2013**, *49*, 1968–1981. [CrossRef]
21. Zanol, R.; Chiariotti, F.; Zanella, A. Drone mapping through multi-agent reinforcement learning. In Proceedings of the 2019 IEEE Wireless Communications and Networking Conference (WCNC), Marrakesh, Morocco, 15–18 April 2019.
22. Klaine, P.V.; Nadas, J.P.B.; Souza, R.D.; Imran, M.A. Distributed Drone Base Station Positioning for Emergency Cellular Networks Using Reinforcement Learning. *Cogn. Comput.* **2018**, *10*, 790–804. [CrossRef] [PubMed]
23. Liu, C.H.; Chen, Z.; Tang, J.; Xu, J.; Piao, C. Energy-Efficient UAV Control for Effective and Fair Communication Coverage: A Deep Reinforcement Learning Approach. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 2059–2070. [CrossRef]
24. Han, Q.; Shi, D.; Shen, T.; Xu, X.; Li, Y.; Wang, L. Joint Optimization of Multi-UAV Target Assignment and Path Planning Based on Multi-Agent Reinforcement Learning. *IEEE Access* **2019**, *7*, 146264–146272.
25. Guo, Q.; Yan, J.; Xu, W. Localized Fault Tolerant Algorithm Based on Node Movement Freedom Degree in Flying Ad Hoc Networks. *Symmetry* **2019**, *11*, 106. [CrossRef]
26. Zavlanos, M.M.; Pappas, G.J. Controlling Connectivity of Dynamic Graphs. In Proceedings of the 44th IEEE Conference on Decision and Control, Seville, Spain, 15 December 2005.
27. Yang, Y.; Wang, J. An Overview of Multi-Agent Reinforcement Learning from Game Theoretical Perspective. *arXiv* **2020**, arXiv:2011.00583.
28. Pfau, D.; Vinyals, O. Connecting Generative Adversarial Networks and Actor-Critic Methods. *arXiv* **2016**, arXiv:1610.01945.
29. Littman, M.L. Markov games as a framework for multi-agent reinforcement learning. In Proceedings of the Eleventh International Conference on International Conference on Machine Learning, New Brunswick, NJ, USA, 10–13 July 1994; pp. 157–163.
30. Gronauer, S.; Diepold, K. Multi-agent deep reinforcement learning: A survey. *Artif. Intell. Rev.* **2021**. [CrossRef]
31. Xu, X.; Li, R.; Zhao, Z.; Zhang, H. Stigmergic Independent Reinforcement Learning for Multi-Agent Collaboration. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, 1–15. [CrossRef]

32. Gupta, J.K.; Egorov, M.; Kochenderfer, M. Cooperative Multi-agent Control Using Deep Reinforcement Learning. In *Autonomous Agents and Multiagent Systems, Proceedings of the International Conference on Autonomous Agents and Multiagent Systems, São Paulo, Brazil, 8–12 May 2017*; Springer: Cham, Switzerland, 2017; pp. 66–83.
33. Sheikh, H.U.; Blni, L. Multi-Agent Reinforcement Learning for Problems with Combined Individual and Team Reward. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020.
34. Wei, F.; Wang, H.; Xu, Z. Research on Cooperative Pursuit Strategy for Multi-UAVs based on DE-MADDPG Algorithm. *Acta Aeronautica Astronautica. Sin.* **2021**, 1–16. [CrossRef]