Multi-UAV Redeployment Optimization Based on Multi-Agent Deep Reinforcement Learning Oriented to Swarm Performance Restoration

Qilong Wu 1,†, Zitao Geng 1,†, Yi Ren 1, Qiang Feng 1,* and Jilong Zhong 2

1 School of Reliability and Systems Engineering, Beihang University, Beijing 100191, China; wql_rse@buaa.edu.cn (Q.W.); gengzitao@buaa.edu.cn (Z.G.); renyi@buaa.edu.cn (Y.R.)
2 Defense Innovation Institute, Academy of Military Science, Beijing 100071, China; z_jilong@sina.cn
* Correspondence: fengqiang@buaa.edu.cn
† These authors contributed equally to this work.

Abstract: Distributed artificial intelligence is increasingly being applied to multiple unmanned aerial vehicles (multi-UAVs). This poses challenges to the distributed reconfiguration (DR) required for the optimal redeployment of multi-UAVs in the event of vehicle destruction. This paper presents a multi-agent deep reinforcement learning-based DR strategy (DRS) that optimizes the multi-UAV group redeployment in terms of swarm performance. To generate a two-layer DRS between multiple groups and a single group, a multi-agent deep reinforcement learning framework is developed in which a QMIX network determines the swarm redeployment, and each deep Q-network determines the single-group redeployment. The proposed method is simulated using Python and a case study demonstrates its effectiveness as a high-quality DRS for large-scale scenarios.

Keywords: distributed reconfiguration strategy; multi-agent deep reinforcement learning; unmanned aerial vehicle (UAV); UAV swarm redeployment

1. Introduction

Recently, mission planning associated with unmanned aerial vehicles (UAVs) has received considerable attention [1,2], and distributed artificial intelligence (AI) technologies have been extensively applied in multiple-UAV (multi-UAV) mission planning, enabling efficient decision-making and yielding high-quality solutions [3,4]. For missions in geographically decentralized environments, the focus is on deploying UAVs to their destinations and repositioning them to adapt to changing circumstances [5]. To minimize the costs of positioning UAVs, Masroor et al. [6] proposed a branch-and-bound algorithm that determines the optimal UAV deployment solution in emergency situations. Savkin et al. [7] employed a range-based reactive algorithm for autonomous UAV deployment. Nevertheless, many existing distributed algorithms lack the security necessary to achieve the global objective.

For the UAVs in a swarm, the placement of individual is important, but the completion of the swarm mission is the ultimate goal. Wang et al. [8] proposed a K-means clustering-based UAV deployment scheme that significantly improves the spectrum efficiency and energy efficiency of cellular uplinks at limited cost, while Yu et al. [9] introduced an evolutionary game-based adaptive dynamic reconfiguration mechanism that provides decision support for the cooperative mode design of unmanned swarm operations. These algorithms take static multi-swarm problems into account. However, some of the UAVs may suffer destruction or break down during a mission [10]. To deal with situations in which the swarm suffers unexpected destruction, adaptive swarm reconfiguration strategies are required [11].
Learning-based methods are gaining increasing attention for their flexibility and efficiency [12,13]. Deep reinforcement learning (DRL) has shown promising results in resolving the task assignment problems associated with multi-UAV swarms [14]. Samir et al. [15] combined DRL with joint optimization to achieve improved learning efficiency, although changes to the dynamic environment can hinder the implementation of this strategy. Zhang et al. [16] investigated a double deep Q-network (DQN) framework for long period UAV swarm collaborative tasks and designed a guided reward function to solve the convergence problem caused by the sparse returns of long period tasks. Huda et al. [17] investigated a surveillance application scenario using a hierarchical UAV swarm. In this case, they used a DQN to minimize the weighted sum cost. As a result, their DRL method exhibited better convergence and effectiveness than traditional methods. Zhang et al. [18] designed a DRL-based algorithm to find the optimal attack sequence for large-scale UAV swarm so that the purpose of destroying the target communication system can be achieved. Mou et al. [19] built a geometric way to project the 3D terrain surface into many weighted 2D patches and proposed a swarm DQN reinforcement learning algorithm to select patches for leader UAVs, which could cover the object area with little redundancies. Liu et al. [20] focused on a latency minimization problem for both communication and computation in a maritime UAV swarm mobile edge computing network, then they proposed a DQN and a deep deterministic policy gradient algorithm to optimize the trajectory of multi-UAVs and configuration of virtual machines. However, multi-agent DRL (MADRL) captures real-world situations more easily than DRL [21,22]. Hence, MADRL is considered an important topic of research. Xia et al. [22] proposed an end-to-end cooperative multi-agent reinforcement learning scheme that enables the UAV swarm to make decisions on the basis of the past and current states of the target. Lv et al. [23] proposed a MADRL-based UAV swarm communication scheme to optimize the relay selection and power allocation, then they designed a DRL-based scheme to improve the anti-jamming performance. Xiang et al. [24] established an intelligent UAV swarm model based on a multi-agent deep deterministic policy gradient algorithm, significantly improving the success rate of the UAV swarm in confrontations.

In summary, developments in distributed AI mean that swarm intelligence is now of vital strategic importance. Under this background, it is vital to develop multi-agent algorithms. However, few reconfiguration studies have investigated this distributed multi-agent scenario. Therefore, this paper proposes a MADRL-based distributed reconfiguration strategy (DRS) for the problem of UAV swarm reconfiguration after large-scale destruction. The main contributions of this paper are as follows:

1. UAV swarm reconfiguration is formulated so as to generate a swarm DRS considering detection missions and destruction. The finite number of UAVs is the constraint, and the coverage area forms the objective.

2. MADRL-based swarm reconfiguration employs multi-agent deep learning and the Q-MIX network. Each agent, representing a group, uses reinforcement learning to select the optimal distributed reconfiguration (DR) actions. The Q-MIX network is used to synthesize the actions of each agent and output the final strategy.

3. When the network has been trained well, the algorithm can effectively utilize various UAV swarm information to support DR decision-making. This enables efficient and steady multi-group swarm DR to achieve the mission objective.

The remainder of this paper is organized as follows. Section 2 presents the swarm mission framework. Section 3 elucidates the DRS, before Section 4 introduces a UAV swarm reconfiguration case study of detection missions. Finally, Section 5 presents the concluding remarks.
2. Problem Formulation

2.1. Mission, Destruction, and Reconfiguration

2.1.1. Mission

A detection mission containing $M$ irregular detection areas is considered. As shown in Figure 1a, the detection areas (colored yellow) are divided into hexagons, which are inscribed hexagons of the mission areas (colored green).

![Figure 1. Mission and UAV swarm. (a) Detection and mission areas. (b) UAV swarm detection.](image)

The swarm detection mission area set can then be expressed as follows:

$$D = \{MA_1, MA_2, ..., MA_m, ..., MA_M\}$$

(1)

where each group mission area $MA_m$, $m \in \{1, 2, ..., M\}$, is covered by a certain number of hexagons, as follows:

$$MA_m = \{ma_{m1}, ma_{m2}, ..., ma_{mn}, ..., ma_{mN_m}\}$$

(2)

where $N_m$ is the total number of hexagons in group mission area $MA_m$, and each hexagon represents a single UAV mission area $ma_{mn}$, $m \in \{1, 2, ..., M\}$, $n \in \{1, 2, ..., N_m\}$.

A UAV swarm, the size of which is determined by the detection area, is dispatched for a detection mission. Each area requires a group to execute the detection mission, and the number of UAVs in the group depends on the number of hexagons in the mission area. Furthermore, each group is formed of one leader UAV and several follower UAVs. To execute a detection mission, as shown in Figure 1b, the radius $R$ of the UAV detection area is determined by the detection equipment installed on the UAVs.

The UAV swarm can then be expressed as follows:

$$Swarm = \{G_1, G_2, ..., G_m, ..., G_M\}$$

(3)

where each group $G_m$, $m \in \{1, 2, ..., M\}$, performs detection in the group mission area $MA_m$, as follows:

$$G_m = \{U_{m1}, U_{m2}, ..., U_{mn}, ..., U_{mN_m}\}$$

(4)
where $U_{mn}$ is the $m$-th UAV in group $n$ and performs detection in UAV mission area $ma_{mn}$, $m \in \{1, 2, ..., M\}$, $n \in \{1, 2, ..., N_m\}$. The first UAV in each group is the leader of that group.

2.1.2. Destruction

The UAV swarm may be subject to local and random destruction, and some UAVs may be destroyed. The effects of this destruction are used as inputs. Each UAV has two states: normal working and complete failure. When a UAV suffers destruction, it enters the failure state. When a leader UAV is destroyed, a follower UAV in the same group assumes the role of the leader of that particular group.

The scope of local destruction is represented by a circle with center coordinates $(i_d, j_d)$ and radius $r_d$, as illustrated in Figure 2a. The values of $(i_d, j_d)$ and $r_d$ are randomly generated.

Random destruction is characterized by a destruction scale, denoted as $S_{rand}$, which is also generated randomly. When random destruction occurs, $S_{rand}$ random UAVs transition from the normal state to the faulty state, as depicted in Figure 2b.

2.1.3. Reconfiguration

UAV swarm reconfiguration is an autonomous behavior that adapts to changes in the environment to enable the execution of the task. When the swarm is affected by dynamic changes during task execution, the system can use a DRS to achieve global mission performance recovery and reconfiguration, thus ensuring mission continuity.

When destruction occurs, the state of the UAV swarm is input into the reconstruction algorithm. The resulting strategy is communicated back to each UAV group. In-group reconstruction and inter-group reconstruction are applied to certain UAVs, as shown in Figure 3a. After the reconstruction is completed, all mission areas should be covered by the detection range of the UAVs, as shown in Figure 3b.
2.2. Objective, Constraints, and Variables

Over a finite time $\tau_{thr}$, swarm reconfiguration aims to maximize the total coverage area (TCA) $\epsilon_{tot}$, which is the mission area detected by the UAVs. This can be expressed as follows:

$$\epsilon_{tot}(\tau) = \sum_{m=1}^{M} \sum_{n=1}^{N_m} \epsilon_{mn}$$  \hspace{1cm} (5)

where $\epsilon_{tot}(\tau)$ is the TCA at the current time $\tau$, $\epsilon_{mn}$ is the detected area of mission area $ma_{mn}$ if $ma_{mn}$ is not covered, $\epsilon_{mn} = 0$.

The problem should be solved at the swarm level. Considering the number of remaining UAVs, the number of UAVs to be repositioned should be less than the number of normal working UAVs. Furthermore, the minimum area detected by the UAVs in each mission area must be set. Therefore, the reconfiguration problem can be expressed as follows:

$$\begin{align*}
\text{Max } \epsilon_{tot} \\
\text{s. t. } & \epsilon_m \geq \epsilon_{min} \\
& N^m_{\text{move}} \leq N^m_{\text{normal}} \\
& d \geq d_{\text{min}}
\end{align*}$$  \hspace{1cm} (6)

where $\epsilon_m$ is the coverage area of group $G_m$, $\epsilon_{min}$ is the specified minimum coverage area for group $G_m$, $N^m_{\text{move}}$ is the number of UAVs in $G_m$ that can be repositioned, $N^m_{\text{normal}}$ is the number of normal-working UAVs in $G_m$, $d$ is the distance between two normal-working UAVs, and $d_{\text{min}}$ is the minimum allowable distance, which is the safety distance between UAVs. This problem considers the UAVs within the communication range. If a UAV exceeds the communication distance, it enters the faulty state due to communication failure.
The initial deployment status depends on whether there is a normal-working UAV at a certain hexagon for each group mission area $MA_m$. Then, the UAV swarm distribution deployment status can be represented by a $I \times J$ matrix $S$. The matrix element $s_{ij} = 1$ if there is a normal-working UAV $U_{mn}$ in hexagon $H_{mi}$ and $s_{ij} = 0$ if not. Therefore, the deployment status information of the UAV swarm can be expressed as follows:

$$S = \begin{bmatrix}
    s_{11} & s_{12} & \cdots & s_{1J} \\
    s_{21} & s_{22} & \cdots & s_{2J} \\
    \vdots & \vdots & \ddots & \vdots \\
    s_{(I-1)1} & s_{(I-1)2} & \cdots & s_{(I-1)J}
\end{bmatrix}$$

(7)

3. MADRL-Based DR Method

An MADRL framework was developed to solve the DR problem described in the previous section, as shown in Figure 4. The framework consists of three parts: a reconfiguration decision-making progress, agent decision-making, and a neural network. The three parts of the framework are described in this section, and the reconfiguration decision-making process is illustrated in Figure 4A.

![Figure 4. Multi-agent deep reinforcement learning framework.](image)

3.1. Reconfiguration Decision Process
The group agents choose the DRAs for the UAV groups. The UAV swarm’s status matrix $S_t$ is used by the group agents as the main input. This process can be expressed as follows:

$$f_{\text{agent-}m}(S_t, M_t^{m-1}) = \text{mov}_m^t[[S_t, M_t^{m-1}]]$$

(8)

where $S_t$ represents the current state matrix of time step $t$, and $M_t^{m-1}$ represents the movement feature set of agent $m$ which consists of this agent’s history of movement features $\{\text{mov}_t^{m-1}, \text{mov}_t^{m-2}, \text{mov}_t^{m-3}\}$. History of movement features are necessary, because the agents are not fully observable solely from the current state, since the DR decision-making is a sequential decision process. The movement feature of agent $m$ can be described as

$$\text{mov}_t^m = [\text{loc}_t^{\text{init}}, \text{loc}_t^{\text{final}}]$$

where both $\text{loc}_t^{\text{init}}$ and $\text{loc}_t^{\text{final}}$ are the location matrices to describe the hexagons in the figures of Section 2. Each element in the location matrix relates to a hexagon, and if the element is 1, the related hexagon is the chosen location. Both $\text{loc}_t^{\text{init}}$ and $\text{loc}_t^{\text{final}}$ have one element which is equal to 1, and the other elements are equal to 0. The location matrix $\text{loc}_t^{\text{init}}$ represents the initial location of the movement feature set of agent $m$ at time step $t$. A swarm agent uses a QMIX network to combine the outputs of all group agents and choose the most efficient one. This can be expressed as follows:

$$f_{\text{agent-\text{qmix}}}(\text{mov}_t^{m=1:M}, S_t, M_{t-1}) = \text{mov}_t[[S_t, M_{t-1}]]$$

(9)

where $\text{mov}_t^{m=1:M} = \{\text{mov}_1^t, \text{mov}_2^t, ..., \text{mov}_M^t\}$ represents the movement feature set of all group agents at time step $t$, $M_{t-1}$ represents the last swarm movement feature set which consists of this swarm’s history of movement features $\{\text{mov}_{t-1}, \text{mov}_{t-2}, \text{mov}_{t-3}\}$, and the output $\text{mov}_t[[S_t, M_t]]$ represents the final chosen movement feature for the swarm.

The DR process consists of mission and destruction features, DR action generation, and renewal features. These three components are described in the following subsections.

### 3.1.1. Mission and Destruction Features

The destruction is randomly initialized at time $t_d$, and the status matrix $S_t$ is then generated. The coverage area at this time is $\varepsilon_a(t_d) = \varepsilon_d$.

To reconfigure the swarm and reach the maximum coverage rate, $M$ agents, representing $M$ different UAV groups, execute a sequence of DR actions. The DR action set is described as follows:

$$\Phi = \{\text{act}_{tnn}\}_{t=1}^T, m\in\{1,2,...,M\}, n\in\{1,2,...,N-m\}$$

(10)

where the $\text{act}_{tnn}$ is the DR action of $UAV_{mn}$ at time step $t$. This DR action is defined as $\text{act}_{tnn} = [\text{cen}(H_{ij}), \text{cen}(H_{ji})]$ which means that the $UAV_{mn}$ in hexagon $H_{ij}$ moves to the target hexagon $H_{ji}$. The parameter $\text{cen}(H_{ij})$ represents the center location of hexagon $H_{ij}$, and the action $\text{act}_{tnn}$ is generated according to the movement feature $\text{mov}_t$. The DR action set of group $m$ can be described as follows:

$$\Phi_m = \{\text{act}_{tnn}\}_{n=1}^{N_m}, m\in\{1,2,...,M\}, t\in\{1,2,...,T\}$$

(11)

After the DR action has finished, agent $m$ uses a search algorithm to select the next DR action $\text{act}_{mnlt}$ for $UAV_{mn}$ in group $G_m$, or chooses to finish the reconfiguration process. This process is repeated in each time step $t$. The neural network of agent $m$ (see Section 3.2) can be described as follows:

$$Q_m(S_t, \text{mov}_t^m) = f_{\text{QNN-}m}(S_t, M_t^{m})$$

(12)

where $Q_m(S_t, \text{mov}_t^m)$ is the value of the movement feature $\text{mov}_t^m$ at time step $t$.

Each time step corresponds to a realistic period of time, the length of which is proportional to the distance the UAV moves in this time step.

### 3.1.2. Reconfiguration Action Generation

For the DR action $\text{act}_{tnn}$, once complete, the moving UAV is considered to perform the detection mission at the new location, then the status matrix $S_t$ can be updated. The term $\varepsilon_a(t)$ can be calculated according to (5) after the movement. The objective of agent
m is to achieve the maximum coverage area as efficiently as possible. Thus, the reward should include both the coverage area and reconfiguration time. All agents use the same reward function, and the reward at time step t is defined as follows:

\[ R_t = \sum_{t=0}^{T-t} \delta^\zeta \int_{t+\zeta}^{t+\zeta-1} (1 - \frac{\varepsilon_{tot}(\tau)}{\varepsilon_0}) d\tau \]

where \( R_t \) is the reward at time step t, \( \tau + \zeta \) is the reconfiguration time of time step \((t + \zeta)\), \( \tau + \zeta - 1 \) is the reconfiguration time of time step \((t + \zeta - 1)\), \( \varepsilon_0 \) is the initial TCA, \( \delta \) is the discounted factor, and \( \tau_T \) is the time to finish reconfiguration (TTFR).

For agent m, an optimization algorithm is used to select the best movement feature of UAV group m. For each time step t, the DQN of agent m outputs a movement value quantity \( Q_m(S_t, mov^m_t) \), then this agent outputs a movement feature \( mov^m_t \). A QMIX network is used to select the most effective action from all possible actions.

The mixing network has two parts: a parameter generating network and an inference network. The former receives the global state \( S_t \) and generates the neuron weights and deviations. The latter receives the control quantity \( Q_m(S_t, mov^m_t) \) from each agent and generates the global utility value \( Q_{tot} \) with the help of the neuron weights and deviations.

The movement utility value \( Q_{tot} \) is used to formulate the final decision for the whole swarm (see Section 3.3), as expressed in (15).

### 3.1.3. Renewal Features

Once the swarm has finished act_{|mn}, the state matrix and feature set \([S_{tn},A_{tn}]\) is used as the new input to the algorithm. The algorithm continues to run and outputs new movement actions or takes the decision to end the reconfiguration process.

### 3.2. Deep Q-Learning for Reconfiguration

The agents use the deep Q-learning algorithm to evaluate the movement action, in which the action-value function is represented by a deep neural network parameterized by \( \theta \). The movement feature \( mov^m \) has a movement value function of \( Q_m(S_t, mov^m) \) as \( E_{S_{t+1:0},mov^m_{t+1:0}}[\sum R_t | S_t, mov^m_t] \), where \( \sum R_t = \sum_{i=0}^{\infty} \delta^i r_{t+i} \) is the discounted return and \( \delta \) is the discount factor.

The transition tuple of each movement action of the group agent m is stored as \([S, mov^m, R, S']\), where \( S \) is the state before \( mov \), \( mov \) is the selected mobile movement feature, \( R \) is the reward for this movement, and \( S' \) is the state after the movement has finished. \( \theta \) is learned by sampling batches of \( b \) transitions and minimizing the squared temporal-difference error:

\[ L(\theta) = \sum_{i=1}^{b} [Q^{DQN}_t(S, mov^m; \theta) - Q_m(S, mov^m; \theta)]^2 \]

where \( \gamma^{DQN} = R + \delta \max_{mov'} Q_m(S', mov'^m; \theta^-) \), \( \theta^- \) represents parameters of the target network that are periodically copied from \( \theta \) and held constant for several iterations, \( b \) is the batch size of transitions sampled from the replay buffer, \( Q_m(S, mov^m; \theta) \) is the utility value of \( mov^m \).

### 3.3. QMIX for Multi-Agent Strategy

The QMIX network is applied to the generated swarm-level DR action. The network represents \( Q_{tot} \) as a monotone function for mixing the individual value functions \( Q(S_t, mov_t) \) of each agent. This can be expressed as follows:

\[ Q_{tot}(S, mov) = f_{qmix}(Q_m(S, mov^m)_{m=1}^M) \]

where \( Q_m(S, mov^m)_{m=1}^M \) is the movement value set, and the \( Q_{tot}(S, mov) \) is a joint movement value of the swarm. The monotonicity of (15) can be enforced by the partial derivative relation \( \frac{\partial Q_{tot}}{\partial Q_m} \geq 0, m \in [1, M] \). To ensure this relationship, QMIX consists of agent networks, a mixing network, and a set of hypernetworks, as shown in Figure 4C.
For each agent \( m \), there is one agent network representing the individual value function \( Q_m(S, mov^m) \). The agent networks are represented as deep recurrent Q-networks (DRQNs). At each time step, the DRQNs receive the status \( S_t \) and last movement \( mov_t \) as input and output a value function \( Q_m(S, mov^m) \) to the mixing network.

The mixing network is a feedforward neural network that monotonically mixes all \( Q_m(S, mov^m) \) with nonnegative weights. The weights of the mixing network are generated by separate hypernetworks, each of which generates the weight of one layer using the status \( S_t \). The biases of the mixing network are produced in the same manner but are not necessarily nonnegative. The final bias is produced by a two-layer hypernetwork.

The whole QMIX network is trained end-to-end to minimize the following loss:

\[
L(\theta) = \sum_{i=1}^{b} [(Q^{DQN}_i - Q_{tot}(S, mov; \theta))^2]
\]

where \( \gamma_{DQN} = R + \delta_{max_{mov'}}Q_{tot}(S', mov'; \theta^-) \), \( Q_{tot}(S, mov; \theta) \) is the globe utility value of \( mov \).

4. Case Study

A case study of UAV swarm reconfiguration was simulated using Python. The numerical simulation is described from the perspective of optimal UAV swarm reconfiguration. The effectiveness of the proposed DR decision-making method is validated using the reconfiguration results under different scenarios. In this section, a fixed-wing UAV swarm is considered, although the proposed method is also applicable to other types of UAV swarms.

4.1. UAV Swarm Reconfiguration

4.1.1. Mission

A detection mission containing seven irregular detection areas is randomly generated, as shown in Figure 5. The yellow areas represent the detection areas, and the map is divided into hexagons. The hexagons of the mission areas (colored green) need to cover the detection areas. A UAV swarm with \( 7 \times 6 \) UAVs is simulated to execute this detection mission; the initial location information of each UAV is presented in Table 1.

![Mission Area Detection Area UAV](image)

**Figure 5.** Mission and UAV swarm deployment.

<table>
<thead>
<tr>
<th>( G_1 )</th>
<th>( UAV_{1,1} )</th>
<th>( UAV_{1,2} )</th>
<th>( UAV_{1,3} )</th>
<th>( UAV_{1,4} )</th>
<th>( UAV_{1,5} )</th>
<th>( UAV_{1,6} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>(3, 74.48)</td>
<td>(3, 77.94)</td>
<td>(3, 81.41)</td>
<td>(6, 76.21)</td>
<td>(6, 79.67)</td>
<td>(9, 77.94)</td>
</tr>
<tr>
<td>( G_2 )</td>
<td>( UAV_{2,1} )</td>
<td>( UAV_{2,2} )</td>
<td>( UAV_{2,3} )</td>
<td>( UAV_{2,4} )</td>
<td>( UAV_{2,5} )</td>
<td>( UAV_{2,6} )</td>
</tr>
<tr>
<td>Location</td>
<td>(12, 76.21)</td>
<td>(12, 79.67)</td>
<td>(12, 83.14)</td>
<td>(15, 77.94)</td>
<td>(15, 81.41)</td>
<td>(18, 79.67)</td>
</tr>
<tr>
<td>( G_3 )</td>
<td>( UAV_{3,1} )</td>
<td>( UAV_{3,2} )</td>
<td>( UAV_{3,3} )</td>
<td>( UAV_{3,4} )</td>
<td>( UAV_{3,5} )</td>
<td>( UAV_{3,6} )</td>
</tr>
<tr>
<td>Location</td>
<td>(33, 53.69)</td>
<td>(33, 57.16)</td>
<td>(33, 60.62)</td>
<td>(36, 55.43)</td>
<td>(36, 58.89)</td>
<td>(39, 57.16)</td>
</tr>
</tbody>
</table>
From the UAV swarm deployment in Figure 5, the initial detection mission state is shown in Figure 6a, where each light-gray circle represents the detection area of one UAV. In this case, all UAVs in the swarm are assumed to have the same detection radius of $3\sqrt{3}$ km, and the initial TCA is $\varepsilon_{\text{tot}} = 770.59$ km$^2$. Furthermore, the safety distance is assumed to be 0.2 km. The detection radius and safety distance can also be assigned based on the actual regions.

<table>
<thead>
<tr>
<th>$G_6$</th>
<th>$UAV_{6,1}$</th>
<th>$UAV_{6,2}$</th>
<th>$UAV_{6,3}$</th>
<th>$UAV_{6,4}$</th>
<th>$UAV_{6,5}$</th>
<th>$UAV_{6,6}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>(15, 12.12)</td>
<td>(15, 15.59)</td>
<td>(15, 19.05)</td>
<td>(18, 13.86)</td>
<td>(18, 17.32)</td>
<td>(21, 15.69)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$G_7$</th>
<th>$UAV_{7,1}$</th>
<th>$UAV_{7,2}$</th>
<th>$UAV_{7,3}$</th>
<th>$UAV_{7,4}$</th>
<th>$UAV_{7,5}$</th>
<th>$UAV_{7,6}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>(33, 5.20)</td>
<td>(33, 12.12)</td>
<td>(33, 8.66)</td>
<td>(36, 6.93)</td>
<td>(36, 10.39)</td>
<td>(39, 8.66)</td>
</tr>
</tbody>
</table>

Figure 6. Destruction and reconfiguration of UAVs. (a) Initial state. (b) Destruction. (c) Reconfiguration step 1. (d) Reconfiguration step T. (e) Redeployment. (f) Performance.

4.1.2. Destruction

The destruction states were randomly generated. Two kinds of destruction, namely local and random destruction, were considered simultaneously. For local destruction, the destruction center is a randomly sampled point on the mission area, and the destruction area is a randomly generated irregular polygon. For random destruction, the number of destroyed UAVs is assumed to follow the Poisson distribution with $\lambda = 1$.

For the mission and swarm deployment case in Figure 6a, the generated destruction states are illustrated in Figure 6b, and consist of two local destruction areas and a random destruction with three UAVs. The destruction centers of these two local destruction areas are (18, 17.32) and (5, 83.13), and the radii are 11 and 4, respectively. The destroyed UAVs in Figure 6b can be described as \{U$_{1,3}$, U$_{1,5}$, U$_{2,2}$, U$_{3,3}$, U$_{4,3}$, U$_{6,2}$, U$_{6,3}$, U$_{6,4}$, U$_{6,5}$, U$_{6,6}$\}, including both local destruction and random destruction. After this destruction process, the
current total coverage area is \( \varepsilon_{\text{tot}} = 615.02 \, \text{km}^2 \). All of the destruction information is presented in Table 2.

### Table 2. Destruction information.

<table>
<thead>
<tr>
<th>Destruction</th>
<th>Parameter</th>
<th>Parameter Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local destruction 1</td>
<td>Destruction center</td>
<td>(18, 17.32)</td>
</tr>
<tr>
<td></td>
<td>Destruction radius</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Destroyed UAVs</td>
<td>( UAV_{6,1}, UAV_{6,2}, UAV_{6,3}, UAV_{6,4}, UAV_{6,5}, UAV_{6,6} )</td>
</tr>
<tr>
<td>Local destruction 2</td>
<td>Destruction center</td>
<td>(5, 83.13)</td>
</tr>
<tr>
<td></td>
<td>Destruction radius</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Destroyed UAVs</td>
<td>( UAV_{1,1}, UAV_{1,5} )</td>
</tr>
<tr>
<td>Random destruction</td>
<td>Destroyed UAVs</td>
<td>( UAV_{3,2}, UAV_{4,3}, UAV_{5,4} )</td>
</tr>
</tbody>
</table>

### 4.1.3. Reconfiguration

For the reconfiguration process, the initial time step and the final time step are shown in Figure 6c and 6d, respectively. The UAV colored yellow represents the initial location of this reconfiguration process, while the UAV colored blue represents the final location. The red arrow represents the reconfiguration route from the initial location to the final location, which can be generated by the movement feature set \( \mathcal{M} \) according to (9). Each reconfiguration action is generated by an agent of the proposed multi-agent framework according to (9). For the case in Figure 6c,d, the DR action set \( \Phi \) is listed in Table 3.

### Table 3. Reconfiguration action set.

<table>
<thead>
<tr>
<th>Swarm Scale</th>
<th>UAV</th>
<th>Agent</th>
<th>Reconfiguration Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 × 6 UAVs</td>
<td>( UAV_{1,6} )</td>
<td>agent 1</td>
<td>(9, 77.94) (6, 79.67)</td>
</tr>
<tr>
<td></td>
<td>( UAV_{2,5} )</td>
<td>agent 2</td>
<td>(15, 81.41) (15, 15.59)</td>
</tr>
<tr>
<td></td>
<td>( UAV_{3,3} )</td>
<td>agent 3</td>
<td>(33, 60.62) (18, 17.32)</td>
</tr>
<tr>
<td></td>
<td>( UAV_{3,6} )</td>
<td>agent 3</td>
<td>(39, 57.16) (33, 57.16)</td>
</tr>
<tr>
<td></td>
<td>( UAV_{4,1} )</td>
<td>agent 4</td>
<td>(82, 36.37) (21, 15.59)</td>
</tr>
<tr>
<td></td>
<td>( UAV_{7,4} )</td>
<td>agent 7</td>
<td>(36, 6.93) (18, 13.86)</td>
</tr>
</tbody>
</table>

After this reconfiguration process, the UAV swarm has finished its redeployment. The current detection state is shown in Figure 6e. All UAVs in the swarm are assumed to have the same speed of 50 km/h. The speed can also be assigned based on the actual regions. The TCA is considered as a metric of UAV swarm performance. During this reconfiguration process, the UAV swarm performance exhibits a fluctuating upward trend, as shown in Figure 6f. The black dashed line in Figure 6f represents the TCA threshold \( \varepsilon_{\text{thr}} \), which is assumed to be 714 \( \text{km}^2 \). This TCA threshold can also be assigned on the basis of actual conditions. After finishing the reconfiguration process, the final TCA is \( \varepsilon_{\text{tot}}(\tau_T) = 732.31 \, \text{km}^2 \).

### 4.2. Discussion

In addressing the UAV swarm reconfiguration, the main objective is to generate an optimal feasible strategy. Extended analyses are now presented covering the method performance and the influence of various factors.

#### 4.2.1. Different Algorithms

This section evaluates the performance of the proposed QMIX method, although the DQN method and a cooperative game (CG) method [25] have also been used to generate this UAV swarm DRS. We used a single machine with one Intel i9 7980XE CPU and four
RTX2080 TI-A11G GPUs to train the QMIX network and the DQN network. During the training process, each episode generated a DRS for the randomly generated mission and destruction, as described in Section 4.1. This section presents the results of the following assessment process: for each training procedure, the training is paused every 100 episodes and the method runs 10 independent episodes with greedy action selection. Figure 7 plots the mean reward across these 10 runs for each method with independent mission and destruction details. As the 10 independent episodes are fixed, the mean reward of the CG method is a constant value. Thus, the reward curve of CG method is a straight line. The shading around each reward curve represents the standard deviation across the 10 runs. Over the training process, 100,000 episodes were executed for each method. The reward curves of these two methods fluctuate upward. In the first 17,000 episodes, the DQN method exhibits faster growth than the QMIX method. However, QMIX achieves a higher upper bound of the reward curve after 20,000 episodes. QMIX is noticeably stronger in terms of the final DR decision-making performance. The superior representational capacity of QMIX combined with the state information provides a clear benefit over the DQN method.

Figure 7. Reward of different algorithms.

4.2.2. Different Destruction Cases

For a given mission and swarm scale, the destruction process was randomly generated. The redeployment results were obtained by executing the QMIX reconfiguration strategy, as shown in Figure 8. The three subgraphs demonstrate the initial deployment status, the destruction status, and the redeployment results of the proposed QMIX algorithm. The geographical distributions of all mission areas and the swarm with $5 \times 6$ UAVs are the same in the three subgraphs, while the destruction states are different. After the reconfiguration process, the redeployment results in the three subgraphs demonstrate that the proposed QMIX method exhibits stable performance for this reconfiguration decision-making problem with different destruction patterns. This is because, during the training process, UAV destruction is generated randomly.
In addressing the UAV swarm redeployment, the main objective was to obtain an optimal feasible DRS strategy. Extended analyses of the optimization strategy were conducted to determine the influences of different methods. The QMIX method with high efficiency was proposed for this optimization strategy, while the DQN method and the CG method had also been used to solve the three destruction cases in Figure 8. The QMIX method gives optimal solutions with better TCA $\varepsilon_{\text{tot}}(\tau_T)$ and less TTFR $\tau_T$ than the other methods, as shown in Figure 9. The proposed method achieves the better solution, since these two methods may lead to local optima, such as a situation in which multiple UAVs have to spend more time moving during the reconfiguration process. The efficiencies of the methods are analyzed in Table 4. According to these results, the solution speeds of the QMIX and DQN are close, while the solution speed of CG method is significantly slower than the other two methods.
Figure 9. Reconfiguration under different destruction cases.

Table 4. Running time (in seconds) of different methods under different destruction cases.

<table>
<thead>
<tr>
<th>Different Destruction Cases</th>
<th>QMIX</th>
<th>DQN</th>
<th>CG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case (a) in Figure 8</td>
<td>20.652</td>
<td>19.215</td>
<td>43.642</td>
</tr>
<tr>
<td>Case (b) in Figure 8</td>
<td>20.857</td>
<td>19.618</td>
<td>44.258</td>
</tr>
<tr>
<td>Case (c) in Figure 8</td>
<td>20.116</td>
<td>19.128</td>
<td>42.289</td>
</tr>
</tbody>
</table>

4.2.3. Different Swarm Scales

Under different missions and swarm scales, the redeployment results obtained by executing the QMIX reconfiguration strategy are shown in Figure 10. The three subgraphs demonstrate the different deployment missions, the destruction status, and the redeployment results of the proposed method. The geographical distributions of all mission areas were randomly generated in the three subgraphs, and the initial swarm scales were 5 × 6, 7 × 6, and 9 × 6. Then, the destruction states were randomly generated. After the reconfiguration process, the redeployment results in the three subgraphs demonstrate that the proposed QMIX method exhibits stable performance under the different missions and swarm scales. During the training process, the missions and swarm scales were generated randomly. Thus, the superior representational capacity of QMIX combined with the mission state and swarm state information provides a clear benefit in terms of reconfiguration decision-making performance.

Figure 10. Reconfiguration under different swarm scales (a–c).

Again, keeping the same cases as in Figure 10 and using the QMIX method, the DQN method, and the CG method, we also analyzed the differences in algorithm performance. Under different missions and swarm scales, the QMIX method also gives optimal solutions with better TCA $\varepsilon_{tot}(T_T)$ and less TTFR $T_T$ than the other methods, as shown in Figure 11. The efficiencies of the methods are analyzed in Table 5. According to these results, the solution speeds of the QMIX and DQN are close for each case, while the solution
The speed of the CG method is significantly slower than the other two methods. Furthermore, the solution speeds of the QMIX and DQN are stable, and they do not exponentially decrease as the swarm scale increases. However, the solution speed of the CG method clearly decreases as the swarm scale increases. Thus, these results show that the proposed QMIX method exhibits stable DR decision-making performance for swarms with different scales.

Figure 11. Reconfiguration under different swarm sizes.

Table 5. Running time (in seconds) of different methods under different swarm sizes.

<table>
<thead>
<tr>
<th>Different Swarm Cases</th>
<th>QMIX</th>
<th>DQN</th>
<th>CG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case (a) in Figure 10</td>
<td>20.542</td>
<td>19.942</td>
<td>44.845</td>
</tr>
<tr>
<td>Case (b) in Figure 10</td>
<td>27.031</td>
<td>26.531</td>
<td>70.275</td>
</tr>
<tr>
<td>Case (c) in Figure 10</td>
<td>32.816</td>
<td>32.116</td>
<td>120.389</td>
</tr>
</tbody>
</table>

5. Conclusions

Distributed AI is gradually being applied to multi-UAVs. This paper has focused on DR decision-making for UAV swarm deployment optimization using a proposed MADRL framework. A two-layered decision-making framework based on MADRL enables UAV swarm redeployment, which maximizes swarm performance. Simulations using Python have demonstrated that the proposed QMIX method can generate a globally optimal DRS for UAV swarm redeployment. Furthermore, the results of the case study show that the QMIX method achieves a better swarm performance with less reconfiguration time than the other methods and exhibits stable and efficient solution speed. The DR decision-making problem considered in this paper is one of redeployment decision-making; the initial deployment planning was not addressed. Future research should emphasize the integration of UAV swarm initial deployments into decision-making frameworks.

Author Contributions: Conceptualization, Y.R., Q.F. and Q.W.; methodology, and Z.G. and Q.W.; software, Q.W.; validation, J.Z., Z.G., and Q.W.; formal analysis, Q.F. and Q.W.; investigation, Q.F. and Q.W.; resources, J.Z. and Q.W.; data curation, J.Z. and Q.W.; writing—original draft preparation, Z.G. and Q.W.; writing—review and editing, Y.R., Q.F. and Q.W.; supervision, Q.F.; project administration, Q.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China: 72001213.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.
References


**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.