drones

MDPI

Article

Task Allocation of Multiple Unmanned Aerial Vehicles Based on Deep Transfer Reinforcement Learning

Yongfeng Yin [1], Yang Guo [1,*], Qingran Su [2] and Zhetao Wang [1]

[1] School of Software, BeiHang University, Beijing 100083, China
[2] School of Computer Science and Engineering, BeiHang University, Beijing 100083, China
* Correspondence: sy2121105@buaa.edu.cn

Abstract: With the development of UAV technology, the task allocation problem of multiple UAVs is remarkable, but most of these existing heuristic methods are easy to fall into the problem of local optimization. In view of this limitation, deep transfer reinforcement learning is applied to the task allocation problem of multiple unmanned aerial vehicles, which provides a new idea about solving this kind of problem. The deep migration reinforcement learning algorithm based on QMIX is designed. The algorithm first compares the target task with the source task in the strategy base to find the task with the highest similarity, and then migrates the network parameters obtained from the source task after training, stored in the strategy base, so as to accelerate the convergence of the QMIX algorithm. Simulation results show that the proposed algorithm is significantly better than the traditional heuristic method of allocation in terms of efficiency and has the same running time.

Keywords: multiple UAVs; task assignment; deep reinforcement learning; transfer learning; neural network

Citation: Yin, Y.; Guo, Y.; Su, Q.; Wang, Z. Task Allocation of Multiple Unmanned Aerial Vehicles Based on Deep Transfer Reinforcement Learning. *Drones* **2022**, *6*, 215. https://doi.org/10.3390/drones6080215

Academic Editor: Oleg Yakimenko

Received: 27 July 2022
Accepted: 19 August 2022
Published: 20 August 2022

1. Introduction

With the development of UAV technology, UAV with autonomous ability has been widely used in various fields because of its strong flexibility and high efficiency, especially for obstacle avoidance [1], rescue [2], mapping [3] and other tasks. Due to the limitations of a single UAV, nowadays, UAVs mostly use the cluster method to perform complex tasks, which inevitably involves the task allocation problem. A good task allocation scheme can improve the efficiency and quality of completing tasks, which has attracted the attention of many scholars.

Most of the existing research at home and abroad uses heuristic methods to solve the problem of UAV task assignment. Zhang Ruipeng et al. [4] used the hybrid particle swarm algorithm to solve the problem of UAV task allocation. In order to jump out of the local optimum, they proposed a variable neighborhood search algorithm. By analyzing the hunting process of gray wolf groups, Peng Yalan et al. [5] modeled the group interaction mechanism and cooperative predation behavior of gray wolves, mapped the cooperative predation behavior mechanism of gray wolves to the dynamic task allocation of UAV clusters, and gave the dynamic task allocation process of UAV clusters. Yang Huizhen et al. [6] proposed a new circular competition method based on the dynamic ant colony labor division model. The method specifically solved the conflict problem caused by the dynamic change in the position of the task with time, and well dealt with the dynamic task allocation problem of heterogeneous multi AUV systems. Boyu Qin et al. [7] proposed a distributed group cooperative dynamic task assignment method based on the extended contract network protocol, which targeted and solved the dynamic assignment problem of UAV swarms performing cooperative reconnaissance and attack tasks on multiple targets in complex combat scenarios. Jiang Shiwen [8] used the swarm optimization algorithm based on beetle antennae search (BSO) in the task allocation process. The defect that particle swarm optimization algorithm is easy to "premature" is avoided by introducing the left and right

whisker bidirectional optimization mechanism of longicorn search into the particle swarm optimization algorithm.And the global optimal value is then locally optimized while enhancing the global optimization ability. However, the above algorithms that using heuristic methods to solve the task allocation problem are inevitably faced with the problem of falling into the local optimum. At this time, the deep reinforcement learning provides a new idea.

Reinforcement learning (RL) is one of the important research fields of machine learning, which is suitable for solving complex sequential decision problems. It has achieved remarkable results in the fields of robot control, scheduling optimization, and multi-agent collaboration [9]. The basic idea is that the agent adopts a trial-and-error strategy to learn the optimal strategy by interacting with the environment. Deep reinforcement learning has a wide range of applications in UAV formation control [10], navigation [11], and tracking [12].

However, the use of reinforcement learning to solve the task assignment problem is still in its infancy, and there are few related studies. Tang Fengzhu et al. [13] used the deep Q-network (DQN) algorithm based on the value function to solve the problem of low task completion due to task completion time constraints in the scenario of the random delivery of UAV tasks. Pengxing Zhu et al. [14] proposed an improved semi-random Q-learning algorithm. This algorithm increases the probability of obtaining a better task assignment scheme by changing the way Q-learning algorithm selects the next action during random exploration. Cheng Ding et al. [15] proposed a new RL approach to solve the task allocation problem of multi-AUV (autonomous underwater vehicle) in ocean currents, and introduced APAA (automatic policy amendment algorithm) to make AUVs understand how their behavior affects the final result. However, these studies using reinforcement learning methods all face the problem of slow algorithm convergence. Transfer reinforcement learning [16,17], as a paradigm of machine learning, can solve this problem very well.

Transfer learning [18] is a machine learning method whose goal is to transfer knowledge from one domain (source domain) to another domain (target domain) so that the target domain can achieve better learning effects. Usually, the application scenario of transfer learning is that the amount of data in the source domain is sufficient, while the amount of data in the target domain is small. The feature that the transferred task requires less data can be used to solve the problem of slow convergence of reinforcement learning.

In this paper, a deep transfer reinforcement learning algorithm based on QMIX is proposed to solve the multi-UAV task assignment problem involving materials scheduling. The use of the QMIX network cleverly avoids the problem of action space explosion and also avoids the problem that heuristic methods easily fall into local optimal solutions. The introduction of transfer learning solves the problem of long training time for deep reinforcement learning and improves real-time performance. Finally, the effectiveness of the algorithm is verified by simulation experiments.

2. Background

2.1. Reinforcement Learning

Reinforcement learning is mainly composed of agents, environments, states, actions, and rewards. The agent selects and executes an action in a certain state, and the environment gives rewards for this action and updates the state. The agent learns according to the reward given by the environment, and its purpose is to maximize the cumulative reward. The reinforcement learning model diagram is shown in Figure 1.

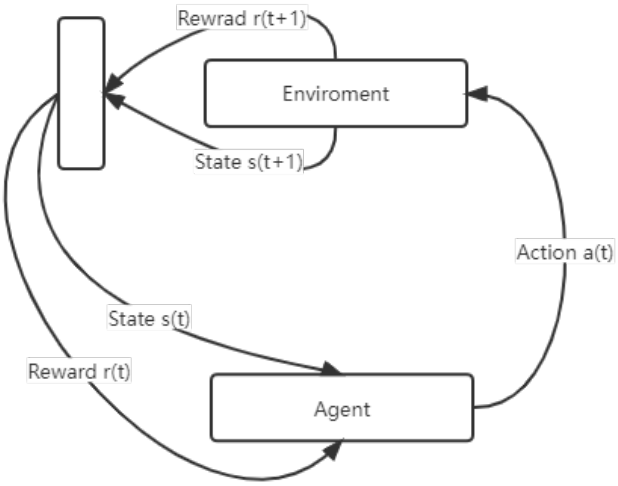

Figure 1. Reinforcement learning model diagram.

Generally, reinforcement learning problems can be described by a Markov decision process (MDP). MDP is defined as a quintuple $< S, A, Psa, Rsa, \gamma >$, where S represents the state space, A represents the action space, Psa represents the probability of taking action a and transitioning to state S' in a certain state S, Rsa represents the reward value obtained by taking action a in state S, and γ represents the discount factor, which is the degree of attenuation of future returns, where $s, s' \in S$, $a \in A$. When psa, Rsa, γ are all known, the model-based dynamic programming method can be used to solve it, otherwise, the model-free reinforcement learning method can be used to solve it. Dynamic programming is a method that divides a complex problem into sub-problems, solves the sub-problems, and finally combines the solutions of the sub-problems to solve the original problem.

Reinforcement learning training algorithms can be divided into two categories: value-based function and strategy-based iteration. The algorithm based on value function needs to update the long-term value $V(s)$ of each state s, and its update function is

$$V_{i+1}(s) \leftarrow \max_a \sum_{s'} (Ts, a, s')[R(s, a, s' + \gamma V_i(s')] \tag{1}$$

The algorithm based on the strategy iteration needs to be divided into two steps. First, a policy is assumed, and the iteration makes the policy cost function converge

$$V_{i+1}^k(s) \leftarrow \sum_{s'} T(s, \pi_k(s), s')[R(s, \pi_k(s), s') + \gamma V_i^k(s')] \tag{2}$$

Then we optimize according to all possible actions for each state,

$$\pi_{k+1}(s) = \arg\max_a \sum_{s'} T(s, a, s')[R(s, a, s') + \gamma V^{\pi_k}(s')] \tag{3}$$

Finally, the optimal strategy for each state is obtained.

2.2. QMIX Algorithm

QMIX [19] is one of the classic algorithms in multi-agent reinforcement learning. The essential difference between multi-agent and single-agent settings is that each agent in the multi-agent environment will interact with the environment, causing the environment to change. However, usually the reward value we obtain, r, is the reward obtained by all the agents acting together, so it is impossible to distinguish the individual reward value of the action taken by a single agent.

The most popular solution to this problem is centralized training with decentralized execution. Centralized training is used during the training process, and the global state can be used to ensure the convergence of the algorithm. In the execution process, each agent can only obtain its own current observation, thus distributed execution.

QMIX uses a hybrid network to combine the local value functions of single-agent, and adds global state information assistance in the training and learning process to improve the performance of the algorithm. The structure can be divided into three parts:

1. Agent network: each agent has to learn an independent value function $Qa(\tau^a, u_t^a)$. The network structure of the agent adopts the DRQN network structure.
2. Hybrid network: This part adopts a fully connected network structure, with $Qa(\tau^a, u_t^a)$ of each agent as input, and takes the joint action value function $Q_{tot}(\tau, u)$ as the output.
3. Hyperparameter network: This part is mainly responsible for generating the weights of the hybrid network.

The cost function of QMIX is

$$L(\theta) = \sum_{i=1}^{b} \left[y_i^{tot} - Q_{tot}(\tau, a, s; \theta) \right] \tag{4}$$

where b represents the number of samples sampled from empirical memory,

$$y^{tot} = r + \gamma \max \overline{Q}(\tau', a', s'; \overline{\theta}) \tag{5}$$

where $\overline{Q}(\tau', a', s'; \overline{\theta})$ represents the objective network function.

2.3. Transfer Learning

The purpose of transfer learning is to use the existing knowledge to learn new knowledge. Its core problem is to find the similarities between the source domain and the target domain, and transfer through these similarities. There are three key points in transfer learning, that is, what to transfer, how to transfer and when to transfer.

Algorithms of transfer learning can be mainly divided into several categories:

1. Instance-based transfer learning: Find data in the source domain that are similar to the target domain, adjust the data, and use them for training in the target domain.
2. Feature-based transfer learning method: The source domain and the target domain have some similar features. After feature transformation, the source domain and the target domain have the same data distribution.
3. Model-based transfer learning method: The target domain and the source domain share the parameters of the model, that is, the model trained with a large amount of data in the source domain is directly applied to the target domain.
4. Relation-based transfer learning method: When the target domain is similar to the source domain, their internal connections are also similar. This method transfers the relationship between data in the source domain to the target domain.

3. Problem Description

The research background of this paper is that natural disasters occur frequently in my country, and the rescue of a large number of trapped people after the disaster is the focus of the current emergency work. The use of drone swarms for the transportation of relief materials is an important research direction. The focus of this paper is the distribution scheme of heterogeneous UAVs to transport the materials carried by them to the rescue area.

3.1. Simulation Environment Modeling

Multiple UAVs form the UAV swarm to be allocated

$$UAV = \{uav_1, uav_2, \ldots, uav_M\} \tag{6}$$

where M is the number of drones in the cluster. In order to digitize the indicators of the drone, the parameters of each drone are represented by a collection

$$\{x_s, y_s, x_c, y_c, speed, voyage, dis, ret, loadlist\} \tag{7}$$

where

x_s: the horizontal axis coordinate of the starting point of the drone;

y_s: the vertical axis coordinate of the starting point of the drone;

x_c: the current horizontal axis coordinate of the drone;

y_c: the current vertical axis coordinates of the drone;

$speed$: the average speed of the drone;

$voyage$: The maximum range that the drone can fly in a single flight without supplementation;

dis: The current flight range of the drone;

ret: Whether the drone is returning, the returning drone will no longer be assigned tasks, the initial value is 0;

$loadlist$: The list of materials carried by the drone, which is an array.

Rescue areas with different urgency and needs form a task set

$$Target = \{target_1, target_2, \ldots, targett_N\} \tag{8}$$

where N is the number of rescue areas. The information of each rescue area is represented by a collection

$$\{x, y, level, loadlist\} \tag{9}$$

where

x: the horizontal axis coordinate of the rescue area;

y: the vertical axis coordinate of the rescue area;

$level$: The emergency level of the rescue area;

$loadlist$: The list of materials needed by the rescue area, which is an array.

When the environment is initialized, all the information about the drone swarm and rescue areas needs to be provided. After initialization, the environment will record the initialization information to facilitate the reset of task assignment. The result of the task assignment is entered in the form of $[x_1, x_2, \ldots, x_M]$, where $x_i \in [0, N]$. $x_i \in [0, N-1]$ means that the drone i goes to the rescue area $x_i + 1$ for material transportation. If the drone can meet the needs of the rescue area, we unload the amount of materials needed in the rescue area from the drone. Otherwise, unload all the materials on the drone to the rescue area, and modify the location of the drone, the amount of materials carried by the drone, the distance flown by the drone, and the amount of materials needed in the rescue area. When $x_i = N$, it means that the drone completes the flight mission and returns, and the parameter $ret = 1$ of the drone needs to be set.

3.2. MDP Model of Task Allocation

Using model-free reinforcement learning to solve the multi-UAV task assignment problem requires modeling the problem as a Markov decision process. Modeling requires the definition of state space, observation space, action space, and reward function.

3.2.1. State Space

According to the description of the simulation environment above, it can be found that only the drone cluster and the rescue areas are involved in this environment, so the state space only needs to contain all the information of the drone cluster and the rescue areas. For the sake of simplicity, we flatten the UAV information and the rescue area information. The

state space data form is shown in (10), where the expression of UAV_i is (11), and $Target_i$ expression is Formula (12).

$$[UAV_1, UAV_2, \cdots, UAV_M, Target_1, \cdots, Target_N] \tag{10}$$

$$\begin{aligned}&[uav_i^{x_s}, uav_i^{y_s}, uav_i^{x_c}, uav_i^{y_c}, uav_i^{speed},\\ &uav_i^{voyage}, uav_i^{dis}, uav_i^{ret}, uav_i^{loadlist_1},\\ &uav_i^{loadlist_2}, \ldots, uav_i^{loadlist_{tot}}]\end{aligned} \tag{11}$$

$$\begin{aligned}&[target_i^x, target_i^y, target_i^{level},\\ &target_i^{loadlist_1}, target_i^{loadlist_2}, \ldots, target_i^{loadlist_{tot}}]\end{aligned} \tag{12}$$

3.2.2. Observation Space

Since the problem involved in this paper is the multi-UAVs task assignment, involving multiple agents, the distributed reinforcement learning method is used to solve it. The observation space of each UAV is related to itself but not related to other UAVs. The observation space is not the same as the state space, so it needs to be redefined. The observation space of the *i*-th UAV is shown in (13).

$$[UAV_i, Target_1, Target_2, \cdots, Target_N] \tag{13}$$

3.2.3. Action Space

The action space of each drone is related to the current state of the drone. When $UAV_i \cdot ret = 0$, that is, when the drone is not in the returning state, the action space of the drone is $[0, 1, \cdots, j, \cdots N-1]$, where action j represents the UAV to perform the task $target_{i+1}$. When $UAV_i \cdot ret = 1$, the drone is in the returning state, and the action space of the drone is $[N]$, where $target_N$ is a virtual target point, which means that the drone continues to return to home.

3.2.4. Reward Function

The reward function is very important for reinforcement learning because the agent needs to learn the action through the reward value. It can be said that the quality of the reward function determines the convergence of the algorithm and the convergence result. The goal of the task allocation problem solved in this paper is to complete the task allocation under the condition that the total voyage of the UAV is as small as possible. At the same time, priority shall be given to meeting the material needs of the rescue areas with a high degree of urgency. Experiments show that the reward value calculation formula shown in (14) has a good effect, where $offerResourcee_{uav_i}$ means that uav_i can satisfy $target_j$ the number of resources required, $needResouce_{target_j}$ indicates the number of resources required by $target_j$, $targett_j$ represents the drone selected by uav_i to perform the task, and $\sum_{i=0}^{M} dis(uav_i, target_j)$ represents the Euclidean distance between the drone and the target point.

$$\begin{aligned}Reward = &\sum_{i=0}^{M} \frac{offerResource_{uav_i}}{needResouce_{target_j}} \cdot target_j \cdot level -\\ &\sum_{i=0}^{M} dis(uav_i, target_j)\end{aligned} \tag{14}$$

4. Task Assignment Algorithm of Deep Transfer Reinforcement Learning Algorithm Based on QMIX

4.1. DQN-Based Task Assignment Method

The multi-UAV dynamic task assignment problem is essentially a NP-hard problem, and it is easy to fall into the local optimal solution when traditional heuristic methods are used to solve this problem. Therefore, the improvement of the algorithm focuses on how to jump out of the local optimum to find the global optimum, and the emergence of deep reinforcement learning provides a new solution. When the UAV makes a decision, it only depends on the state at the current moment, so the dynamic Markov decision process can be used to solve such problems. At present, most of the related research on the UAV task assignment based on reinforcement learning uses the DQN algorithm [20] based on the value function to solve it.

Since DQN is a single-agent reinforcement learning algorithm, in the implementation process, all UAVs need to be treated as a single-agent. The current state space S is the current data of all drones and the current data of all tasks, and the action space A is the combination of actions of all drones. The update algorithm of its Q-value function is

$$Q(s,a) \leftarrow Q(s,a) + \mu[R + \gamma \max_{a} Q(s',a) - Q(s,a)] \tag{15}$$

4.2. Task Allocation Algorithm Based on QMIX

For the task assignment scenario mentioned in this article, if the DQN algorithm is used to solve it, its action space is M^N (M is the number of drones, N is the number of target points); such an action space is obviously unacceptable when M and N are large. To solve this problem, this paper chooses the QMIX algorithm. Like all reinforcement learning methods, the agent in the QMIX algorithm interacts with the environment through a trial-and-error mechanism, optimizes by maximizing cumulative rewards, and finally achieves the optimal strategy. Different from the centralized strategy of DQN algorithm, the QMIX algorithm adopts the structure of centralized training and distributed execution, and its network is divided into the mixing network and agent network. Each UAV has its own agent network, the input of the network is the observation space of the UAV, and the output is the Q value function of the UAV. The input of the mixing network includes the state space, and the output is the Q-value function of the joint action. The upper limit of the action space of each drone is N, and the action space decreases from M^N to N, which solves the problem of action space explosion. The specific implementation steps are in Algorithm 1.

4.3. Deep Transfer Reinforcement Learning Algorithm Based on QMIX

All deep reinforcement learning algorithms have a common problem: poor real-time performance. Faced with the same allocation problem, the time required by the QMIX method is 10 times or even 100 times that of the genetic algorithm. In the face of such a time difference, although the allocation result obtained by the QMIX algorithm is obviously better than the genetic algorithm, its availability is also very low. In response to this problem, this paper introduces transfer learning to solve it. First, a representative set of source tasks that can cover most of the assigned tasks is learned, and the network parameters of its mixing network and agent network are stored in the policy library. Whenever a new task needs to be assigned, the similarity between the task and the task in the policy library is calculated, and the task with the highest similarity is found to transfer network parameters. Combining the transfer learning with the QMIX algorithm, the algorithm proposed in this paper is the deep transfer reinforcement learning algorithm based on QMIX. The algorithm flowchart of the algorithm is shown in Figure 2, and the specific implementation steps are in Algorithm 2.

Algorithm 1 Task allocation algorithm based on QMIX.

Input: UAV swarm information, target point information
Output: Agent Network and Mixing Network network parameters

Initialize network parameters, data storage unit D, capacity M, total number of iteration rounds T, target network parameter update frequency p
while i $\leq$ T **do**
    Simulation environment, UAV cluster information, target point information restoration
    while step $\leq$ step_{limit} **do**
        Obtain the current mission state S, the observed value O of each UAV and the feasible action A of each UAV from the environment
        Each drone obtains the Q value of each action taken in the current state through eval netwrok
        Choose an action based on the resulting Q value
        Get reward R from the environment based on the actions of all drones in the current state
        Store S,S_{next}, the observed value of each UAV O, the action taken by each UAV A, the feasible action of each UAV U, the reward R, and whether the assigned task is completed *terminated*, into the data storage unit D
    if len(D) $\geq$ M **then**
        Randomly sample some data from D
        Update the network parameters according to the loss function

$$L(\theta) = \sum_{i=1}^{b} \left[y_i^{tot} - Q_{tot}(\tau, a, s; \theta) \right]$$

    if i mod $p == 0$ **then**
        Update the parameters of the target network

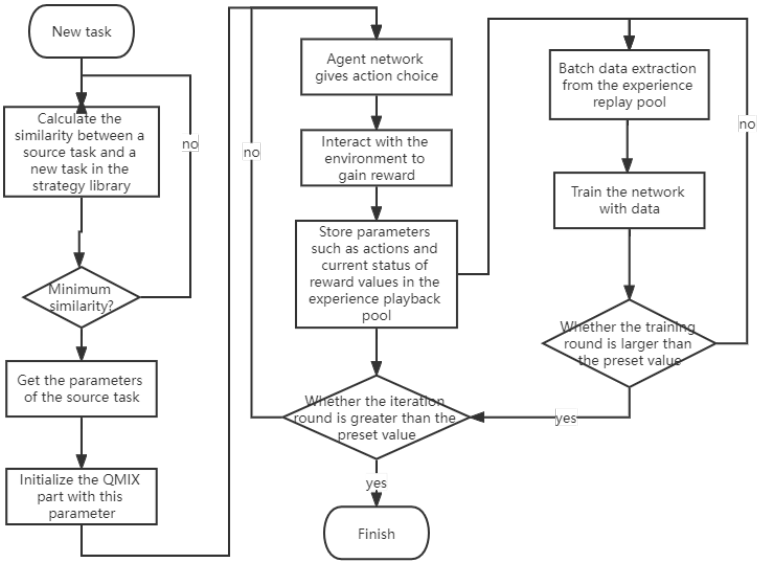

Figure 2. Flow chart of QMIX based deep transfer reinforcement learning algorithm.

Algorithm 2 Deep transfer reinforcement learning algorithm based on QMIX.

Input: UAV swarm information, target point information, strategy library and its corresponding source task information

Output: Task assignment results

while i $\leq$ N **do**
    Simulation environment, UAV cluster information, target point information restoration
    Calculate the sum total of the squared differences between the input information and the information corresponding to the ith source task
    if total $\leq$ min **then**
        min = total
        flag = i
Copy the network parameters of the *i* source task to the network of the new task
Execute Algorithm 1, but do not need to initialize the network parameters

5. Algorithm Verification and Analysis

5.1. Experimental Setup

In order to verify the effectiveness of this algorithm, 5 UAVs and 10 target points were randomly generated this time, and allocated for 3 kinds of resources. The data structures of UAVs and target points are shown in Figures 3 and 4. This experiment is coded in Python, and the neural network part of the QMIX algorithm is implemented using the Pytorch deep learning framework. The experimental platform is a computer with Apple M1 pro chip and 16 GB memory.

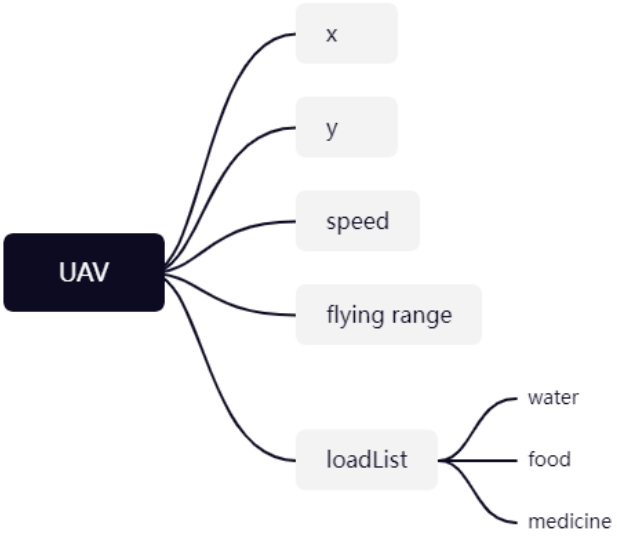

Figure 3. UAV data structure.

5.2. Experimental Process

5.2.1. Task Allocation Based on QMIX Algorithm

The parameter settings of the algorithm are shown in Table 1, and the cumulative reward value of the QMIX algorithm is shown in Figure 5. It can be seen from the figure that the cumulative reward value obtained by this algorithm increased rapidly in the first 7700 cycles, and then converged to 2.443. Through analysis, it can be seen that the fluctuations after 5000 cycles are affected by random actions with small probability.

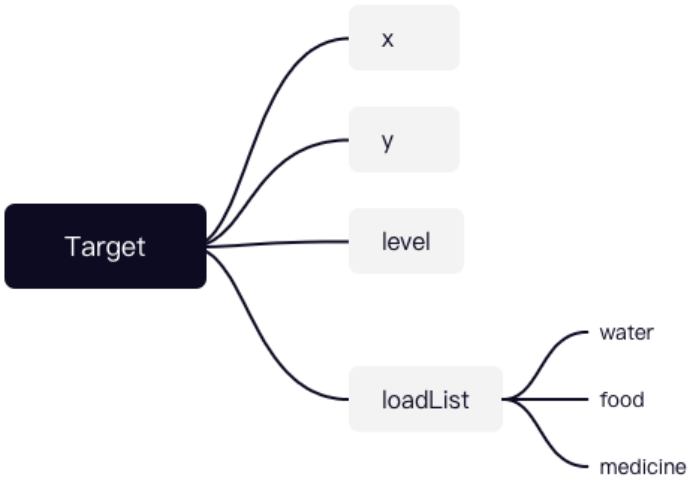

Figure 4. Target point data structure.

Table 1. QMIX algorithm parameter list.

Name	Value
seed	123
n_epochs	50,000
evaluate_per_epoch	100
batch_size	32
buffer_size	100
update_target_params	200
drqn_hidden_dim	64
qmix_hidden_dim	32

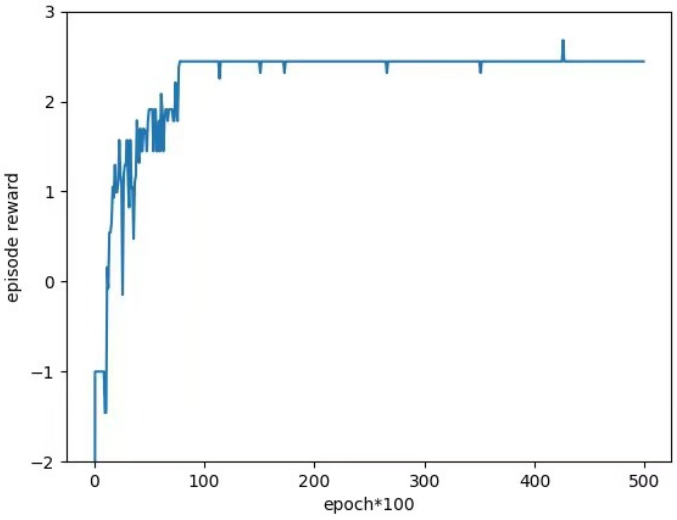

Figure 5. Cumulative reward value of QMIX algorithm.

5.2.2. Task Assignment Based on Genetic Algorithm

The cumulative reward value obtained by the heuristic genetic algorithm for the same set of data is shown in Figure 6, and its parameter settings are shown in Table 2. Observing the image, it can be found that in about the first 20 cycles, the cumulative reward value obtained by the algorithm shows an extremely rapid upward trend. In the 20th to 70th cycles, the cumulative reward obtained by the algorithm shows a certain platform period, but eventually rises. It can be judged that it falls into the local optimum. Due to the existence of random actions, it successfully jumps out of the local optimum and

gradually moves toward global optimization. From the 70th epoch, the cumulative reward system is no longer increased, and finally converges to 1.589, significantly lower than the qmix algorithm.

Table 2. Genetic algorithm parameter list.

Name	Value
seed	123
num_total	30
iteration	200
max_step	5

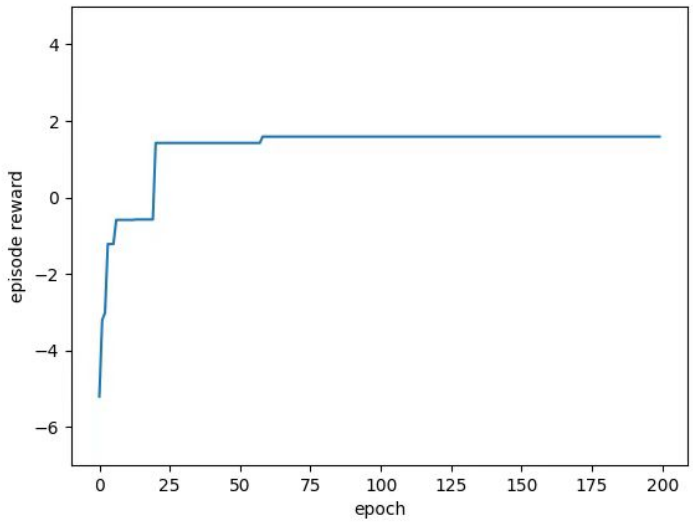

Figure 6. Cumulative reward value of genetic algorithm.

5.2.3. Deep Transfer Reinforcement Learning Algorithm Based on QMIX

From the above two experiments, we can find that although the cumulative reward value of the genetic algorithm is much lower than that of the QMIX algorithm, the iteration period required for the QMIX algorithm to converge is 250 times that of the genetic algorithm. Therefore, it is necessary to introduce transfer learning at this time, and find the source task with the smallest similarity difference with the secondary task in the strategy library to perform parameter transfer. After the network parameters are migrated, the comparison of the cumulative reward value between this algorithm and the genetic algorithm is shown in Figure 7, and the parameter settings of this algorithm are shown in Table 3. Observing the image, it can be found that the algorithm drops sharply in the first 20 cycles. After analysis, it is because the new task is different from the source task. Then, it shakes sharply in 20–160 cycles, and finally converges at 2.828. The result is significantly better than the genetic algorithm.

Table 3. Algorithm parameter setting in this paper.

Name	Value
seed	123
n_epochs	200
evaluate_per_epoch	1
batch_size	32
buffer_size	100
update_target_params	200
drqn_hidden_dim	64
qmix_hidden_dim	32

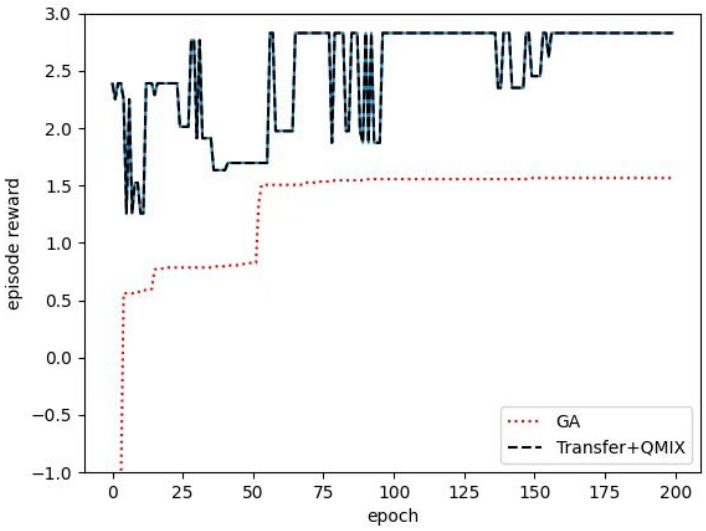

Figure 7. Comparison chart of cumulative reward value between genetic algorithm and this algorithm.

Table 4 shows the comparison of the running time of the genetic algorithm, the algorithm based on QMIX, and the algorithm proposed in this paper. It can be found that the optimal value converged by the algorithm proposed in this paper is obviously better than the genetic algorithm, and its running time is basically the same as that of the genetic algorithm.

Table 4. Algorithm runtime comparison.

Algorithm	Running Time/s
Genetic algorithm	6
QMIX algorithm	120
Algorithm in this paper	8

5.2.4. Universal Verification

The experiment given above is the allocation effect when the number of UAVs is less than the number of target points. Now experiments are carried out for the number of UAVs equal to the number of target points, and the number of UAVs is greater than the number of target points to verify the universality of the algorithm given in this paper. In this paper, the algorithm is tested on 5, 10, and 20 UAVs and 10, 20, and 30 target points. The experimental results show that the cumulative reward value of the algorithm proposed in this paper is obviously better than that of genetic algorithm, and the time is equivalent to that of genetic algorithm.

Figures 8 and 9 are the task assignment results of 10 UAVs and 10 target points. It can be seen that the optimal value of the final convergence of the algorithm proposed in this paper is obviously better than that of the genetic algorithm.

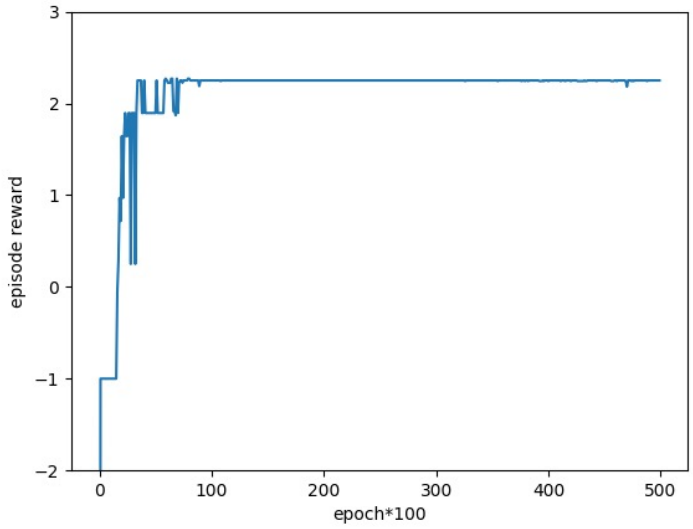

Figure 8. Cumulative reward value of qmix algorithm (M = N).

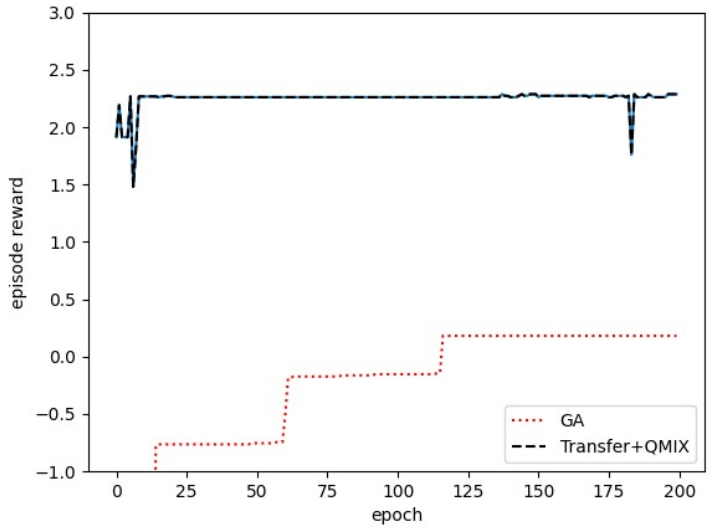

Figure 9. Comparison chart of cumulative reward value between genetic algorithm and this algorithm (M = N).

Figures 10 and 11 show the task assignment results of 10 UAVs and 5 target points. It can be seen that the optimal value of the final convergence of the algorithm proposed in this paper is still significantly better than that of the genetic algorithm.

Through the experimental verification in this section, the algorithm proposed in this paper is effective when the number of drones is less than the number of target points, the number of drones is equal to the number of target points, and the number of drones is greater than the number of target points.

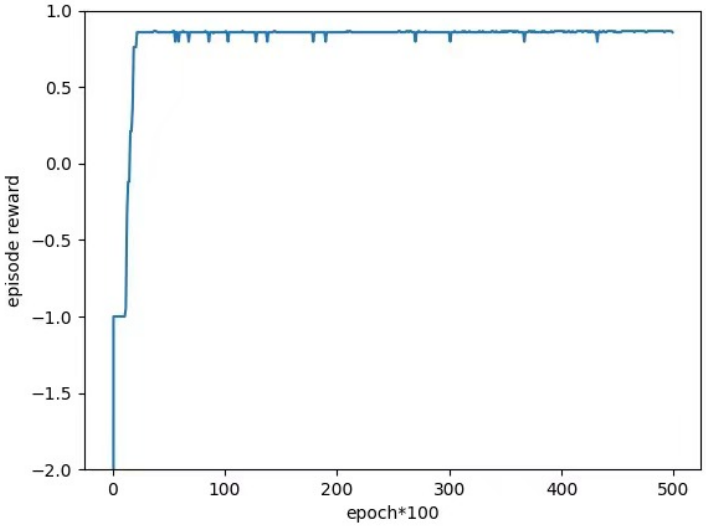

Figure 10. Cumulative reward value of qmix algorithm (M > N).

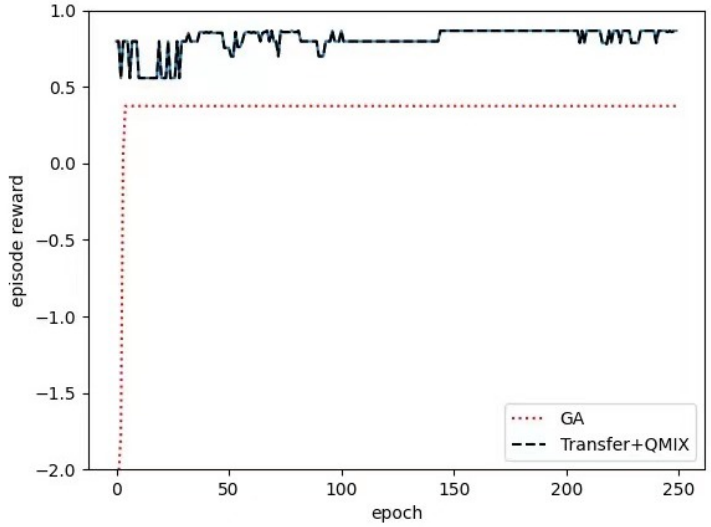

Figure 11. Comparison chart of cumulative reward value between genetic algorithm and this algorithm (M > N).

6. Conclusions

Aiming at the task assignment problem of multiple UAVs, this paper proposes a task assignment algorithm based on deep transfer reinforcement learning based on QMIX. The algorithm helps to combine the experience of the source task to the new task and accelerate the convergence speed by migrating the network parameters learned by QMIX in the source task. The experimental results show that the algorithm proposed in this paper is significantly better than the traditional heuristic method in allocation efficiency, and the running time is basically the same. The algorithm provides a new idea for using artificial intelligence methods to solve the task assignment problem.

The future research work is mainly divided into the following aspects: First, improve the similarity calculation method between the new task and the source task group. The existing similarity calculation method is too straightforward, which may lead to the source tasks with high similarity not being identified. Second, consider whether there is a better migration method. The existing migration mode is network parameter migration, which requires the number of UAVs and target points of the migrated task to be consistent with

that of the new task. However, QMIX is a distributed decision algorithm, and this migration method does not make good use of its advantages. Third, the current algorithm mainly considers the range constraint, but time is also a very important factor in practical scenarios, which needs to be taken into account in subsequent research.

Author Contributions: Funding acquisition, Y.Y.; Investigation, Y.G.; Resources, Y.Y.; Software, Y.G.; Validation, Z.W.; Writing—original draft, Y.G.; Writing—review & editing, Y.Y. and Q.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

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

References

1. Aldao, E.; González-deSantos, L.M.; Michinel, H.; González-Jorge, H. UAV Obstacle Avoidance Algorithm to Navigate in Dynamic Building Environments. *Drones* **2022**, *6*, 16. [CrossRef]
2. Zimroz, P.; Trybała, P.; Wróblewski, A.; Góralczyk, M.; Szrek, J.; Wójcik, A.; Zimroz, R. Application of UAV in search and rescue actions in underground mine—A specific sound detection in noisy acoustic signal. *Energies* **2021**, *14*, 3725. [CrossRef]
3. Steenbeek, A.; Nex, F. CNN-Based Dense Monocular Visual SLAM for Real-Time UAV Exploration in Emergency Conditions. *Drones* **2021**, *6*, 79. [CrossRef]
4. Zhang, R.; Feng, Y.; Yang, Y. Hybrid particle swarm optimization algorithm for cooperative task allocation of multiple UAVs. *J. Aeronaut.* **2022**, 1–15.
5. Peng, Y.; Duan, H.; Zhang, D.; Wei, C. Dynamic task allocation of UAV cluster imitating gray wolf cooperative predation behavior. *Control Theory Appl.* **2021**, *38*, 1855–1862.
6. Yang, H.; Wang, Q. Multi AUV task allocation method based on dynamic ant colony labor division model. *Control. Decis.-Mak.* **2021**, *36*, 1911–1919.
7. Qin, B.; Zhang, D.; Tang, S.; Wang, M. Distributed Grouping Cooperative Dynamic Task Assignment Method of UAV Swarm. *Appl. Sci.* **2022**, *12*, 2865. [CrossRef]
8. Jiang, S. *Research and Simulation of Multi UAV Mission Planning Algorithm in Dynamic Environment*; University of Electronic Science and Technology: Chengdu, China, 2021.
9. Li, Y.; Hu, J. Application and Prospect of reinforcement learning in the field of unmanned vehicles. *Inf. Control* **2022**, *51*, 129–141.
10. Xiang, X.; Yan, C.; Wang, C.; Yin, D. Coordinated control method of fixed wing UAV formation based on deep reinforcement learning. *J. Aeronaut.* **2021**, *42*, 420–433.
11. Huang, H.; Yang, Y.; Wang, H.; Ding, Z.; Sari, H.; Adachi, F. Deep reinforcement learning for UAV navigation through massive MIMO technique. *IEEE Trans. Veh. Technol.* **2019**, *69*, 1117–1121. [CrossRef]
12. Akhloufi, M.A.; Arola, S.; Bonnet, A. Drones Chasing Drones: Reinforcement Learning and Deep Search Area Proposal. *Drones* **2019**, *3*, 58. [CrossRef]
13. Tang, Y.; Tang, X.; Li, C.; Li, X. Dynamic task allocation of multiple unmanned aerial vehicles based on deep reinforcement learning. *J. Guangxi Norm. Univ. (Nat. Sci. Ed.)* **2021**, *39*, 63–71.
14. Zhu, P.; Fang, X. Multi-UAV Cooperative Task As-signment Based on Half Random Q-Learning. *Symmetry* **2021**, *13*, 2417. [CrossRef]
15. Ding, C.; Zheng, Z. A Reinforcement Learning Approach Based on Automatic Policy Amendment for Multi-AUV Task Allocation in Ocean Current. *Drones* **2022**, *6*, 141. [CrossRef]
16. Hu, P.; Pan, Q.; Wu, S.; Ma, J.; Guo, Y. Multi agent system cooperative formation obstacle avoidance and collision avoidance control based on transfer reinforcement learning. In Proceedings of the 2021 China Automation Conference, Zhanjiang, China, 5–7 November 2021.
17. Shi, H.; Li, J.; Mao, J.; Hwang, K.S. Lateral transfer learning for multiagent reinforcement learning. *IEEE Trans. Cybern.* **2021**. [CrossRef] [PubMed]
18. Zhuang, F.; Qi, Z.; Duan, K.; Xi, D.; Zhu, Y.; Zhu, H.; Xiong, H.; He, Q. A comprehensive survey on transfer learning. *Proc. IEEE* **2020**, *109*, 43–76. [CrossRef]
19. Rashid, T.; Samvelyan, M.; Schroeder, C.; Farquhar, G.; Foerster, J.; Whiteson, S. Qmix: Monotonic value function factorisation for deep mul-ti-agent reinforcement learning. *Int. Conf. Mach. Learn.* **2018**. Available online: https://arxiv.org/abs/2003.08839 (accessed on 27 July 2022).
20. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv* **2013**, arXiv:1312.5602.