electronics

MDPI

Article

Task Assignment for UAV Swarm Saturation Attack: A Deep Reinforcement Learning Approach

Feng Qian, Kai Su *, Xin Liang and Kan Zhang

Department of Management Engineering and Equipment Economics, Naval University of Engineering, Wuhan 430033, China

* Correspondence: keppelsue@163.com

Abstract: Task assignment is a challenging problem in multiple unmanned aerial vehicle (UAV) missions. In this paper, we focus on the task assignment problem for a UAV swarm saturation attack, in which a deep reinforcement learning (DRL) framework is developed. Specifically, we first construct a mathematical model to formulate the task assignment problem for a UAV swarm saturation attack and consider it as a Markov Decision Process (MDP). We then design a policy neural network using the attention mechanism. We also propose a training algorithm based on the policy gradient method so that our agent can learn an effective task assignment policy. The experimental results have shown that our DRL method can generate high-quality solutions for different problem scales, which meets the requirements of real-time and flexibility in the actual situation.

Keywords: task assignment; deep reinforcement learning (DRL); unmanned aerial vehicles (UAVs); saturation attack

1. Introduction

In recent years, the technology and applications of unmanned aerial vehicles (UAVs) have developed rapidly. Compared with manned aerial vehicles, UAVs have higher maneuverability and better operational sustainability. The use of UAVs for mission execution can effectively reduce personnel risk and maintenance costs, and can therefore replace manned vehicles to perform some harsh, dangerous, and tedious missions. A typical application of UAVs in the military could be UAV Swarm Saturation attack [1]. Specifically, it refers to how the attacking side adopts a high density, continuous attack strategy to cause the enemy's defense system to collapse. Then, the UAV Swarm will have a much higher probability of breaking through defenses and achieve the goal of destroying the enemy's high-value targets. In order to achieve cooperation and control of multiple UAVs, the task assignment for UAV swarm saturation attack is of great significance.

Task assignment for a UAV swarm is to allocate multiple UAVs to specific tasks according to the number and types of the vehicles, the tasks to be performed, and the conditions of the environment, which is essentially an optimization problem under multiple constraints [2]. Besides, the task assignment problem is also NP-hard [3], which means that the optimal solution comes from searching the whole solution space. In practical scenarios, the task assignment problem for a UAV swarm is challenging, especially in the complex dynamic environment. The traditional method is to use the deterministic algorithms. Darrah et al. considered the multiple UAV dynamic task allocation problem in a Suppression of Enemy Air Defense (SEAD) mission in [4]. They used Mixed Integer Linear Programming (MILP) to assign vehicles to specific tasks. Schumacher et al. also proposed a method based on MILP to solve the constrained optimization problem for UAV task assignment in [5]. In [6], Nygard et al. proposed the dynamic network flow optimization models and used the centralized optimization algorithm to solve the air vehicle resource allocation problem. Ye et al. developed an extended consensus-based bundle algorithm

check for updates

Citation: Qian, F.; Su, K.; Liang, X.; Zhang, K. Task Assignment for UAV Swarm Saturation Attack: A Deep Reinforcement Learning Approach. *Electronics* **2023**, *12*, 1292. https://doi.org/10.3390/electronics12061292

Academic Editor: Shiho Kim

Received: 13 February 2023
Revised: 26 February 2023
Accepted: 1 March 2023
Published: 8 March 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

with task coupling constraints (CBBA-TCC) in [7] to solve the multi-task assignment problem with task coupling constraints in the heterogeneous multi-UAV system.

However, the computational complexity of the task assignment problem increases exponentially with the growth of the number of targets and UAVs. It is hard for the exact algorithms that pursue the optimal solution to complete the searching process in an acceptable time. In order to speed up the solving process, another commonly used method is applying heuristic algorithms. Shima et al. proposed genetic algorithms to solve the multiple task assignment problem for cooperating UAVs in [3,8]. Jia et al. considered the cooperative multiple task assignment problem with stochastic velocities and time windows for heterogeneous UAVs in [9] and proposed a modified genetic algorithm to solve it. In [10], Zhu et al. focused on the multirobot task allocation problem (MRTAP) and proposed the self adaptive evolutionary game particle swarm optimization algorithm to solve it. Zhao et al. considered the search and rescue scenario in [11] and they proposed a heuristic distributed task allocation method to solve the problem. Zhen et al. proposed an intelligent self-organized algorithm to solve a cooperative search–attack mission planning problem for multiple unmanned aerial vehicles in [12]. Fan et al. proposed a modified nature-inspired meta-heuristic methodology for heterogeneous unmanned aerial vehicle system task assignment problem in [13]. Xia et al. proposed a system framework for solving the problem of multi-UAV cooperative task assignment and track planning for ground moving targets in [14]. The heuristic algorithms can obtain a solution for large-scale task assignment problems in an acceptable time, but usually fall into a local optimum and stop searching process early. Besides, the solutions need to be encoded into a vector in these algorithms. Once the problem scale or characteristics changes, the old encoding strategy is hard to apply to the new problem.

With the development of artificial intelligence technology in recent years, deep reinforcement learning (DRL) has achieved remarkable breakthroughs in many fields. Deep reinforcement learning is mainly used to make sequential decisions, i.e., to make action choices based on the current environmental conditions and continuously adjust their strategies based on the feedback from the actions to achieve the goals. The performance of deep reinforcement learning on problems such as AlphaGo Zero [15] and Atari [16] in recent years has demonstrated its powerful learning and optimization decision-making capabilities. Additionally, deep reinforcement learning techniques have shown a significant advantage for combinatorial optimization problems. Combinatorial optimization, which is the optimal selection of decision variables in a discrete decision space, naturally has similar characteristics to "action selection" in reinforcement learning. Therefore, it is a good choice to solve the traditional combinatorial optimization problem using deep reinforcement learning methods. Compared with traditional optimization algorithms, DRL-based combinatorial optimization algorithms have a series of advantages such as fast solving speeds and high generalization ability, which is a hot research topic in recent years. In 2015, Vinyals et al. [17] proposed a pointer network (Ptr-Net) model for solving combinatorial optimization problems by analogizing it to a machine translation process. The network was trained using supervised learning and achieved good optimization results on the TSP problem. Bello et al. [18] used reinforcement learning to train the pointer network model. They considered each problem instance as a training sample, used the REINFORCE reinforcement learning algorithm for training, and introduced the critic network as a baseline to reduce the training variance. Kool et al. [19] proposed a new method for solving multiple combinatorial optimization problems using the attention mechanism, based on the Transformer model [20]. Furthermore, the algorithm outperforms previous pointer network models on multiple optimization problems.

Furthermore, deep reinforcement learning techniques have also been applied to many practical optimization problems. Zhao et al. [21] considered a task allocation problem for UAVs in the presence of environment uncertainty and proposed a Q-learning based fast task allocation (FTA) algorithm. Tian et al. [22] presented the multi-robot task allocation algorithm for fire disaster response based on reinforcement learning. Yang et al. [23]

focused on a resource management problem for the ultra-reliable and low-latency internet of vehicle communication networks and presented a decentralized actor–critic reinforcement learning model with a new reward function to learn the optimal policy. Luo et al. [24] focused on the missile target assignment (MTA) problem and proposed a data-driven policy optimization with deep reinforcement learning (PODRL) for the adversarial MTA. Liang et al. [25] proposed a deep reinforcement learning model to control the traffic light. Huang et al. [26] focused on online computation offloading in wireless powered mobile–edge computing networks and proposed a deep reinforcement learning-based online offloading (DROO) framework.

In practice, task assignment for UAV swarm saturation attacks usually occur in the hostile environment, which is complex and stochastic. In our paper, we propose a deep reinforcement learning method to solve the task assignment problem, which meets the requirements of real-time and flexibility in the actual situation. The simulation and experiments have shown that our reinforcement learning agent based on deep neural network converges rapidly and stably using the policy gradient method. Additionally, the solutions obtained from our policy network are effective for both large- and small-scale problems. The contributions of our work are listed as follows.

1. We construct a mathematical model to formulate the task assignment problem for UAV swarm saturation. Furthermore, we consider the task assignment model as a Markov Decision Process from the reinforcement learning perspective;
2. We build a task assignment framework based on the deep neural network to generate solutions for adversarial scenarios. The policy network uses the attention mechanism to pass information and guarantees the effectiveness and flexibility of our algorithm under different problem scales;
3. We propose a training algorithm based on the policy gradient method so that our agent can learn an effective task assignment policy from the simulation data. We also design a critic baseline to reduce the variance of the gradients and increase the learning speed.

The rest of our article is organized as follows. The problem formulation of task assignment for UAV swarm saturation attack is provided in Section 2. In Section 3, a task assignment framework based on deep reinforcement learning is constructed to provide solutions for the problem. Then, the simulation and analysis of the proposed method is conducted in Section 4. Finally, the conclusions and future works are presented in Section 5.

2. Problem Formulation

In this section, a formulation of the task assignment problem for a UAV swarm saturation attack is presented. Our research focuses on the combat scenario against hostile surface ships involving a group of heterogeneous UAVs. A mathematical programming model of the task assignment problem is established and we formulate it as a combinatorial optimization problem. Furthermore, we consider the task assignment model as a Markov Decision Process from the reinforcement learning perspective.

2.1. Scenario Description and Assumptions

In this paper, we focus on the combat scenario of a UAV swarm saturation attack against hostile surface ships and aim to research the issue of a multi-UAV task assignment. In order to obtain the optimal combat effectiveness, we allocate tasks to heterogeneous UAVs according to the battlefield situation. When multiple UAVs perform a saturation attack mission, there are many factors that may affect combat effectiveness. However, it is not necessary to consider all the realistic factors as this can cause the model construction to be extremely complex. To allow us to concentrate on the main problem, several assumptions are proposed by simplifying the engagement model as follows:

1. In general, mission planning for UAV swarm operation can be broken down into two phases: arrive at the mission area from the base and then begin executing specific tasks.

In this paper, UAVs are assumed to have completed infiltration and have reached the perimeter of the target area. The next step is to carry out the attack task, which is the focus of this paper;

2. We assume that the flight of the UAVs can be restricted to a plane at a given altitude and that the UAVs avoid collisions by the stratification of the altitude. Furthermore, in this paper, the flight paths of the UAVs are simplified to straight lines, and each UAV flies at a constant velocity;
3. In the UAV swarm saturation attack model proposed in this paper, the vessels are treated as stationary targets. We present this assumption for two main reasons: firstly, the flight speeds of the fixed-wing UAVs are much greater than the vessel's speed, and the saturation attack is carried out in a very short period of time, so that the vessels' movement is insignificant; secondly, the UAVs attack the target vessels via anti-ship missiles, and the strike can be executed only if the target is within the missile's effective range. Thus, the slight movement of the vessels does not affect the effectiveness of the attack.

2.2. UAV Swarm Saturation Attack Model

2.2.1. Vehicles

For the combat scenario presented in this paper, heterogeneous fixed-wing UAVs are used. UAVs with different characteristics are assigned to different tasks in order to achieve optimal combat effectiveness.

Let

$$U = \{U_1, U_2, U_3 \ldots, U_{N_U}\} \tag{1}$$

Be the set of N_U heterogeneous fixed-wing UAVs. We assume that the UAV position information can be defined by $\left[x_{U_i}, y_{U_i}\right]$, which are the i-th UAV's horizontal coordinates in a Cartesian inertial reference frame. Besides, the UAV is flying at a constant velocity as mentioned above and we use vel_i to denote the constant speed of vehicle U_i. The attack capability of the UAV is another important feature in this combat scenario. The onboard weapons used to attack are limited resources and the UAVs with different load capacities can be mounted with different numbers of weapons. Thus, we use a_i to indicate the maximum number of attacks that the vehicle U_i can conduct. Furthermore, r_i is denoted as the effective range of the i-th UAV in this paper. A UAV cannot launch an attack on a target unless the target is in its effective range.

2.2.2. Targets and Tasks

Our paper focuses on the combat scenario of UAV swarm saturation attack against hostile surface ships. The enemy vessels are the targets of saturation attack. Let

$$V = \{V_1, V_2, V_3 \ldots, V_{N_V}\} \tag{2}$$

be the set of N_V target vessels. $\left[x_{V_j}, y_{V_j}\right]$ is denoted as the horizontal position of the j-th target and we indicate v_j as the j-th target's value. Different targets may have different values according to the defensive and combat capability, strategic significance, and so on. Our research mainly focuses on the target assignment problem and we do not discuss how to measure the values of targets in our paper. We use p_{ij} to represent the damage probability that the i-th UAV attacks the j-th target.

The task is another important concept in our paper. We regard one UAV attack on a target vessel as a single task implementation. Multiple attack tasks can be performed by a single drone and it is possible for a target to be attacked multiple times.

2.2.3. Combinatorial Optimization Problem

To solve the task assignment for a UAV swarm saturation attack, we formulate it as a combinatorial optimization problem. A solution S to the task assignment problem proposed in this paper adopts the form of a collection of ordered lists:

$$S = \{s_1, s_2, s_3 \ldots, s_{N_U}\}, \tag{3}$$

$$s_i = \{T_1^i, T_2^i, T_3^i, \ldots, T_{K_i}^i\}, \quad i = 1, 2, 3, \ldots N_U \tag{4}$$

where s_i is the ordered list of tasks allocated to the vehicle U_i, T_k^i is the k-th task that the vehicle U_i needs to perform, and K_i is the number of tasks allocated to the vehicle U_i.

We choose the sum of the expected damage value $E(S)$ for a valid solution S as the objective function to be maximized and use it to measure the combat effectiveness,

$$E(S) = \sum_{j=1}^{N_V} v_j \left[1 - \prod_{i=1}^{N_U} (1 - p_{ij})^{m_{ij}}\right] \tag{5}$$

where p_{ij} is the damage probability that the i-th UAV attacks the j-th target and m_{ij} is the number of tasks that are allocated to the vehicle U_i to attack the j-th target.

There are two constraints that must be satisfied in solving the problem. The first constraint

$$K_i \leq a_i \quad i = 1, 2, 3 \ldots N_U \tag{6}$$

refers to the number of tasks allocated to the vehicle U_i and cannot be more than its number of attacks a_i. The second constraint

$$D_i \leq vel_i time \quad i = 1, 2, 3 \ldots N_U, \tag{7}$$

$$D_i = \sum_{k=1}^{K_i} d_{ik} \quad i = 1, 2, 3 \ldots N_U, \tag{8}$$

$$d_{ik} = \max\left(\sqrt{\left(x_{U_i} - x_{V_{T_k^i}}\right)^2 + \left(y_{U_i} - y_{V_{T_k^i}}\right)^2} - r_i, 0\right) \quad i = 1, 2, 3 \ldots N_U, k = 1, 2, 3 \ldots K_i \tag{9}$$

is posed to ensure that the time consumed to perform all the tasks is within the predetermined time limit. Note that time is only consumed for movement in our model, not performing an attack task. In Equation (7), *time* is the predetermined time limit and D_i is the total flight distance of the vehicle U_i during the operation. As shown in Equation (8), we calculate D_i as the sum of the flight distance for each task performed by the vehicle U_i and d_{ik} is the flight distance of the i-th UAV during the implementation of the k-th task. As mentioned above, the flight paths of the UAVs are simplified into straight lines and each UAV flies at a constant velocity. Besides, the UAV can attack the target only if it is within its effective range. We use Equation (9) to calculate the flight distance of the UAVs.

Figure 1 presents an example of the task assignment for a UAV swarm saturation attack. The first row of the figure shows the combat situation and the corresponding task scheduling is in the following row. In the first stage, the drones have reached their predetermined positions and are ready to attack. Then, UAV_1 is assigned to attack $Target_2$ and UAV_2 is assigned to attack $Target_3$ in the second stage. In the third stage, UAV_1 is assigned to attack $Target_1$ and UAV_2 is assigned to attack $Target_3$ again. Note that the targets are within the effective range of the UAVs, so the UAVs do not need to move.

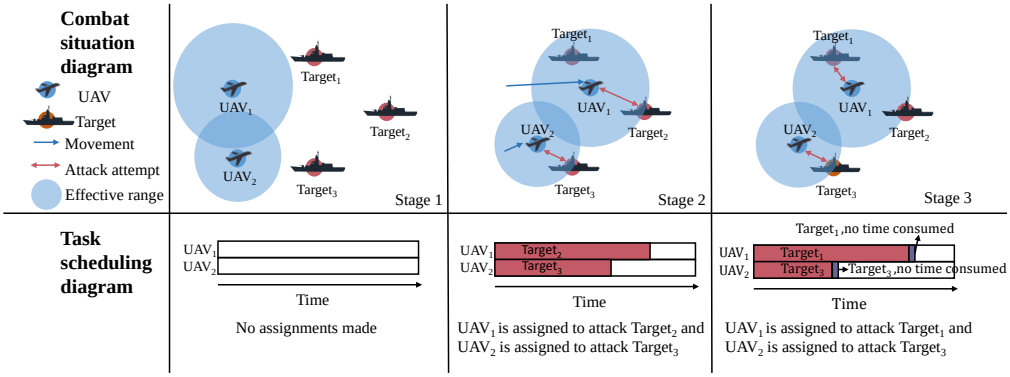


Figure 1. Example task assignment for UAV swarm saturation attack.

2.3. Markov Decision Process for Task Assignment

We consider the task assignment process from the reinforcement learning perspective and formulate this combinatorial optimization problem as a Markov Decision Process (MDP) [27]. Generally, a complete MDP can be represented by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$. In the tuple, $\mathcal{S}$ represents the set of possible states, $\mathcal{A}$ is the set of available actions, $\mathcal{P}$ indicates the probability of state transition, and $\mathcal{R}$ is the reward function. In our optimization problem, we do not need to use the transition probability, so we formulate this task assignment problem into an MDP using a triple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R} \rangle$ as follows.

State Space $\mathcal{S}$: The variables we use to represent state information are divided into three categories: UAV, Target, and UAV–Target pair. The current position, velocity, number of available attacks, effective range, and available time are used to indicate the states of UAVs. The current position $\left[x_{U_i}^t, y_{U_i}^t\right]$ are the current x and y coordinates of the i-th UAV at stage t and will be changed during the decision process. The number of available attacks a_i^t of the i-th UAV at stage t will be reduced by one after each execution of the task. The available time $time_i^t$ represents the current time left to the time limit of the i-th UAV at stage t. Besides, the velocity vel_i and effective range r_i are two constant variables. We use the position and expected remaining value to indicate the state of the target. As assumed above, the position $\left[x_{V_j}, y_{V_j}\right]$ of the j-th target is considered constant. The expected remaining value val_j^t of the j-th target at stage t will be reduced as the process of decision making moves forward and we use the value of the j-th target v_j as the initial expected remaining value val_j^0. The expected remaining value can be calculated as follows:

$$val_j^t = val_j^{t-1}(1 - p_{ij}). \tag{10}$$

The variables we use to represent the state of the UAV–Target pair are damage probability, time cost, and infeasibility flag. We use p_{ij} to indicate the damage probability of the i-th UAV attack on the j-th target. The time cost tc_{ij}^t is the time required by the i-th UAV to complete the attack task against the j-th target at stage t and can be formulated as follows:

$$tc_{ij}^t = \max\left(\sqrt{\left(x_{U_i}^t - x_{V_j}\right)^2 + \left(y_{V_i}^t - y_{V_j}\right)^2} - r_i, 0\right) \Big/ vel_i. \tag{11}$$

The state representation also includes the infeasibility flag inf_{ij}^t, which represents whether the action of UAV$_i$-Vessel$_j$ pair would violate the constraints. inf_{ij}^t is equal to 0 if it would not violate any constraints, and is 1 otherwise. Our decision process will not stop until the infeasibility flag for all the UAV–Target pairs are assigned the value 1. All the variables we use to represent the state information are listed in Table 1. The variables marked with an * are dynamic and may change in the decision proces.

Table 1. Variables used to represent the state information.

$\mathbf{UAV}_i$	$\mathbf{Target}_j$	$\mathbf{UAV}_i - \mathbf{Target}_j$
Current position* $\left[x_{U_i}^t, y_{U_i}^t\right]$	Position $\left[x_{V_j}, y_{V_j}\right]$	Damage probability p_{ij}
Velocity vel_i	Expected remaining value* val_j^t	Time cost* tc_{ij}^t
Number of available attacks* a_i^t		Infeasibility flag* inf_{ij}^t
Effective range r_i		
Available time* $time_i^t$		

Action Space $\mathcal{A}$: We use the UAV_i-Vessel_j pair to denote the action for each step in the decision process. The UAV_i-Vessel_j pair means to ask the i-th UAV to perform an attack task against the j-th target. Only the feasible UAV_i-Vessel_j pair can be used as the action and the UAV_i-Vessel_j pair with an infeasibility flag will not be in the action space.

Reward $\mathcal{R}$: In the UAV swarm saturation attack model proposed in this paper, we use the sum of the expected damage value in Equation (5) to measure the combat effectiveness. However, it is not appropriate to assign part of the combat effectiveness to each action, because we cannot obtain the final combat effectiveness until the decision process is over. Inspired by the concept of "marginal return" proposed in [28], we construct a marginal-return-based reward function and denote the additional combat effectiveness caused by the action as the reward:

$$r_t = E(S_t) - E(S_{t-1}). \tag{12}$$

3. Proposed Method

In this section, a reinforcement learning agent is proposed for the task assignment problem. We construct the decision agent using a deep neural network. The deep neural network we proposed is a stack-based architecture that uses attention layers to pass information. Then, in order to obtain a well-trained neural network, we resort to policy gradient methods to train the policy neural network.

3.1. Network Architecture for Task Assignment

We parameterize our reinforcement learning agent using a deep neural network architecture. Based on the MDP model we have constructed above, the input to the deep neural network is the state of each process and the output is the action. Inspired by the Transformer architecture [20,29], the deep neural network we proposed in this paper is a stack-based architecture that uses attention layers to pass information between different potential assignments. In our network architecture, there is an initial embedding layer that we use to encode the state matrix. Following this, a set of "stacks" with the same structure but different parameters implement similar operations. Each stack consists of two communication layers: Agent-wise Communication Layer and Task-wise Communication Layer. In order to gain a more holistic understanding of the current situation, we use these stacks to pass information between different agents and tasks. Then, we use a linear feed-forward layer to transform the hidden representations of each UAV–Target pair into a scalar. Finally, a Softmax layer is used to process the final results and we obtain the output. The full network architecture for the task assignment is shown in Figure 2.

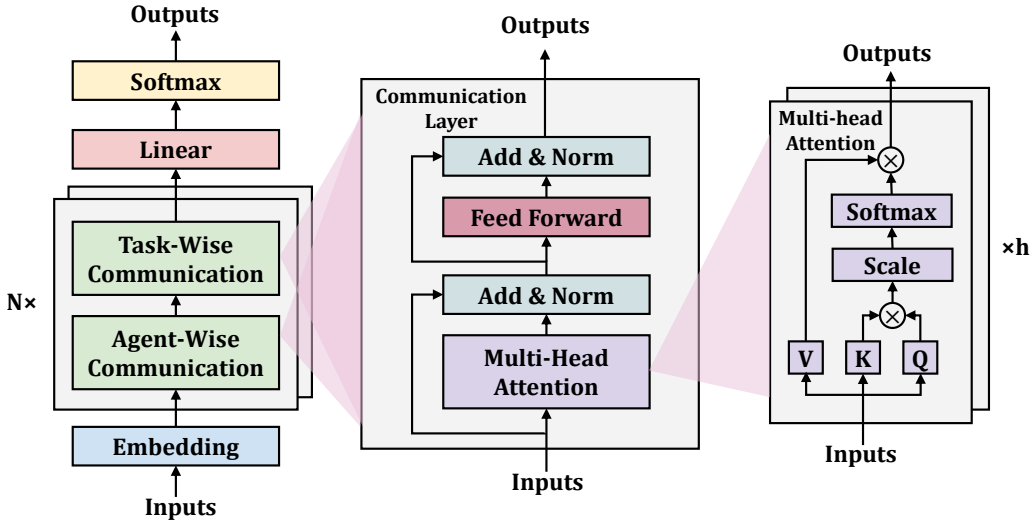


Figure 2. Network architecture for task assignment.

Input: As in the MDP model we constructed above, we use the state of each process as the input to the deep neural network. A three-dimensional array $X \in \mathbb{R}^{N_U \times N_V \times |X|}$ is used in our paper to represent the state information, where $|X|$ is the total number of the variables we use to indicate the state information. In the three-dimensional array, we can index X through i and j to view information about the specific UAV–Target pairs:

$$X_{i,j} = \begin{bmatrix} \text{UAV}_i & \text{Target}_j & \text{UAV}_i - \text{Target}_j \end{bmatrix}^\top. \tag{13}$$

UAV_i represents the i-th UAV's properties, Target_j indicates the state information of the j-th target, and $\text{UAV}_i - \text{Target}_j$ corresponds to the UAV–Target pairwise state information.

Embedding Layer: The input X is first processed by the embedding layer $\mathcal{E}$, which is a feed-forward neural network. We use the embedding layer to project the last dimension of the input X into the dimension $\mathbb{R}^{|\mathcal{E}|}$. In other words, every X_{ij} is processed through a a feed-forward layer element-wise with parameter $\theta_\mathcal{E}$ as follows:

$$\mathcal{E}(X) = \begin{bmatrix} F(X_{1,1};\theta_\mathcal{E}) & \dots & F(X_{1,N_V};\theta_\mathcal{E}) \\ \vdots & \ddots & \vdots \\ F(X_{N_U,1};\theta_\mathcal{E}) & \dots & F(X_{N_U,N_V};\theta_\mathcal{E}) \end{bmatrix} \tag{14}$$

Communication Layer: Inspired by the Transformer architecture [20], we construct the communication layer in our paper. Each communication layer is composed of two sublayers: a multi-head attention layer and a feed-forward layer. We also add skip–connection [30] and batch normalization [31] for each sublayer.

We utilize the attention mechanism [20] as a message passing method between different UAVs or targets in our task assignment problem. There are two kinds of communication layers: Agent-wise communication and Task-wise communication. The two kinds of communication layers perform the same operations, only the dimensions they operate on are different. Here, we will illustrate the Task-wise communication as an example; the other one is the same. For the embedded input $\mathcal{E}(X) \in \mathbb{R}^{N_U \times N_V \times |\mathcal{E}|}$, which is a three-dimensional array, we can divide it into M groups according to different UAVs. Each group can be represented as a two dimensional vector $Y \in \mathbb{R}^{N_V \times |\mathcal{E}|}$. Our network architecture will apply a self-attention operation on each group separately using the same parameters. Self-attention performs three feed-forward layers separately to generate queries $q \in \mathbb{R}^{N_V \times d_k}$,

keys $k \in \mathbb{R}^{N_V \times d_k}$, and values $v \in \mathbb{R}^{N_V \times d_v}$ first. Then, we operate the attention mechanism as follows:

$$\text{attention}(q, k, v) = \text{softmax}\left(\frac{qk^\top}{\sqrt{d_k}}\right)v. \tag{15}$$

We utilize multi-head attention to operate the multiple attention mechanism in parallel. In our paper, we use h heads: $q = (q_1, q_2, \ldots, q_h)$, $k = (k_1, k_2, \ldots, k_h)$, and $v = (v_1, v_2, \ldots, v_h)$. Then, the whole self-attention heads will be concatenated together:

$$\text{MHA}(Y) = [H_1 H_2 \ldots H_h] \tag{16}$$

where

$$H_l = \text{attention}(q_l, k_l, v_l) \tag{17}$$

Thus, $H_l \in \mathbb{R}^{N_V \times d_v}$ and $\text{MHA}(Y) \in \mathbb{R}^{N_V \times h d_v}$.

After the multi-head attention sublayer, we then use a fully connected feed-forward (FF) layer to process the data. Furthermore, we also add a skip-connection and batch normalization (BN) for each sublayer:

$$Z = \text{BN}(\mathcal{E} + \text{MHA}(\mathcal{E})), \tag{18}$$

$$C(Z) = \text{BN}(Z + \text{FF}(Z)) \tag{19}$$

where $\mathcal{E}$ is the result of the embedding layer and C is the result of the communication layer.

Linear and Softmax Layer: After the operation of N stacks, where each stack performs the self-attention operation, we construct a Linear layer. The Linear layer we use is simply a fully connected feed-forward layer. We use the Linear layer to further process the state information. In the end, we use the Softmax layer to process the final result. However, before the Softmax operation, the infeasible "UAV–Task" pairs need to be masked out. We perform the mask out operation on the temporary output $M \in \mathbb{R}^{N_U \times N_V}$:

$$\overline{M}_{i,j} = M_{i,j} - \beta inf_{i,j} \tag{20}$$

where β is a very large positive constant. After the mask operation, we use Softmax layer to obtain the final result.

3.2. Optimization with Policy Gradients

In our paper, we resort to policy gradient methods to train the policy neural network. Our goal is to find an optimal policy and maximize the expected return of this policy in the environment. We define the objective function of policy learning as follows:

$$J(\theta) = E_{s_0}[V^{\pi_\theta}(s_0)]. \tag{21}$$

Here, s_0 denotes the initial state and V^{π_θ} indicates the state–value function that represents the expected return starting from the state following the policy π_θ. We derive the objective function with respect to θ. The gradient ascent method can then be used to maximize this objective function to obtain the optimal policy. The gradient of the objective function is formulated as follows:

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[Q^{\pi_\theta}(s, a) \nabla_\theta \log \pi_\theta(a \mid s)] \tag{22}$$

where Q^{π_θ} represents the action–value function, which indicates the expected return from performing the action on the current state when the MDP follows the policy π_θ.

We use the REINFORCE algorithm [32] to formulate the gradient of the objective function. The REINFORCE algorithm utilizes the Monte Carlo method to sample trajec-

tories and estimate $Q^{\pi_\theta}(s, a)$. Considering the large variance of the gradient estimates of the original REINFORCE algorithm, we introduce the baseline function in the training algorithm to reduce the variance of the gradients and therefore increase the learning speed:

$$\nabla_\theta J(\theta) = E_{\pi_\theta}\left[\sum_{t=0}^{T-1}\left(\sum_{k=t+1}^{T}\gamma^{k-t-1}r_k - b(s_t)\right)\nabla_\theta \log \pi_\theta(a_t \mid s_t)\right] \tag{23}$$

where $b(s_t)$ indicates the baseline function that does not depend on the action. In our paper, we utilize a parametric baseline function to estimate the expected return. An auxiliary network, called a critic, is introduced in the training period to accelerate the learning. The critic network $v(s_t, \boldsymbol{w})$ parameterized by $\boldsymbol{w}$ has the similar architecture with the policy network but the end of the network is a little different, which the output of the critic network is a scalar estimating the expected return. We train the critic baseline function with a stochastic gradient descent method on a mean squared error objective between the expected return sampled by the most recent policy and its predictions $v(s_t, \boldsymbol{w})$. The pseudocode of the training algorithm is provided in Algorithm 1.

Algorithm 1: REINFORCE with critic baseline

Initialize: the max number of episodes E, policy network with random parameters $\pi(a|s, \boldsymbol{\theta})$, critic network with random parameters $v(s, \boldsymbol{w})$, learning rate $lr^{\boldsymbol{\theta}}$, $lr^{\boldsymbol{w}}$;
for $e = 1$ *to* E **do**
  Reset the training environment;
  Generate an episode $s_0, a_0, r_1, \ldots, s_{T-1}, a_{T-1}, r_T$,following $\pi(a|s, \boldsymbol{\theta})$;
  for $t = 0$ *to* $T - 1$ **do**
    $G \leftarrow \sum_{k=t+1}^{T}\gamma^{k-t-1}r_k$;
    $\delta \leftarrow G - v(s_t, \boldsymbol{w})$;
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + lr^{\boldsymbol{w}}\delta\nabla v(s_t, \boldsymbol{w})$;
    $\theta \leftarrow \theta + lr^{\boldsymbol{\theta}}\gamma^t\delta\nabla \ln \pi(a_t|s_t, \boldsymbol{\theta})$;
  end
end

3.3. Searching Strategy

After the training process, the well-trained policy network can be used to infer the optimal task assignment scheme. In the inference period, we consider two searching strategies to construct the final assignment scheme: greedy search and sampling.

Greedy Search: Our first approach is simply select the UAV–Target pair with the largest probability in each process. Given the initial state representing a task assignment problem, the well-trained policy network adopts the state as the input and outputs the probability distribution over different UAV–agent pairs. The greedy search strategy select the index with the largest probability greedily. Then, the next state information is fed into the decision agent and the above process is repeated until the end of the allocation.

Sampling: Considering constructing a task assignment scheme using our reinforcement learning agent is inexpensive, so we utilize the sampling approach to sample multiple candidate solutions from the policy network and select the best one. Our sampling strategy obtains each solution based on the action probability distribution from the output of the policy neural network. More candidate solutions may produce a better result, but it will also consume more time. We can choose a proper scale of the candidates to achieve a balance between performance and efficiency.

4. Simulation and Analysis

In this section, we validate the performance of our proposed method for a task assignment problem. First, we introduce the detailed settings about the generation of the problem

instances. Then, we show the training process of the policy network and in different scenarios, and the hyperparameters of the algorithm are also introduced. After that, we compare our deep reinforcement learning algorithm with other algorithms in terms of the solution quality. Finally, we perform a cost-effectiveness analysis on different number of UAVs using our algorithm.

4.1. Setting Up

Before conducting simulation and analysis, the detailed settings of our experiments will be first introduced. In our simulated experiment environment, each problem instance will construct multiple drones and hostile targets. The UAVs will be assigned different attack tasks to destroy the enemy targets according to a certain task assignment policy. Furthermore, multiple problem instances of different scales will be tested to analyze the performance of the task assignment policy based on different algorithms. All the simulation experiments are carried out on a PC with an AMD 3.8 GHz CPU, 16 GB internal memory, GeForce RTX 3070 GPU, and Windows 10 operating system. Furthermore, the simulation computer programs that we use in our experiments are developed based on Python 3.10.

Our paper focuses on the UAV swarm saturation attack operation against enemy ships and we construct three different scales of vessel formations in our experiments. There are 3, 6, and 10 targets in these three formations, respectively. These targets are distributed in an area of 30 × 30 km and different targets may have different values. We list all the positions and value information of the targets in Table 2. Here, we construct the three target formations to conduct the experiments, but our model is certainly applicable to other target distributions as well.

Table 2. Position and value information of the target formations.

		Target 1	**Target 2**	**Target 3**	**Target 4**	**Target 5**	**Target 6**	**Target 7**	**Target 8**	**Target 9**	**Target 10**
Formation I	**Position/km**	(10, 15)	(20, 20)	(20, 10)							
	Value	50	50	50							
Formation I	**Position/km**	(0, 15)	(10, 25)	(10, 5)	(20, 15)	(30, 25)	(30, 5)				
	Value	50	50	50	100	50	50				
Formation I	**Position/km**	(0, 20)	(0, 10)	(10, 30)	(10, 0)	(15, 20)	(15, 10)	(20, 30)	(20, 0)	(30, 20)	(30, 10)
	Value	50	50	50	50	100	100	50	50	50	50

According to the capabilities, the fixed-wing UAVs used in our experiments can be divided into three types, which are denoted by Type I, Type II, and Type III. Different types of UAVs vary in terms of velocity, number of attacks, and effective range. The performance information of the drones is listed in Table 3. We will apply different scales of drones in the task assignment problem. The amount of Type I and Type II are both set as 2 M/5 and the amount of Type III is set as M/5 given the M UAVs. In our experiment, we sample the damage probability for any UAV–Target pair uniformly from [0.1, 0.3]. Meanwhile, we also assume that the UAVs are randomly distributed in the area of 30 × 30 km next to the target area before the execution of tasks. In order to obtain better saturation attack effectiveness, we limit all attack tasks to be completed in a relatively short period of time and set the predetermined time limit to be 0.1 h.

Table 3. Capabilities of the three types of UAVs.

	Velocity(km/h)	**Number of Attacks**	**Effective Range (km)**
Type I	400	1	15
Type II	500	2	20
Type III	600	3	25

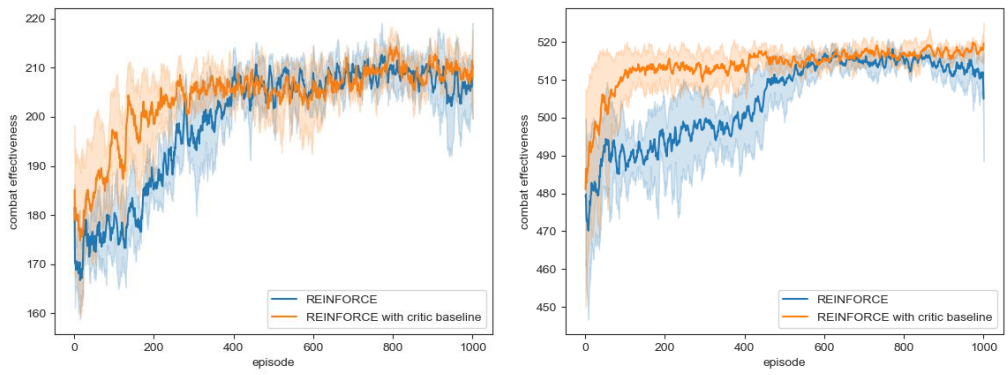
4.2. Convergence of Deep Reinforcement Learning

In this section, we will show the training process of the policy network and the convergency of the deep reinforcement learning in different scenarios. In order to illustrate the details of our algorithm, all the hyperparameters used in our experiments are listed in Table 4.

Table 4. Hyperparameters used in the experiments.

Hyperparameters of Network Architecture		Hyperparameters of Training	
Hidden size	128	**Training episodes**	1000
Number of stacks	2	**Learning rate**	1×10^{-5}
Attention heads	4	**Discount rate**	0.98
Initializer	Random normal	**Optimizer**	Adam
Activation	ReLU		

In order to clearly demonstrate the convergence performance of our deep reinforcement learning algorithm, we train the policy network with the settings in Table 4. Here, the algorithm is, respectively, trained in two problem scales: 10 UAVs–10 targets and 50 UAVs–10 targets. Furthermore, we train the neural network in both two methods: REINFORCE and REINFORCE with a critic baseline. Each method is trained five times with 1000 episodes in each problem scale. To visualize how the algorithm performs, we plot the learning curves that represent the training process, respectively, in Figure 3. We use dark lines to represent the mean combat effectiveness and the light area to represent the 95% confidence interval. As shown in Figure 3, it is clearly illustrated that the REINFORCE with the critic method performs better and improves the convergence speed and quality in both problem scales. In Figure 3a, the learning curve of REINFORCE with the critic baseline converges faster and arrives at the highest level (nearly 210) in approximately 300 episodes. In comparison, the REINFORCE method does not converge to the highest level until 400 episodes. In a larger problem scale, the advantage of REINFORCE with a critic baseline method will be more obvious. In Figure 3b, the curve of REINFORCE with the critic baseline reaches the highest level (nearly 515) in about 150 episodes. Furthermore, we can see that the confidence interval is more narrow, which indicates that the policy network trained has more stable performance. By contrast, the curve of REINFORCE method does not converge until 500 episodes.



(a) 10UAVs-10Targets (b) 50UAVs-10Targets

Figure 3. Curves of combat effectiveness with respect to training episodes.

4.3. Performance Comparison on Solution Quality

We compare the solutions produced using our deep reinforcement learning framework with those found using the Genetic Algorithm (GA) and the Random Selection Algorithm (RSA). In the comparison experiment, the algorithms based on our deep reinforcement learning framework utilize two types of searching strategies: greedy search and sampling. We have already introduced the two searching strategies above and present no further

description here. Considering the balance between the performance and efficiency, there are two sampling agents used in our experiment: sampling 5 candidates and sampling 10 candidates. The genetic algorithm is an adaptive heuristic search algorithm based on the ideas of natural selection and genetics. It is commonly used to generate high-quality solutions for optimization problems and search problems. Shima et al. proposed genetic algorithms to solve the multiple task assignment problem for cooperating UAVs in [3]. Jia et al. proposed a modified genetic algorithm to solve the cooperative multiple task assignment problem with stochastic velocities and time windows in [9]. For the task assignment problem, we utilize the GA to generate solutions as individuals and improve the solution by simulating the evolution of population. We also utilize the random selection algorithm for comparison. The RSA used in our experiment employs the same decision framework with our deep reinforcement learning method. The difference is that the task assignment decision is not performed by the policy network but a random selector.

Table 5. Performance and running time of the algorithms.

		RL_Greedy	RL_Sampling (5)	RL_Sampling (10)	GA	RSA
50 UAVs-10Targets	**Performance**	520.86 ± 4.26	526.85 ± 4.72	**528.16±3.97**	514.92 ± 9.06	502.76 ± 10.20
	Time	1.09 s	5.37 s	10.75 s	5.47 s	4.60 ms
20 UAVs–10Targets	**Performance**	338.80 ± 10.47	345.22 ± 6.86	**352.08 ± 5.80**	332.87 ± 11.61	304.90 ± 22.17
	Time	0.25 s	1.22 s	2.42 s	2.24 s	2.00 ms
10 UAVs–10Targets	**Performance**	208.15 ± 11.74	222.23 ± 6.95	**226.61 ± 7.79**	206.88 ± 10.84	184.98 ± 23.61
	Time	0.11 s	0.44 s	0.88 s	1.15 s	1.00 ms
50 UAVs–6Targets	**Performance**	334.42 ± 1.64	337.81 ± 1.21	**338.12 ± 1.34**	336.25 ± 2.16	332.84 ± 5.03
	Time	0.76 s	3.76 s	7.50 s	4.31 s	3.50 ms
20 UAVs–6Targets	**Performance**	257.77 ± 5.58	264.86 ± 4.97	**267.68 ± 4.43**	260.52 ± 4.53	245.58 ± 9.91
	Time	0.18 s	0.92 s	1.86 s	1.73 s	1.40 ms
10 UAVs–6Targets	**Performance**	177.31 ± 10.84	182.58 ± 7.23	**184.48 ± 6.74**	183.37 ± 8.92	164.26 ± 14.00
	Time	0.08 s	0.37 s	0.74 s	0.89 s	1.00 ms
30 UAVs–3Targets	**Performance**	146.27 ± 0.73	147.75 ± 0.30	**147.77 ± 0.33**	147.62 ± 0.48	146.55 ± 1.21
	Time	0.27 s	1.32 s	2.63 s	2.15 s	1.55 ms
20 UAVs–3Targets	**Performance**	139.15 ± 1.89	139.82 ± 1.33	140.54 ± 1.38	**140.71 ± 1.02**	136.66 ± 2.72
	Time	0.17 s	0.76 s	1.55 s	1.45 s	1.10 ms
10 UAVs–3Targets	**Performance**	110.72 ± 3.68	112.17 ± 3.21	112.61 ± 3.03	**113.54 ± 4.34**	106.91 ± 7.93
	Time	0.08 s	0.31 s	0.64 s	0.73 s	1.00 ms

In the comparative experiment, we use the combat effectiveness $E(S)$ in Equation (5) as the metric to measure the performance of the algorithms. In order to ensure the reliability of the experiment results, we conduct the experiment on multiple problem instances with different problem scales. Here, we introduce nine problem scales: 50 UAVs–10 Targets, 20 UAVs–10 Targets, 10 UAVs–10 Targets, 50 UAVs–6 Targets, 20 UAVs–6 Targets, 10 UAVs–6 Targets, 30 UAVs–3 Targets, 20 UAVs–3 Targets, and 10 UAVs–3 Targets. Furthermore, we run 20 instances of different problem scales for each algorithm. The average performance and running time of the algorithms are listed in Table 5 and the optimal result of each problem scale is marked in bold for significance. We also present the experimental results in Figure 4.

As shown in the experimental results, GA achieves relatively good performance in the instances of small problem scales and obtains the optimal results in two scales: 10 UAVs–3 Targets and 20 UAVS–3 Targets. GA is the intelligent exploitation of a random search provided with historical data to direct the search into the region of better performance in solution space. When facing small-scale problems, GA only needs to search a relative small dimension of solution space and it is easier to search a high-quality solution. However, as the problem becomes more complex, the dimension of the solution space increases rapidly and the GA can hardly search a good solution. In our experiment, we set the maximum of generation as 200 to ensure that a satisfactory solution can be obtained. As shown in Table 5, we can see that it costs much time for the GA to search the solution space, thus resulting in low time efficiency.

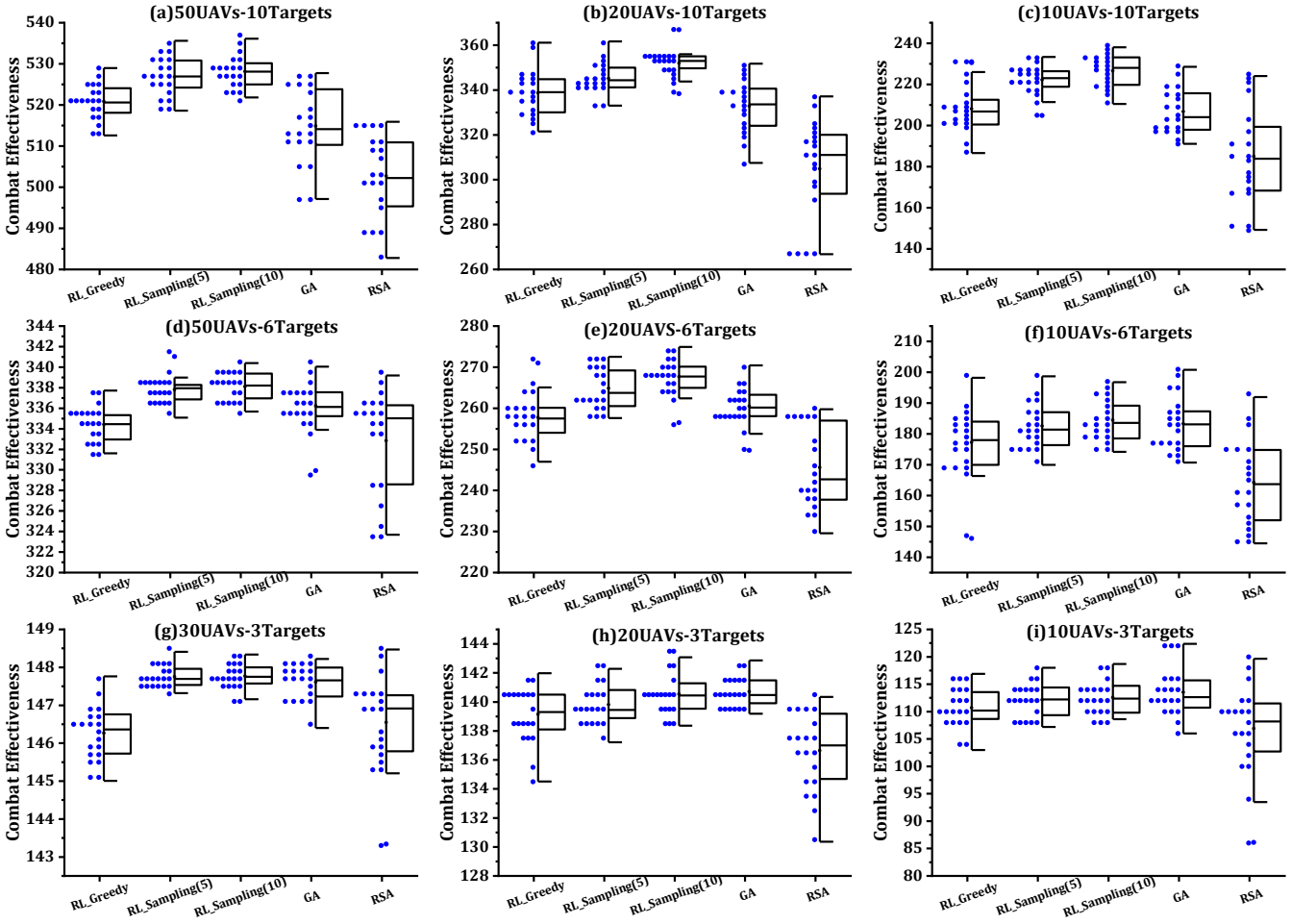


Figure 4. Performance comparison on solution quality.

By comparison, the algorithms based on our deep reinforcement learning framework show superiority in most problem instances, especially in those relatively large problem scales. For example, the average combat effectiveness obtained by the reinforcement learning with sampling strategy outperforms that of other algorithms by 2.6% in a 50 UAVs–10 Targets problem scale and 5.8% in a 20 UAVs–10 Targets problem scale. Besides, we can see that the standard deviation of the combat effectiveness obtained by the RL based algorithms are much smaller, which means that the performance is more stable. Moreover, the performance of our RL algorithm is also pretty close to other comparative algorithms in small problem scales. Our policy neural network uses attention layers to pass information between different potential assignments and has a more comprehensive understanding of the task assignment problem. Thus, our reinforcement learning agent is able to make better task assignment decisions. On the other hand, our RL algorithm has a clear advantage in time efficiency. Reinforcement learning with a greedy search algorithm obtains satisfactory solutions with the shortest time in all the problem scales. As shown in Table 5, we can see that the reinforcement learning with a greedy search strategy is around 80% faster than the genetic algorithm in the 50 UAVs–10 Targets problem scale and around 90% faster in the 10 UAVs–3 Targets problem scale. This characteristic ensures that our algorithm can satisfy the real-time requirement of the task assignment problem. When facing specific problems, we can also use the sampling strategy and choose a proper scale of the candidates to achieve a balance between the performance and efficiency. Besides, the policy neural network used in our experiments is only trained with the 10 UAVs–3 Targets problem instances to reduce the training time. However, it can also perform well in other problem scales. It means

that the well-trained policy network can generalize well to different problem scales, which represents better applicability and flexibility in practical scenarios.

4.4. Cost-Effectiveness Analysis on Different Number of UAVs

In real combat scenarios, various weapon resources are often limited and expensive. Besides the pursuit of greater combat effectiveness, we should also pay attention to the cost of weapon resources. We should choose the appropriate amount of UAVs to perform the task and achieve maximum combat effectiveness under limited cost. In this section, our model and algorithm are applied to the analysis of the number of UAVs for a saturation attack. We introduce the cost of the vehicles and extend the model to solve the multi-objective task assignment problem. In addition to the combat effectiveness, we also include the cost in the objective function:

$$B(S, U) = E(S|U) - C(U) \tag{24}$$

where U is the set of UAVs used in the task, $C(U)$ is the total cost of the UAVs and $B(S, U)$, here, is named as the combat benefit. In our experiment, we set the cost of a Type I UAV as 1, the cost of a Type II as 2, and the cost of a Type III as 3. The cost-effectiveness analysis on different numbers of UAVs for the three vessel formations are performed here; we run our deep reinforcement learning algorithm 20 times in each instance.

Figure 5 shows the combat benefit with respect to the number of UAVs for different vessel formations. As shown in Figure 5, all curves rise and then fall for the three formations. The combat benefit reaches its highest point when the number of UAVs is about 20 for the 3 targets formation, 40 UAVs for the 6 targets formation, and 60 UAVs for the 10 targets formation. The combat benefit gradually increases when the number of UAVs is small. Furthermore, after the number of UAVs exceeds the maximum point, the combat benefit starts to decline, which means that the number of drones is redundant and the benefits of adding vehicles do not cover the costs.

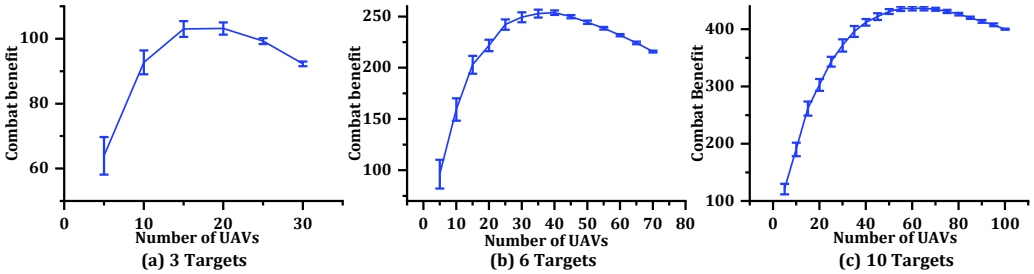


Figure 5. Cost-effectiveness analysis on different number of UAVs.

5. Conclusions

This paper studies the task assignment problem for a UAV swarm saturation attack. First, we construct the mathematical programming model of the task assignment problem and formulate it as a combinatorial optimization problem. Additionally, we develop the task assignment model as a Markov Decision Process from the sequential decision-making perspective. Then, we propose a deep reinforcement learning framework to solve the task assignment problem. A decision agent based on the attention mechanism is developed and we resort to the policy gradient methods to train the policy neural network. The simulation and experimental results have shown that our deep reinforcement learning method can generate high-quality solutions for different problem scales, which meets the requirements of real-time and flexibility in the actual situation. In the future, we will construct a more detailed mathematical model to better reflect the real combat environment. Furthermore, we will extend our algorithm to decentralized decision scenarios in order to adapt to the situation of limited communication.

Author Contributions: Conceptualization, K.S.; methodology, F.Q.; writing—original draft preparation, F.Q.; writing—review and editing, K.S., X.L. and K.Z.; supervision, X.L. 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 (NSFC) under Grant No. 61802425 and National Social Science Foundation of China under 18BGL285, 19CGL073 and 2021-SKJJ-C-017.

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. Otto, R.P. *Small Unmanned Aircraft Systems (SUAS) Flight Plan: 2016–2036. Bridging the Gap between Tactical and Strategic*; Technical report; Air Force Deputy Chief of Staff: Washington, DC, USA, 2016.
2. Deng, Q.; Yu, J.; Wang, N. Cooperative task assignment of multiple heterogeneous unmanned aerial vehicles using a modified genetic algorithm with multi-type genes. *Chin. J. Aeronaut.* **2013**, *26*, 1238–1250. [CrossRef]
3. Shima, T.; Rasmussen, S.J.; Sparks, A.G.; Passino, K.M. Multiple task assignments for cooperating uninhabited aerial vehicles using genetic algorithms. *Comput. Oper. Res.* **2006**, *33*, 3252–3269. [CrossRef]
4. Darrah, M.; Niland, W.; Stolarik, B. *Multiple UAV Dynamic Task Allocation Using Mixed Integer Linear Programming in a SEAD Mission*; Infotech@ Aerospace: Arlington, VA, USA, 2005; p. 7164.
5. Schumacher, C.; Chandler, P.; Pachter, M.; Pachter, L. Constrained optimization for UAV task assignment. In Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit, Monterey, CA, USA, 5–8 August 2004; p. 5352.
6. Nygard, K.E.; Chandler, P.R.; Pachter, M. Dynamic network flow optimization models for air vehicle resource allocation. In Proceedings of the IEEE 2001 American Control Conference.(Cat. No. 01CH37148), Arlington, VA, USA, 25–27 June 2001; Volume 3, pp. 1853–1858.
7. Ye, F.; Chen, J.; Sun, Q.; Tian, Y.; Jiang, T. Decentralized task allocation for heterogeneous multi-UAV system with task coupling constraints. *J. Supercomput.* **2021**, *77*, 111–132. [CrossRef]
8. Shima, T.; Rasmussen, S.J.; Sparks, A.G. UAV cooperative multiple task assignments using genetic algorithms. In Proceedings of the IEEE 2005, American Control Conference, Portland, OR, USA, 8–10 June 2005; pp. 2989–2994.
9. Jia, Z.; Yu, J.; Ai, X.; Xu, X.; Yang, D. Cooperative multiple task assignment problem with stochastic velocities and time windows for heterogeneous unmanned aerial vehicles using a genetic algorithm. *Aerosp. Sci. Technol.* **2018**, *76*, 112–125. [CrossRef]
10. Zhu, Z.; Tang, B.; Yuan, J. Multirobot task allocation based on an improved particle swarm optimization approach. *Int. J. Adv. Robot. Syst.* **2017**, *14*, 1729881417710312. [CrossRef]
11. Zhao, W.; Meng, Q.; Chung, P.W. A heuristic distributed task allocation method for multivehicle multitask problems and its application to search and rescue scenario. *IEEE Trans. Cybern.* **2015**, *46*, 902–915. [CrossRef] [PubMed]
12. Zhen, Z.; Xing, D.; Gao, C. Cooperative search-attack mission planning for multi-UAV based on intelligent self-organized algorithm. *Aerosp. Sci. Technol.* **2018**, *76*, 402–411. [CrossRef]
13. Fan, C.; Han, S.; Li, X.; Zhang, T.; Yuan, Y. A modified nature-inspired meta-heuristic methodology for heterogeneous unmanned aerial vehicle system task assignment problem. *Soft Comput.* **2021**, *25*, 14227–14243. [CrossRef]
14. Xia, C.; Yongtai, L.; Liyuan, Y.; Lijie, Q. Cooperative task assignment and track planning for multi-UAV attack mobile targets. *J. Intell. Robot. Syst.* **2020**, *100*, 1383–1400. [CrossRef]
15. Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. Mastering the game of go without human knowledge. *Nature* **2017**, *550*, 354–359. [CrossRef]
16. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
17. Vinyals, O.; Fortunato, M.; Jaitly, N. Pointer networks. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 3134.
18. Bello, I.; Pham, H.; Le, Q.V.; Norouzi, M.; Bengio, S. Neural combinatorial optimization with reinforcement learning. *arXiv* **2016**, arXiv:1611.09940.
19. Kool, W.; Van Hoof, H.; Welling, M. Attention, learn to solve routing problems! *arXiv* **2018**, arXiv:1803.08475.
20. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 1–11.
21. Zhao, X.; Zong, Q.; Tian, B.; Zhang, B.; You, M. Fast task allocation for heterogeneous unmanned aerial vehicles through reinforcement learning. *Aerosp. Sci. Technol.* **2019**, *92*, 588–594. [CrossRef]
22. Tian, Y.T.; Yang, M.; Qi, X.Y.; Yang, Y.M. Multi-robot task allocation for fire-disaster response based on reinforcement learning. In Proceedings of the IEEE 2009 International Conference on Machine Learning and Cybernetics, Hebei, China, 12–15 July 2009; Volume 4, pp. 2312–2317.

23. Yang, H.; Xie, X.; Kadoch, M. Intelligent resource management based on reinforcement learning for ultra-reliable and low-latency IoV communication networks. *IEEE Trans. Veh. Technol.* **2019**, *68*, 4157–4169. [CrossRef]
24. Luo, W.; Lü, J.; Liu, K.; Chen, L. Learning-based policy optimization for adversarial missile-target assignment. *IEEE Trans. Syst. Man, Cybern. Syst.* **2021**, *52*, 4426–4437. [CrossRef]
25. Liang, X.; Du, X.; Wang, G.; Han, Z. A deep reinforcement learning network for traffic light cycle control. *IEEE Trans. Veh. Technol.* **2019**, *68*, 1243–1253. [CrossRef]
26. Huang, L.; Bi, S.; Zhang, Y.J.A. Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks. *IEEE Trans. Mob. Comput.* **2019**, *19*, 2581–2593. [CrossRef]
27. Sutton, R.S.; Barto, A.G. *Introduction to Reinforcement Learning*; MIT Press: Cambridge, UK, 1998; Volume 135.
28. Xin, B.; Wang, Y.; Chen, J. An efficient marginal-return-based constructive heuristic to solve the sensor–Weapon–Target assignment problem. *IEEE Trans. Syst. Man, Cybern. Syst.* **2018**, *49*, 2536–2547. [CrossRef]
29. Gibbons, D.; Lim, C.C.; Shi, P. Deep learning for bipartite assignment problems. In Proceedings of the 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), Bari, Italy, 6–9 October 2019; pp. 2318–2325.
30. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 770–778.
31. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning (PMLR), Lille, France, 6–11 July 2015; pp. 448–456.
32. Williams, R.J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Reinf. Learn.* **1992**, *8*, 5–32.

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.