




Article

Dynamic Target Assignment by Unmanned Surface Vehicles Based on Reinforcement Learning

Tao Hu [†] , Xiaoxue Zhang ^{*,†} , Xueshan Luo and Tao Chen 

National Key Laboratory of Information Systems Engineering, National University of Defense Technology, Changsha 410073, China; ht_nudt@nudt.edu.cn (T.H.); xsluo@nudt.edu.cn (X.L.); chentao@nudt.edu.cn (T.C.)

* Correspondence: zxiaoxue@nudt.edu.cn; Tel.: +86-1387-593-6743

[†] These authors contributed equally to this work.

Abstract: Due to the dynamic complexities of the multi-unmanned vessel target assignment problem at sea, especially when addressing moving targets, traditional optimization algorithms often fail to quickly find an adequate solution. To overcome this, we have developed a multi-agent reinforcement learning algorithm. This approach involves defining a state space, employing preferential experience replay, and integrating self-attention mechanisms, which are applied to a novel offshore unmanned vessel model designed for dynamic target allocation. We have conducted a thorough analysis of strike positions and times, establishing robust mathematical models. Additionally, we designed several experiments to test the effectiveness of the algorithm. The proposed algorithm improves the quality of the solution by at least 30% in larger scale scenarios compared to the genetic algorithm (GA), and the average solution speed is less than 10% of the GA, demonstrating the feasibility of the algorithm in solving the problem.

Keywords: moving targets; weapon-target assignment; unmanned surface vessels; reinforcement learning; multi agent

MSC: 68T42



Citation: Hu, T.; Zhang, X.; Luo, X.; Chen, T. Dynamic Target Assignment by Unmanned Surface Vehicles Based on Reinforcement Learning. *Mathematics* **2024**, *12*, 2557. <https://doi.org/10.3390/math12162557>

Academic Editor: Jonathan Blackledge

Received: 1 July 2024

Revised: 29 July 2024

Accepted: 13 August 2024

Published: 19 August 2024



Copyright: © 2024 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/>).

1. Introduction

In recent years, unmanned platforms have gradually been applied to the military field. It is widely adopted that unmanned platforms will play a vital role in future warfare. In surface or subsurface operations, such as nearshore defense operations, Unmanned Surface Vehicles (USVs) may be used for escort cruising, collaborative strikes, and regional detection [1]. How to foster collaboration among swarm systems becomes an important issue and attracts a lot of research interest. The Defense Advanced Research Projects Agency (DARPA) also sponsored several research programs, such as OFFensive Swarm-Enabled Tactics (OFFSET) [2], Collaborative Operations in Denied Environment (CODE) [3], etc., to advance capabilities for swarms in complex environments.

Modern war is an all-round war, and coastal defense is undoubtedly indispensable. For countries with territorial seas, coastal defense is the first line of defense for their sovereignty and security [4]. From the maritime confrontation during World War II to the current territorial conflicts such as in the Black Sea and the Indian Ocean [5,6], the importance of maritime defense lines has been reflected [7]. Surface operation with USVs is a typical scenario in future maritime conflicts. A swarm of USVs equipped with weapons can be used to strike multiple targets. To achieve aforementioned mission, each USV should be assigned to at least one target, with a planned path to attack position. In other words, target assignment and path finding are key problems.

Many researchers focus on it and propose a problem named weapon-target assignment. Traditional weapon-target assignment models mainly establish models and utility functions. For the model, many scholars focus on the static target assignment problem, that is, they do

not consider the change in the position of the strike target, simplifying it to a target fixed at a specified location within a certain time window [8]; or they fix a weapon firepower unit [9]. However, in actual nearshore combat scenarios, both unmanned surface vessel formations and the strike target are dynamically changing. So that the application of static models will cause deviations in the strike results. Furthermore, traditional models mostly concentrate on the allocation while not considering the path programming, which is not suitable for USV planning. Finally, for utility functions, most only consider combat efficiency, such as the product of target threat and probability of damaging the target, without considering the impact of time costs.

The target assignment problem is NP (non-deterministic polynomial)-hard [10] and can be solved with heuristic algorithms, including construction algorithms and local search algorithms [11]. Construction algorithms mainly sacrifice the quality of the solution under the premise of high efficiency. They are usually used to generate initial solutions [12,13], while local search algorithms mainly search for better solutions through different search operators. Existing algorithms include simulated annealing [14], taboo search [15], GA, and large neighborhood search [16] algorithms. For heuristic algorithms, the final result depends on the initial solution. Therefore, for large-scale problems [17], inappropriate initial solutions will cause the entire algorithm to fall into local optima and a large amount of calculation time. In recent years, reinforcement learning (RL) has made breakthroughs in the field of combinatorial optimization [18–20] due to its ability to estimate useful patterns that are difficult to find manually, especially in large-scale problems [21] and the fast route generation process [22]. It is now widely used to solve NP-hard optimization problems in VRP based on RL models [23]. For example, the literature [24] uses a Deep Deterministic Policy Gradient algorithm for longitudinal speed control to implement a trajectory tracking control method for self-driving vehicles.

Therefore, to solve the large-scale dynamic weapon firepower allocation problem, this paper proposes a dynamic weapon firepower allocation solution model based on deep reinforcement learning (DRL) because the heuristic algorithm cannot accomplish the real-time assignment of dynamic targets. As this moving target problem pays more attention to the cooperation between different agents, we used multi-agent reinforcement learning instead of single-agent. The main contributions are as follows: ① We propose a new mathematical model to describe the surface unmanned craft fire assignment problem in dynamic environments, and mathematically solve the model initially including the strike bearing as well as the strike time. ② We combine task completion time with target threat level to establish a task evaluation model for the current problem. ③ In contrast to the previous use of heuristic algorithms, we propose a MADDPG-based algorithm for solving dynamic problems at different scales and experimentally validate it.

The remainder of this study is organized as follows: Section 1 introduces the background and importance of the moving target assignment. Section 2 provides a brief survey of related studies on model and solving algorithms. Section 3 proposes a mathematical formulation. Section 4 introduces the MTWTA algorithm to solve the formulation. Section 5 test and discuss the algorithm with several experiences. Finally, the conclusions and discussion are presented in Section 6.

2. Related Work

In this section, we mainly focus on the related research in the model of WTA and the solving algorithm.

2.1. Wta

The WTA problem was first proposed by Manne [25] when studying the optimization of ballistic missile defense operations, and was initially called the missile allocation problem (MAP) to allocate one's own interceptor missiles to intercept incoming ballistic missiles, to achieve the purpose of optimal protection of their own facilities. With the increase of researchers, the research point is gradually expanded by air defense fire distribution,

in which the modeling can be divided into static weapon assignment (SWTA) and dynamic weapon assignment (DWTA). For the static assignment, Li [26] studied the weapon-target assignment problem of static multi-target ground air defense, aiming to maximize the protection of own assets while minimizing weapon consumption. Xu [8] studied the WTA problem of ballistic missile defense from a static perspective, considering the multi-target WTA problem under uncertain conditions, to achieve the maximum interception efficiency and minimum interception loss. Choi [27] mainly studied the problem of artillery firing order under uncertain conditions and proposed an optimization method for the order of artillery firing. Li [28] mainly studied the problem of static dual-target fighter jets attacking ground targets.

Bertsekas [29] studied the WTA problem of battlefield missile defense, considering the diversity of incoming targets, the size of own asset value, and the fineness of defense weapons, with the maximization of the preservation value of own assets in the final stage as the optimization goal. Davis [30] studied the problem of multi-round salvo of ballistic missile defense under uncertainty from a dynamic perspective. Shalumov [9] analyzed the cooperative interception scenario between multiple moving targets (aircraft), target defenders, and attacking target missiles. Wang [31] proposed a model to maximize the attack on enemy targets, with the application background of fighter jets carrying air-to-ground missiles attacking ground moving targets.

2.2. Solving Algorithm

In terms of the weapon-target assignment algorithm, the current algorithm is mainly divided into an exact algorithm and an approximate algorithm, and the approximate algorithm includes the rule-based heuristic algorithm, Lagrange relaxation method, meta-heuristic algorithm and machine learning algorithm. Ha [32] designed a branch-and-bound algorithm for field artillery fire scheduling, which could find the optimal solution to the medium-scale problem within a reasonable time. Feghhi [33] designed a branch-and-bound algorithm based on a mixed depth-first selection strategy to solve the small-scale nonlinear integer programming WTA problem. Lu [34] provided an accurate method to quickly solve large-scale WTA problems by modeling WTA problems as a 0-1 integer linear programming models and applying column generation and branch and bound methods to solve them. Xin [35] designed a rule-based heuristic construction algorithm to solve the DWTA problem. In the process of generating feasible solutions, the saturation states of different constraints are dynamically verified to achieve the constraint satisfaction in the process of weapon allocation. Xin [36] proposed a marginal return-based constructive heuristic (MRBCH) algorithm to solve the formulaic sensor-weapon-target assignment problem.

Lee [37] further described the local search method in GA as a eugenic process, and described the greedy recombination scheme search strategy proposed in the literature [38] as greedy eugenics, and proposed an improved greedy eugenics genetic algorithm for solving general WTA problems. Li [39] proposed an improved genetic algorithm to solve the sensor-weapon-target assignment problem. It adopted decimal coding, the coding length was the sum of the number of weapons and the number of sensors, and population initialization was carried out based on rules. Li [40] improved the PSO algorithm for the multi-layer ballistic missile defense WTA problem. Bisht [41] combined the traditional GA and SA algorithms to propose a new hybrid GA to solve the WTA problem. Bogdanowicz [42] designed an innovative algorithm based on enumerating all possible arms-asset combinations to quickly assess the possible collateral damage of a pre-set WTA to neutral/friendly assets. In the field of machine learning, Wang [43] proposed a new adaptive self-organizing mapping algorithm with a recurrent neural network (RNN) controller, which can automatically assign defense missiles to incoming targets and set monitors to reduce the error of matching with the ideal. Gibbons [44] proposed a deep learning approach that automatically learns heuristic rules to solve combinatorial assignment problems, and validated the approach with WTA problems.

Overall, unlike the existing research on weapon-target assignment, this paper will focus on the allocation of movable targets. Meanwhile, compared with the previous heuristic algorithms, a generalized reinforcement learning algorithm will be used to improve the solution efficiency.

3. Moving Targets WTA Model

This study primarily focuses on constructing a target assignment model that targets dynamic objectives using a single firepower node. The model considers a relative positional motion model to determine the real-time positions of the firepower strike node and the dynamic objectives. Additionally, a task evaluation model is established to assess the threat level posed by the dynamic objectives. Finally, mathematical formulas are proposed to minimize the enemy threat level within the entire target assignment model.

In summary, this research aims to develop an effective firepower allocation strategy by considering the real-time positions of the targets and evaluating their threat levels. By minimizing the enemy threat level, the proposed model seeks to optimize the allocation of firepower resources.

3.1. Problem Description

The primary objective of this research is to establish a model for firepower allocation targeting dynamic objectives. In this scenario, the enemy firepower units are dispersed around bases and engage in reconnaissance and destruction missions. The strategy involves deploying a swarm of unmanned surface vessels (USVs) from the base to strike all enemy firepower units. This can be represented by an array $\langle A, N, E, C \rangle$, where A represents the USV swarm, N represents the enemy firepower units with their survival and threat assessment information, including the number of units, E represents the real-time position and situational information of both firepower units and enemy targets, and C represents the constraints for the model during the mission.

In the context of dynamic target firepower allocation, determining the optimal path for firepower nodes faces two main challenges. Firstly, enemy targets typically move in different directions within a specific range during their reconnaissance and destruction operations. This necessitates continuous observation of their positions by these firepower nodes, emphasizing the need for minimal travel time. Secondly, each target possesses different characteristics in terms of size, velocity, direction, and distance. For example, priority may be given to swiftly advancing enemy units or larger targets. Therefore, it is crucial to prioritize targets with higher threat levels. Considering these factors, the model considers the time required for firepower nodes to reach each strike position and the threat levels of targets when determining the optimal firepower allocation path. The model is based on the following assumptions:

- Real-time position, direction, and velocity information of targets can be obtained through intelligence reconnaissance systems. The model in this paper only considers the enemy speed and direction at the calculation time.
- The time required for firepower strikes is not explicitly considered. It is assumed that once firepower nodes reach the designated strike positions, the attacks on enemy targets are successfully executed.

3.2. Time

In order to determine the transfer route of the fire nodes, we analyze the relative movement of fire nodes and targets. We establish a Cartesian coordinate system. Due to the dynamic nature of the enemy targets, it is necessary to determine the position points and time at which the enemy targets are within the striking range of the firepower strike equipment. The meaning of parameters is shown in Table 1.

Table 1. The parameters.

Parameters	Meaning
N	targets
A	unmanned surface vessel
V_A	the speed of USV
V_n	the speed of target n
β_n	the angle between direction and the vertical
L_n	Strike position for the target n
(X_i, Y_i)	the real-time location of USV
$(X_{n,0}, Y_{n,0})$	the initial location of the target n
(X_{n,t_i}, Y_{n,t_i})	the real-time location of target in t
L_n	strike position for target n
l	the distance

3.2.1. The Position Points

If the distance between the firepower attacking nodes and the enemy target is greater than the attacking range, we can determine the position point based on the following two hypotheses.

Hypothesis 1. *When the target and the initial position of these firepower attacking nodes are on the same straight line as the attacking point, it is the optimal attacking point.*

As shown in Figure 1, we construct a circle with the enemy target B as the center, L as the optimal attacking point, and any point L' on the attacking circle as the attacking point. According to the perpendicular theorem and Pythagorean theorem, we have:

$$(l_2 + |OL|)^2 + |OL'|^2 = l_1^2 \geq l_2^2 \tag{1}$$

Therefore, it can be concluded that the L point is the optimal striking point as it results in the shortest possible path.

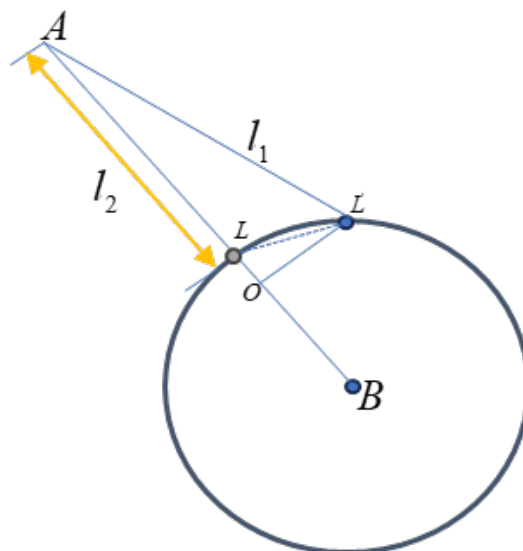


Figure 1. Striking circle.

Hypothesis 2. *When the time for the strike equipment to reach the striking position is equal to the time for the enemy to reach the target position, it is considered the shortest time.*

As shown in Figure 2, L_2 the optimal strike point, L_1 means that the time it takes for the firepower strike node to reach the impact point is less than the time it takes for the enemy to arrive at the target location. Analysis indicates that when the fire strike node A arrives at L_2 ahead of time, it needs to wait for a duration equal to the time it takes for the enemy target to reach B' , which is longer than the duration to reach strike point L_1 ; if the target arrives B_{best} early, the enemy target has not yet entered firing range. Therefore, it is proved that the arrival time of the enemy target and firepower node should be consistent.

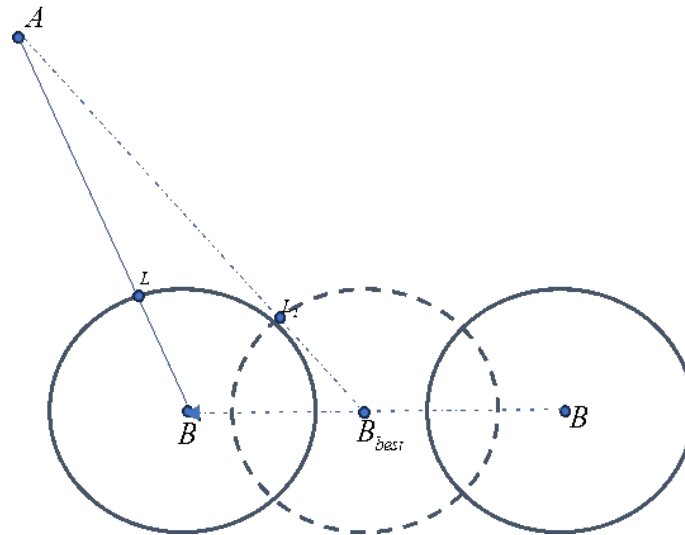


Figure 2. Enemy target movement chart.

3.2.2. The Time of Fight

In order to determine the transfer route for the fire strike node, we have established a Cartesian coordinate system based on the relative motion of fire strike node and the enemy target. Due to the dynamic nature of the enemy target, we need to determine the position and time at which the enemy target falls within the striking range of fire strike equipment.

Scenario 1: Initial Position Transfer.

As shown in Figure 3, at the initial moment, we depart from the base to engage the first designated target. Let us assume that we arrive at the firing position after a certain time, denoted as $t_{0,i}$. Point A represents the location of the base, point B represents the initial location of the enemy's first target, L represents the designated strike position, and B' represents the mapped point of the enemy target. In $\Delta ABB'$, we can apply the cosine rule to calculate:

$$(l_{AL} + r)^2 = l_{AB}^2 + |BB'|^2 - 2|l_{AL} + r| \cdot |l_{AB}| \cdot \cos \angle ABB' \tag{2}$$

the length is:

$$l_{AL} = v_A \cdot t_{0,i} \tag{3}$$

$$|AB| = \sqrt{(X_0 - X_{i,0})^2 + (Y_0 - Y_{i,0})^2} \tag{4}$$

$$l_{B'B} = v_i \cdot t_{0,i} \tag{5}$$

where β_i represents the direction of the enemy target's movement, which can be obtained from Equation (6):

$$\alpha_i = \arctan \frac{Y_{i,0} - Y_0}{X_{i,0} - X_0} \tag{6}$$

$$\cos \angle ABB' = -\sin(\alpha_i + \beta_i) \tag{7}$$

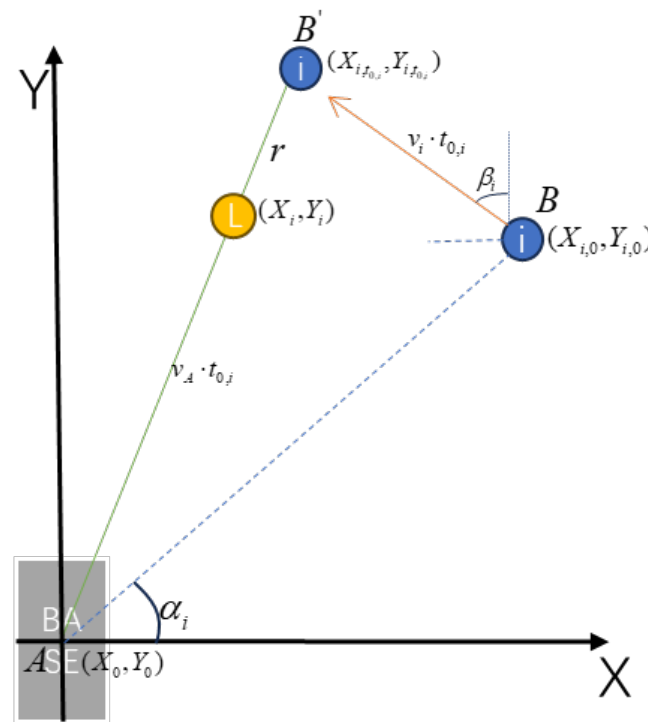


Figure 3. Path diagram from the base to the first strike point.

Scenario 2: intermediate node transfer.

After determining the time when the fire strike node completes the strike on target i , we can calculate the position changes of each target. As shown in Figure 4, when the fire strike node completes the strike on target i at time t_i , the positions of the fire strike node and target j are at points $L_i(X_i, Y_i)$ and $J(X_{j,t_i}, Y_{j,t_i})$. Now, we consider that the fire strike node departs from L_i to complete the strike on target J . Assuming that after the interval $t_{i,j}$, the fire strike node moves from point L_i to point L_j , and the enemy target moves from point J to point J' . At this moment, the enemy target enters the firing range. In $\Delta L_i J J'$, according to the cosine rule, we have:

$$(v_A \cdot t_{i,j} + r)^2 = |L_i J|^2 + |v_j \cdot t_{i,j}|^2 - 2|v_j \cdot t_{i,j}| \cdot |L_i J| \cdot \cos \angle L_i J J' \tag{8}$$

$$|L_i J| = \sqrt{(X_{i,t_i} - X_i)^2 + (Y_{i,t_i} - Y_i)^2} \tag{9}$$

The location of enemy target at t_i :

$$\begin{aligned} X_{i,t_i} &= X_{i,0} + v_i t_i \sin \beta_i \\ Y_{i,t_i} &= Y_{i,0} + v_i t_i \cos \beta_i \end{aligned} \tag{10}$$

the $\angle L_i J J'$ can be calculated by equal (11):

$$\cos \angle L_i J J' = \frac{\vec{L_i J} \cdot \vec{J J'}}{|\vec{J J'}| \cdot |L_i J|} \tag{11}$$

$$\frac{\vec{J J'}}{|\vec{J J'}|} = (\sin \beta_j, \cos \beta_j) \tag{12}$$

According to Equations (8)–(12), we can calculate the time:

$$M = \sqrt{v_A^2 + v_J^2 \cdot \frac{r^2}{l^2} + 2v_A \cdot v_J \cdot \frac{r}{l} \cos \angle L_i J J' - v_J^2 \sin^2 \angle L_i J J'} \tag{13}$$

$$t = \frac{M - (v_A \cdot \frac{r}{l} + v_J \cos \angle L_i J J')}{(v_A^2 - v_J^2)} \tag{14}$$

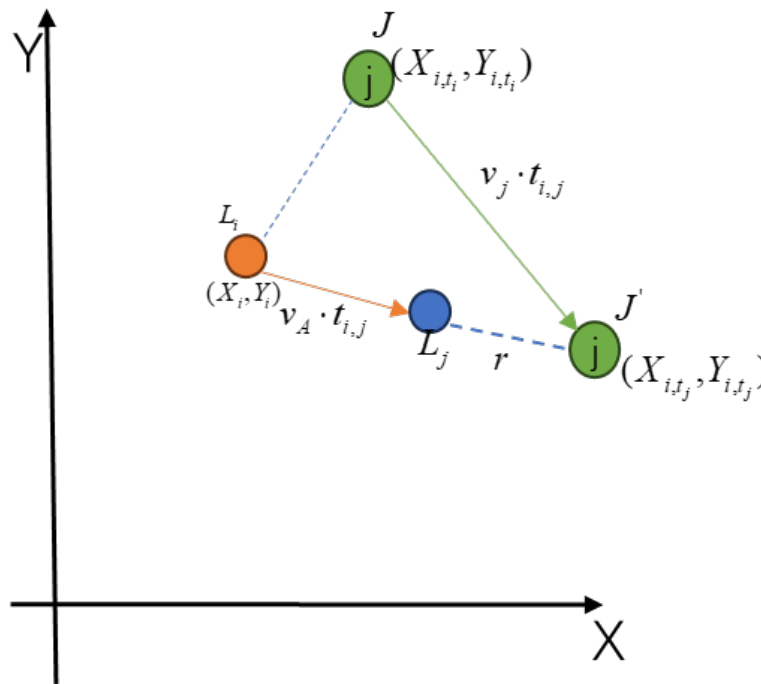


Figure 4. The route of intermediate node transfer.

3.2.3. Model for Threat Assessment

The traditional VRP problem typically measures the quality of a selected solution by the total length of the path. However, in the context of military operations, the main objective is to maximize the overall combat effectiveness by minimizing the threat level posed by the enemy. Therefore, this article primarily focuses on evaluating this criterion from two perspectives:

(a) **Enemy Threat Level:** Enemy threat: The primary goal of VRP in combat is to minimize the threat of the enemy. In offshore defense, the threat degree of the target is initialized according to the speed and type of the enemy attacking ships. The calculation of the threat degree of the enemy can be obtained from the literature [45].

(b) **Strike Time (time from target detection to engagement):** The enemy targets may gather information about resources and infrastructure through reconnaissance and sabotage missions. As time goes on, the enemy can collect more intelligence, posing a greater threat to us. Modern warfare emphasizes quick response and mobility. Conducting operations within a short time frame can shorten the enemy’s reaction time, reducing their interference and obstruction of actions. Swiftly striking targets can increase mobility and flexibility.

$$c = \sum_{i=1}^n vt_i \tag{15}$$

Since the target strike order is different and the rewards obtained are different, we define the threat reduction of the enemy target as follows:

$$\omega'_i = \omega_i * \varphi \tag{16}$$

$$\varphi = 0.9^{\lambda-1} \quad (17)$$

λ is the strike sequence of the target in the unmanned surface vessels; if it is the second strike, then $\lambda = 2$.

Therefore, the reward for one of the unmanned surface vessels is defined as:

$$\omega = \sum_{i=1}^n \omega_i - c \quad (18)$$

3.2.4. Mathematical Formulation

$$\min \omega \quad (19)$$

s.t.

$$u_{i,j} = \{0, 1\}, i, j \in N, i \neq j \quad (20)$$

$$\sum_{i \in N} u_{0,i} = 1 \quad (21)$$

$$\sum_{i \in N} u_{i,0} = 1 \quad (22)$$

$$\sum_{i \in N} \sum_{j \in N} u_{i,j} = n \quad (23)$$

$$|L_i J_i| \leq r \quad (24)$$

$$\sum_{i,j \in N} t_{i,j} \leq T_{\max} \quad (25)$$

Constraint (20): At most, one visit from i to j . This constraint states that in path planning, each target point can only be visited once in the path from the starting point i to the destination point j .

Constraints (21) and (22): Must depart from and return to the base.

These constraints ensure that the path planning must start from the base, pass through a series of target points, and finally return to the base.

Constraint (23): Must visit all target points. This constraint requires that the path planning must pass through all target points, without skipping or ignoring any of them.

Constraint (24): Distance between target and strike point must be within strike radius. This constraint ensures that the selected target points in the path planning are within the strike radius of force, enabling effective strikes.

Constraint (25): Maximum endurance time. This constraint limits the maximum endurance time of the aircraft in the path planning, ensuring that the aircraft can complete the mission within the specified time frame.

4. Reinforcement Learning Model

In this section, we will introduce the algorithm. As shown in Figure 5, it is the framework overview of MTWTA, which is based on the MADDPG. And this article has combined it with the RNN and attention mechanism. The whole framework is made up of three parts: the encoder, the network of actors and the network of critics. The encoder draws on RNN and self-attention mechanisms for state representation, the actor network for action selection, and the critic network for training. The following will describe each component in detail.

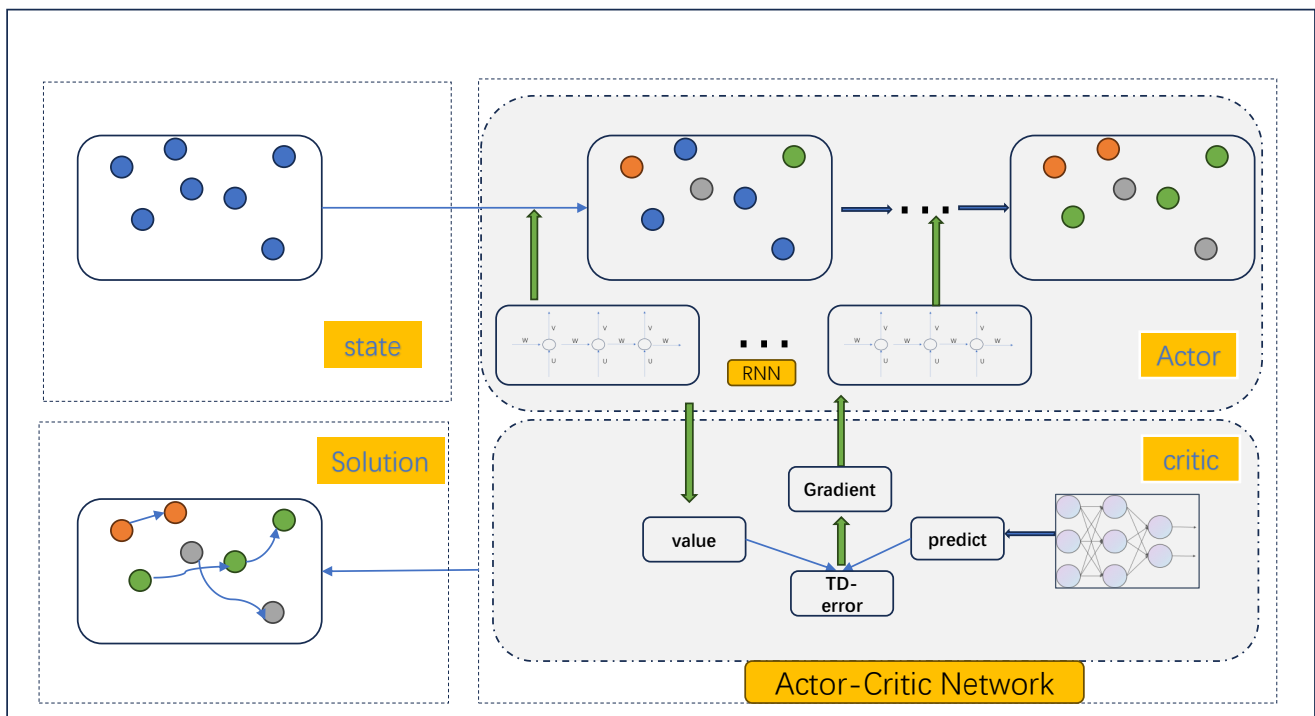


Figure 5. The framework of MTWTA.

4.1. Multi-Agent Reinforcement Learning Setting

State: Since the entire environment for the side is open information, that is, all unmanned surface vessels share target location information and attack information, we define the state of a single agent to include its own information and environmental information. The state of each unmanned surface vessel includes its location information, the sequence of targets it has already visited, and the remaining energy of the unmanned surface vessels; the public information in the environment includes the location, speed, threat level, attack status of the target, and the locations of other unmanned surface vessels.

$$s_i = ((x_i, y_i), (tar_1, \dots, tar_m), (agent_1, \dots, agent_n)) \tag{26}$$

Action: The unmanned surface vessels can choose an un-attacked target or stop moving. This can be encoded as a discrete action space. We use one-hot encoding to represent each action value. For example, if the third target is selected, it is defined as $(0, 0, 1, \dots, 0)$ and the stop action is represented as $(0, 0, 0, \dots, 1)$. When the unmanned surface vessels reach their maximum capacity or there are no targets to attack in the entire environment, the returned action is ‘-1’, that is, return to the origin. The step indicates that one target is assigned, and the other unassigned targets will be reassigned to achieve online dynamic allocation.

Reward: When the unmanned surface vessels completes the target attack, a positive reward is given. The size of the reward is related to the threat level of the target. At the same time, each time an action is executed, the unmanned surface vessels will consume a certain amount of energy, so each action will receive a negative reward proportional to the energy consumption.

Mask: Due to the limitations of the target visit times and the maximum capability of the unmanned surface vessels, there may be moments when the target can no longer be selected or the unmanned surface vessels cannot carry out the strike mission. Therefore,

we use a mask to indicate whether the unmanned surface vessels can perform the action at a certain moment. The mask rule is defined as follows:

$$m_i = \begin{cases} 1, & \text{un - assigned} \\ 0, & \text{otherwise} \end{cases} \quad (27)$$

When the unmanned surface vessels reach their maximum capability, we define the mask as $(0, 0, 0, \dots, 1)$, indicating that the unmanned surface vessels cannot perform more actions.

4.2. Multi-Agent Deep Deterministic Policy Gradient

In an environment with multiple unmanned surface vessels, traditional heuristic algorithms face significant challenges. In this environment, each unmanned surface vessel is an individual agent that needs to continuously learn to obtain the optimal strategy. From the perspective of each agent, the environment is no longer static but dynamic. The appearance of the MADDPG algorithm [46] can effectively solve such problems.

The Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm is an extension of the Deep Deterministic Policy Gradient (DDPG) algorithm [47]. It is an intelligent algorithm that can handle multi-agent cooperation problems that traditional reinforcement methods cannot address. The MADDPG algorithm adopts a “centralized training, decentralized execution” framework for learning, as shown in Figure 6. In a multi-agent environment, the behavior of each agent affects the observation results and rewards of other agents, making the dynamics of the environment non-stationary and posing challenges to learning. MADDPG addresses this problem by allowing each agent to have its own actor–critic network, where the actor is the action network and the critic is the evaluation network. During the training process, each agent’s policy network only uses the agent’s own observations and actions, while the value function network uses the observations and actions of all agents.

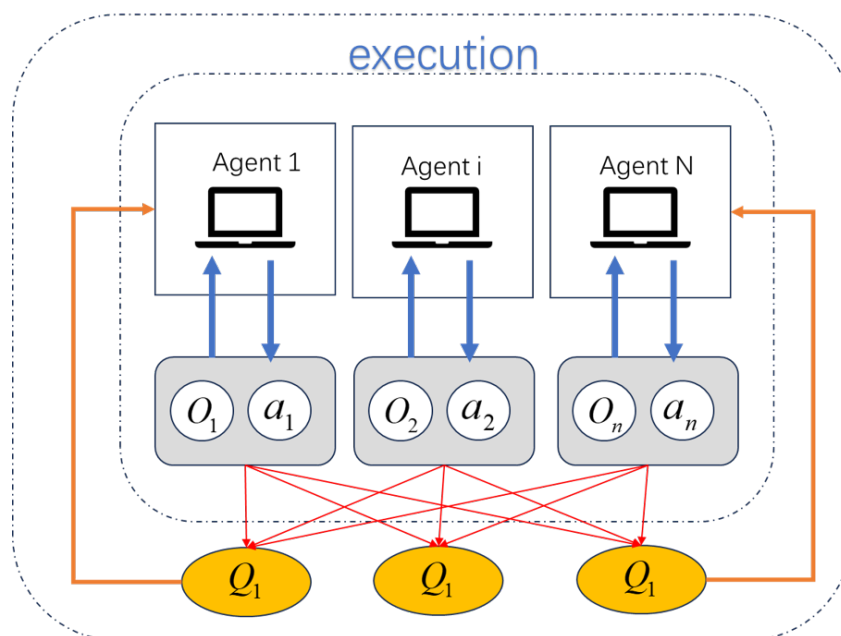


Figure 6. MADDPG.

The training process of MADDPG includes two main steps: policy update and value function update.

In the policy update step, each agent i generates actions $a_i = \mu_{\theta_i}(o_i)$ according to its policy function, and then uses the value function to calculate the expected rewards of these

actions $Q_{\phi_i}(o_1, \dots, o_N, a_1, \dots, a_N)$. Then, the parameters of the policy function are updated through gradient ascent to maximize the expected reward:

$$\nabla_{\theta_i} J(\mu_{\theta_i}) = \mathbb{E}_{o_1, \dots, o_N \sim D} \nabla_{\theta_i} \mu_{\theta_i}(o_i) \nabla_{a_i} Q_{\phi_i} |_{a_i = \mu_{\theta_i}(o_i)} \tag{28}$$

In the value function update step, each agent i uses its policy function and the policy functions of other agents to generate actions $a_i = \mu_{\theta_i}(o_i)$, and then uses these actions and observations to calculate the target value $y = r_i + \gamma Q_{\phi'_i}(o'_1, \dots, o'_N, a'_1, \dots, a'_N)$. Then, the parameters of the value function are updated through gradient descent to minimize the difference between the target value and the actual value:

$$\nabla_{\phi_i} L(\phi_i) = \mathbb{E}_{o_1, \dots, o_N, a_1, \dots, a_N, r_i, o'_1, \dots, o'_N \sim D} [(Q_{\phi_i} - y)^2] \tag{29}$$

where D is the experience replay buffer, γ is the discount factor, and $'$ indicates the next time step.

Furthermore, gradient vanishing may occur during training. Gradient vanishing is a common problem during the training of neural network models, especially when using back-propagation algorithms. As the number of network layers increases, the gradient may gradually decrease to close to zero during back-propagation, resulting in slow or stagnant network weight update, which affects the training effect. Gradient vanishing mainly focuses on unreasonable limitations of the number of network layers, inappropriate activation functions or poor weight initialization. It is suggested in the literature [48] that the effect of gradient vanishing can be reduced by dynamically adjusting the learning rate and optimizing the loss function. On this basis, we reduce the possibility of gradient vanishing by constantly adjusting the number of network layers and using unsaturated activation functions such as ReLU.

4.3. Encoding

In the research, we propose a novel approach to handle variable-length navigation trajectory sequences. This method combines recurrent neural networks (RNN) and self-attention mechanisms to capture long-distance dependencies in the sequence and handle inputs of different lengths.

Firstly, we employ an RNN to process the input navigation trajectory sequence. RNNs are a type of neural network capable of handling sequence data, capturing temporal dependencies in the sequence. In the model, we use gated recurrent units (GRU) as the basic unit of the RNN, as GRUs can effectively handle long sequences and have relatively low computational complexity. The update equations for GRU are as follows:

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \tag{30}$$

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \tag{31}$$

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t]) \tag{32}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{33}$$

where r_t and z_t are the reset and update gates, h_t is the hidden state, x_t is the input, $*$ denotes element-wise multiplication, σ is the sigmoid function, and W represents the weight matrices.

Next, we employ a self-attention mechanism to further process the output of the RNN. The self-attention mechanism is a method capable of capturing long-distance dependencies in the sequence. It computes an interaction between each element and all other elements in the sequence to generate a context vector as a weighted average. The computation for self-attention is as follows:

$$Q = W_q \cdot h \tag{34}$$

$$K = W_k \cdot h \tag{35}$$

$$V = W_v \cdot h \quad (36)$$

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (37)$$

where Q, K, V are the query, key, and value, h is the output of the RNN, W_q, W_k, W_v are the weight matrices, d_k is the dimension of the key, and softmax is the softmax function.

By combining RNN and self-attention mechanisms, the model can effectively handle variable-length navigation trajectory sequences and capture long-distance dependencies in the sequence.

4.4. Algorithm

In this section, we will show the algorithm of action selection and the MTWTA. As shown in Algorithm 1, is the action selection for each agent used the self-attention with RNN, and Algorithm 2 shows the whole progress of the algorithm we have proposed.

Algorithm 1 Self-Attention with RNN and Mask for Action Selection

Initialize random weights of RNN
 Initialize self-attention mechanism
 Initialize action space A
 For each episode:
 4.1 Get initial state s_0
 4.2 Pass s_0 through RNN to get hidden state h_0
 4.3 For each time step t :
 4.3.1 Pass h_{t-1} through self-attention to get attention output o_t
 4.3.2 Concatenate s_t and o_t to form the input x_t for the RNN
 4.3.3 Pass x_t through RNN to get h_t
 4.3.4 Compute action probabilities p_t using gumbel-softmax function on h_t
 4.3.5 Apply mask to p_t to get masked probabilities p'_t
 4.3.6 Select action a_t from A according to p'_t
 4.3.7 Execute action a_t and observe reward r_t and new state s_{t+1}

Algorithm 2 Processing procedure of WTA

Initialize Q, μ, Q', μ', R
 For each episode, initialize N , receive s_1
 For each time step t :
 3.1 For each agent i , select $a_i = \mu(s_i) + N_t$, apply mask
 3.2 Execute $a = (a_1, \dots, a_N)$, observe r, s' , store (s, a, r, s') in R , set $s = s'$
 3.3 Sample minibatch of N transitions from R , set $y_i = r_i + \gamma Q'(s'_i, \mu'(s'_i))$
 3.4 Update critic by minimizing $L = (y_i - Q(s_i, a_i))^2$
 3.5 Update actor policy using sampled policy gradient
 3.6 Update target networks:
 $\theta_{Q'} = \tau\theta_Q + (1 - \tau)\theta_{Q'}$, $\theta_{\mu'} = \tau\theta_\mu + (1 - \tau)\theta_{\mu'}$

5. Experiment

Numerate experiments are divided into two parts. The first one is a case study; we set an instance of (5, 25) to calculate WTA. The second part is a comparative performance evaluation. The experiment is carried out on AMD Ryzen7 5800H@3.2 Hz, RTX3060 with 16 GB RAM, code in Python 3.10.

Dataset: Due to the restrictions of military datasets, all data in this experiment are randomly generated. A specified number of targets are randomly generated according to the problem scale, and their coordinates are quantified and limited within $(-10, 10)$, with their speed within $(1, 2)$. Their initial threat levels are calculated through a table and all threat levels are normalized. The initial coordinates of agents are all $(0, 0)$, with a set speed of 10, and the maximum attack capability of unmanned surface vessels is 5.

Baseline: The conventional solution to planning problems in the military field mainly uses the GA algorithm. Therefore, this experiment mainly compares the genetic algorithm under different scales. When the target is greater than or equal to the maximum allocation ability, we allocate by bit [49], that is, we group the chromosome encoding according to the maximum ability of the unmanned surface vessels and allocate it to unmanned surface vessels in turn. When the enemy target has less than the maximum allocation ability, we refer to the literature, insert the virtual encoding of the unmanned surface vessels into the target chromosome encoding, and define the chromosome segment between two unmanned surface vessels as the attack segment of the previous unmanned surface vessels. The chromosome sequence is the attack sequence of the unmanned surface vessels. If the number between the unmanned surface vessels is greater than the maximum ability of a single boat, the first max ones are taken as the attack sequence of this unmanned surface vessels.

Gray Wolf Algorithm: The optimal solution to the problem is achieved by simulating the social and predatory behavior of grey wolves. A hierarchical model of social hierarchy is formed by ranking each grey wolf according to its fitness value. The grey wolf algorithm adjusts its movement strategy according to the current position and the target position to better approximate the optimal solution and finds the optimal solution in the search space and approaches the target gradually by adjusting their position. We use the basic grey wolf algorithm with 300 iterations as a comparison algorithm

Parameter setting: the parameter of MTWTA is shown in Table 2.

Table 2. The training parameter.

Actor learning rate	1×10^{-4}
Critic learning rate	1×10^{-4}
Batch size	32
max iteration	1×10^6
hidden dimension	128
elite rating	0.3
mutation probability	0.4

We set the reward as $\omega = \left(\sum_{i=1}^n \omega_i - c \right) * 100$, and the fitness function of GA is set as same as the reward for that the result is my optimization goal. But different with the RL, we only calculate after the entire pre-assignment is completed, without considering dynamic reward.

5.1. Case Study

We use the MTWTA method to solve the fire distribution problem, where the number of unmanned surface vessels is 5 and the number of enemy unmanned surface vessels is 25. We set the number of training times to 10,000, the dimension of the hidden layer to 128, and the model updates every 50 times. We apply the trained model to solve the problem, and the results obtained are as shown in Figure 7; the assignment is that

s1: 2 → 20 → 19 → 24 → 10, s2: 3 → 9 → 21 → 16 → 15, s3: 4 → 17 → 22 → 13 → 6, s4: 7 → 0 → 23 → 11 → 1, s5: 12 → 5 → 8 → 14 → 18. The coordinates in Table 3 represent the positions of each unmanned surface vessel when attacking the final target. It shows the final position of each unmanned surface vessel before it comes back to the base.

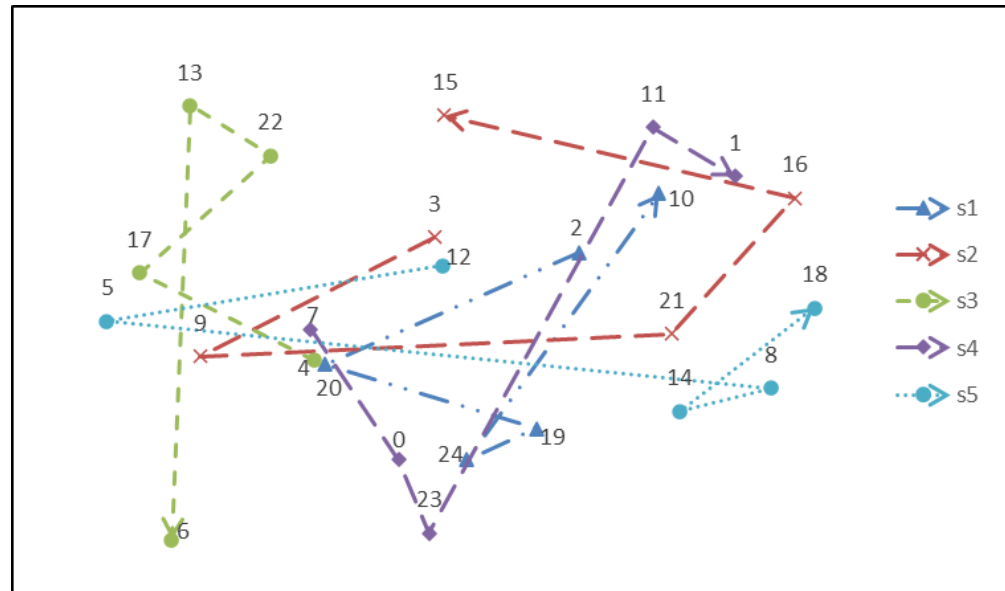


Figure 7. Solution.

Table 3. Final assignment.

	Fight Location	Target ID	Target Initial Location	Target Final Location
s1	(1.886, 1.686)	10	(4.146, 5.487)	(3.726, 5.258)
s2	(2.1415, 5.618)	15	(−0.078, 8.737)	(−0.414, 8.710)
s3	(−3.591, −5.604)	6	(−5.453, −8.908)	(−3.625, −9.619)
s4	(2.448, 6.434)	1	(5.132, 6.023)	(5.640, 6.221)
s5	(3.406, −0.79)	18	(7.221, 0.729)	(7.234, 0.483)

5.2. Comparative Performance Analysis

5.2.1. The Quality of Solution

To verify the scalability of the algorithm, this experiment is designed with three comparative tests, which are conducted, respectively, when the enemy targets exceed unmanned surface vessels’ total allocation ability, when the enemy targets equal with maximum allocation ability, and when the enemy targets are less than the maximum allocation ability.

Experiment 1: When the target is less than maximum capability.

We set the number of unmanned surface vessels and the enemy, respectively, as (30, 100), (20, 60), and (5, 16) for three sets of experiments. Here, GA100, GA200, and GA300 represent experiments with iteration times of 100, 200, and 300, respectively. For reinforcement learning, we calculate the average of 100 times, and the solution of the GA algorithm is the average of 10 solutions. The quality of the solution is shown in Table 4. The results represent the ratio of the solution rewards compared to the proposed algorithm. The larger the result, the worse the quality of the solution.

Table 4. The rewards when the target is more than the max of the ability of USVs.

	N = 5, M = 16	N = 30, M = 100	N = 60, M = 200
MTWTA	1	1	1
GA100	1.29	1.53	1.82
GA200	1.17	1.50	1.75
GA300	0.98	1.44	1.71
GWO	0.85	1.32	1.59

Experiment 2: When the target is equal to the maximum capability.

We set the number of unmanned surface vessels and the enemy, respectively, as (30, 100), (20, 60), and (5, 16) for three sets of experiments. The quality of the solution is shown in Table 5.

Table 5. The rewards when the target is equal to the max of the ability of USVs.

	N = 5, M = 25	N = 20, M = 120	N = 30, M = 150
MTWTA	1	1	1
GA100	0.86	1.36	1.97
GA200	0.75	1.28	1.81
GA300	0.80	1.26	1.69
GWO	0.82	1.10	1.58

Experiment 3: When the target is more than maximum capability.

We set the number of unmanned surface vessels and the enemy, respectively, as (3, 17), (20, 120), and (30, 160) for three sets of experiments. The quality of the solution is shown in Table 6.

Table 6. The rewards when the target is more than the max of the ability of USVs.

	N = 3, M = 17	N = 20, M = 120	N = 30, M = 160
MTWTA	1	1	1
GA100	0.69	2.06	1.77
GA200	0.58	1.95	1.73
GA300	0.61	1.93	1.71
GWO	0.59	1.78	1.52

Through experiments under three different scenarios, we found that the algorithm proposed in this paper is superior to the baseline algorithm in terms of solution quality when the scale is large. However, when the scale is small, the solution quality obtained by the MTWTA algorithm is lower than that of the GA, especially when the number of enemy targets exceeds the maximum capacity, the rewards obtained by MTWTA are much lower than GA.

The reason for this phenomenon is that when the problem scale is small, the solution space of the entire experiment is also small. As a search algorithm, the GA is more likely to find the optimal solution in a smaller solution space. However, when the problem scale expands, the solution space also increases, and the quality of the solution cannot be guaranteed in a short number of iterations. From the algorithmic design point of view, reinforcement learning interacts with the environment in real time and uses a trial-and-error learning mechanism. The intelligent body learns the optimal strategy by continuously trying different actions and observing the results, and with the balance of exploration and utilization, can lead the intelligent body to learn a better strategy, while GA focuses more on searching for optimal solutions through the evolution of individuals in a population. Although it has a certain exploration ability, its learning mechanism is relatively

passive, mainly relying on the evaluation of the fitness function and genetic manipulation, and therefore has relatively little feedback and is slow to guide changes in the quality of the solution.

5.2.2. The Running Time

In practical applications, the solution time is a key indicator to measure the performance of an algorithm. Therefore, we have calculated and compared the solution times of various algorithms at different scales, as shown in Figure 8. We have calculated the average time of running the algorithm ten times. This method can help us to evaluate the performance of the algorithm more accurately.

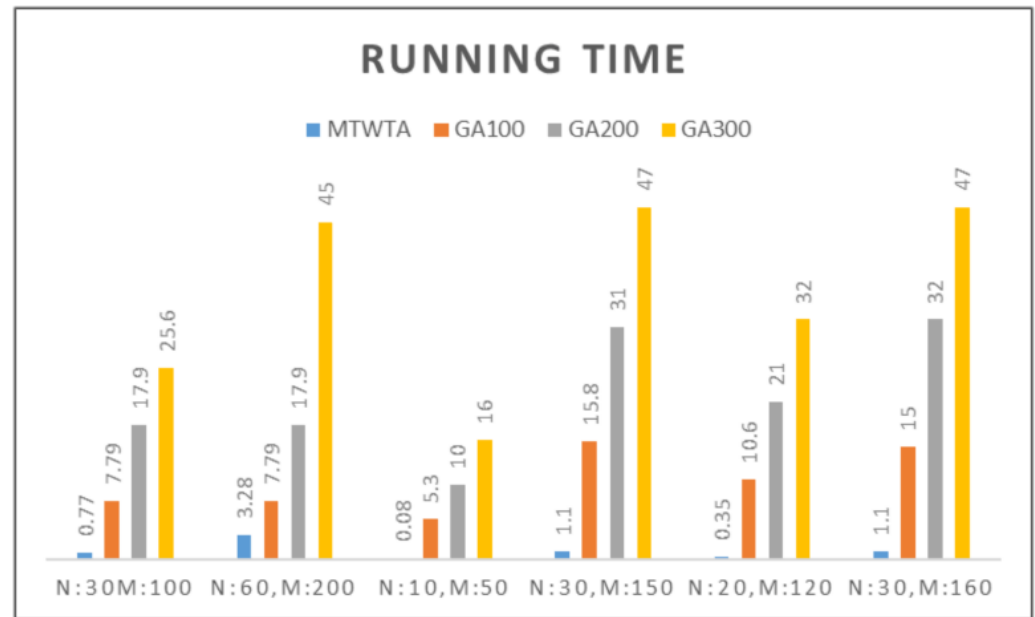


Figure 8. The time to solve the problem.

In terms of comparing the solution time of algorithms, the algorithm proposed in this paper shows significantly better performance than GA. Although the genetic algorithm may find a better solution for small-scale problems, as the scale of the problem increases, its solution time also significantly increases. Experimental results show that the solution time of the proposed algorithm is much less than that of GA, and as the scale of the problem increases, this time advantage becomes more prominent. For the GA, with the increase of the number of genetic generations, it will continuously expand its search space due to the influence of various parameters, such as crossover mutation operation, and at the same time, it is necessary to calculate the fitness function value of each individual in each iteration, which increases the solution time of the whole problem.

5.2.3. The Stability

In this paper, we mainly focus on the solution stability. The quality of solutions provided by search algorithms largely depends on the quality of the initial solution space. If the initial solution space is of poor quality, even the best search algorithms may not be able to find satisfactory solutions. Therefore, the stability of solutions becomes an important criterion for measuring algorithm performance.

The stability of solutions reflects the consistency of the algorithm's performance under different initial solution spaces. A highly stable algorithm can find satisfactory solutions consistently, regardless of the quality of the initial solution space. This is particularly important for real-world problems, as we often cannot guarantee the quality of the initial solution space.

Therefore, we use the stability of solutions as an important comparison criterion to evaluate and compare the performance of different algorithms. In this way, we can understand and evaluate the performance of algorithms more comprehensively and accurately, thereby selecting the algorithm that is most suitable for the actual problem. The result is shown in Figure 9.

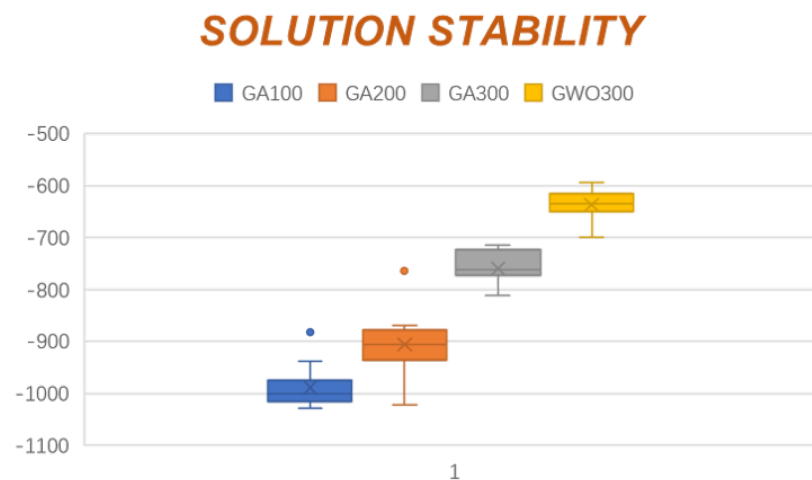


Figure 9. Solution stability.

Experimental results show that the solutions of GA are volatile, while the model trained by MTWTA shows stability in solving problems. This is because the MTWTA algorithm continuously optimizes strategies by learning from environmental feedback. Even in cases where the initial solution space is of poor quality, it can find satisfactory solutions through continuous learning and iteration. On the contrary, if the initial solution space of the genetic algorithm is of poor quality, it may be difficult to get rid of the local solution space by operations such as crossover and mutation, while the randomness of the crossover and mutation operations can lead to fluctuations in the solution of the problem.

6. Summary and Discussion

This study proposes a new model to describe and solve the weapon-target assignment problem of unmanned surface vessels in the near sea. Unlike the existing problems, the target of the surface unmanned craft is in a moving state, so the existing modeling methods have certain constraints. This means that it needs to consider the strike position of each target and the movement direction of the unmanned surface vessels at different times. To better describe this problem, this research has established a new weapon-target assignment model and a reasonable performance evaluation system. A constrained mathematical model with the goal of maximizing the target was set up. In order to solve this problem, we modified the embedding of MADDPG with the help of the principle of Seq2seq, and proved the timeliness, stability and quality of the proposed framework through the comparison experiment with the genetic algorithm. Through the experimental results, we found that our proposed framework is able to obtain a stable solution that is 30% better in less than 10% of the time in situations of different sizes.

This study can provide some help in algorithm and modeling for subsequent dynamic fire distribution. In the current intelligent battlefield, the timeliness and stability of the algorithm can better assist the commander in making decisions. At the same time, this algorithm is not the only solution to this problem. In the subsequent research process, we will focus on the fire distribution of continuous actions, and consider the the confrontation.

Author Contributions: Methodology, X.Z.; Writing—original draft, T.H.; Funding acquisition, X.L. and T.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by the National Natural Science Foundation of China (No. 62302510).

Data Availability Statement: Data will be made available on request.

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

References

1. Zheng, Y.; Tao, J.; Sun, Q.; Sun, H.; Chen, Z.; Sun, M.; Xie, G. Soft Actor–Critic based active disturbance rejection path following control for unmanned surface vessel under wind and wave disturbances. *Ocean. Eng.* **2022**, *247*, 110631. [[CrossRef](#)]
2. Chung, T.H.; Daniel, R. DARPA OFFSET: A Vision for Advanced Swarm Systems through Agile Technology Development and Experimentation. *Field Robot.* **2023**, *3*, 97–124. [[CrossRef](#)]
3. Drubin, C. Collaboration Vital to DARPA’s CODE for Success. *Microw. J.* **2019**, *62*, 41.
4. Gray, C.S. *Modern Strategy*; Oxford University Press: Oxford, UK, 1999.
5. Haward, M. *Maritime Power in the Black Sea*; Routledge: London, UK, 2015.
6. Mahan, A.T. Maritime Security Challenges in South Asia and the Indian Ocean: Response Strategies. In Proceedings of the Center for Strategic and International Studies—American-Pacific Sealanes Security Institute Conference on Maritime Security in Asia, Honolulu, HI, USA, 18–20 January 2004.
7. Till, G. *Seapower: A Guide for the Twenty-First Century*; Routledge: London, UK, 2004.
8. Xu, H.; Xing, Q.; Tian, Z. MOQPSO-D/S for Air and Missile Defense WTA Problem under Uncertainty. *Math. Probl. Eng.* **2017**, *2017*, 9897153. [[CrossRef](#)]
9. Shalumov, V.; Shima, T. Weapon-target-allocation strategies in multiagent target-missile-defender engagement (Article). *J. Guid. Control. Dyn.* **2017**, *40*, 2452–2464. [[CrossRef](#)]
10. Lee, Z.J.; Lee, C.Y.; Su, S.F. An immunity-based ant colony optimization algorithm for solving weapon–target assignment problem. *Appl. Soft Comput.* **2003**, *2*, 39–47. [[CrossRef](#)]
11. Paraskevopoulos, D.C.; Laporte, G.; Repoussis, P.P.; Tarantilis, C.D. Resource constrained routing and scheduling: Review and research prospects. *Eur. J. Oper. Res.* **2017**, *263*, 737–754. [[CrossRef](#)]
12. Grangier, P.; Gendreau, M.; Lehuédé, F.; Rousseau, L.M. An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization. *Eur. J. Oper. Res.* **2016**, *254*, 80–91. [[CrossRef](#)]
13. Lulij, I.; Kramer, S.; Schneider, M. A hybrid of adaptive large neighborhood search and tabu search for the order-batching problem. *Eur. J. Oper. Res.* **2018**, *264*, 653–664.
14. Kirpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by simulated annealing. *Readings Comput. Vis.* **1987**, *220*, 671–680.
15. Glover, F. Tabu Search—Part I. *Orsa J. Comput.* **1989**, *1*, 89–98. [[CrossRef](#)]
16. Pisinger, D.; Ropke, S. A general heuristic for vehicle routing problems. *Comput. Oper. Res.* **2007**, *34*, 2403–2435. [[CrossRef](#)]
17. Liong, C.Y.; Wan, I.; Khairuddin, O. Vehicle Routing Problem: Models and Solutions. *J. Qual. Meas. Anal.* **2008**, *4*, 205–218.
18. Bello, I.; Pham, H.; Le, Q.V.; Norouzi, M.; Bengio, S. Neural Combinatorial Optimization with Reinforcement Learning. *arXiv* **2016**, arXiv:1611.09940.
19. Dai, H.; Dai, B.; Song, L. Discriminative Embeddings of Latent Variable Models for Structured Data. *arXiv* **2016**, arXiv:1603.05629.
20. Nazari, M.; Oroojlooy, A.; Snyder, L.V.; Takáč, M. Reinforcement Learning for Solving the Vehicle Routing Problem. *Adv. Neural Inf. Process. Syst.* **2018**, *31*, 9861–9871.
21. James, J.Q.; Yu, W.; Gu, J. Online Vehicle Routing with Neural Combinatorial Optimization and Deep Reinforcement Learning. *IEEE Trans. Intell. Transp. Syst.* **2019**, *20*, 3806–3817.
22. Gasse, M.; Chételat, D.; Ferroni, N.; Charlin, L.; Lodi, A. Exact Combinatorial Optimization with Graph Convolutional Neural Networks. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 15554–15566.
23. Kool, W.; Hoof, H.V.; Welling, M. Attention, learn to solve routing problems! In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
24. Liu, Y.; Ding, W.; Yang, M.; Zhu, H.; Liu, L.; Jin, T. Distributed Drive Autonomous Vehicle Trajectory Tracking Control Based on Multi-Agent Deep Reinforcement Learning. *Mathematics* **2024**, *12*, 1614. [[CrossRef](#)]
25. Manne, A.S. A Target-Assignment Problem. *Oper. Res.* **1958**, *6*, 346–351. [[CrossRef](#)]
26. Huang, T.J. Weapon-Target Assignment Problem by Multiobjective Evolutionary Algorithm Based on Decomposition. *Complexity* **2018**, *2018*, 8623051.
27. Choi, Y.B.; Jin, S.H.; Kim, K.S.; Chung, B.D. A robust optimization approach for an artillery fire-scheduling problem under uncertain threat. *Comput. Ind. Eng.* **2018**, *125*, 23–32. [[CrossRef](#)]
28. Li, Y.; Kou, Y.; Li, Z.; Xu, A.; Chang, Y. A Modified Pareto Ant Colony Optimization Approach to Solve Biobjective Weapon-Target Assignment Problem. *Int. J. Aerosp. Eng.* **2017**, *2017*, 1746124. [[CrossRef](#)]
29. Bertsekas, D.P.; Homer, M.L.; Logan, D.A.; Patek, S.D.; Sandell, N.R. Missile defense and interceptor allocation by neuro-dynamic programming. *IEEE Trans. Syst. Man Cybern. Part A* **2000**, *30*, 42–51. [[CrossRef](#)]
30. Davis, M.T.; Robbins, M.J.; Lunday, B.J. Approximate dynamic programming for missile defense interceptor fire control. *Eur. J. Oper. Res.* **2016**, *259*, 873–886. [[CrossRef](#)]

31. Wang, J.; Luo, P.; Zhou, J.; Lan, X. Optimizing Weapon-target assignment in Air to Ground Strike Based on Adaptive Immune Genetic Algorithm. In Proceedings of the 2017 4th International Conference on Information Science and Control Engineering (ICISCE), Changsha, China, 21–23 July 2017.
32. Cha, Y.H.; Kim, Y.D. Fire scheduling for planned artillery attack operations under time-dependent destruction probabilities. *Omega* **2010**, *38*, 383–392. [[CrossRef](#)]
33. Feghhi, N.; Kosari, A.R.; Atashgah, M.A.A. A real-time exhaustive search algorithm for the weapon-target assignment problem. *Sharif Univ. Technol.* **2019**, *28*, 1539–1551.
34. Lu, Y.; Chen, D.Z. A new exact algorithm for the Weapon-Target Assignment problem. *Omega* **2021**, *98*, 102138. [[CrossRef](#)]
35. Xin, B.; Chen, J.; Peng, Z.; Dou, L.; Zhang, J. An Efficient Rule-Based Constructive Heuristic to Solve Dynamic Weapon-Target Assignment Problem. *IEEE Trans. Syst. Man Cybern. Part A* **2011**, *41*, 598–606. [[CrossRef](#)]
36. 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](#)]
37. Lee, Z.J.; Su, S.F.; Lee, C.Y. Efficiently solving general weapon-target assignment problem by genetic algorithms with greedy eugenics. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **2003**, *33*, 113–121.
38. Lee, Z.J.; Su, S.F.; Lee, C.Y. A genetic algorithm with domain knowledge for weapon-target assignment problems. *J. Chin. Inst. Eng.* **2002**, *25*, 287–295. [[CrossRef](#)]
39. Li, X.; Zhou, D.; Yang, Z.; Pan, Q.; Huang, J. A Novel Genetic Algorithm for the Synthetical Sensor-Weapon-Target Assignment Problem. *Appl. Sci.* **2019**, *9*, 3803. [[CrossRef](#)]
40. Li, L.; Liu, F.; Long, G.; Guo, P.; Bie, X. Modified particle swarm optimization for BMDS interceptor resource planning. *Appl. Intell.* **2016**, *44*, 471–488. [[CrossRef](#)]
41. Bisht, S. Hybrid Genetic-simulated Annealing Algorithm for Optimal Weapon Allocation in Multilayer Defence Scenario. *Def. Sci. J.* **2004**, *54*, 395–405. [[CrossRef](#)]
42. Bogdanowicz, Z.R.; Patel, K. Quick Collateral Damage Estimation Based on Weapons Assigned to Targets. *IEEE Syst. Man, Cybern. Syst.* **2014**, *45*, 762–769. [[CrossRef](#)]
43. Wang, C.H.; Chen, C.Y.; Hung, K.N. Toward a new task assignment and path evolution (TAPE) for missile defense system (MDS) using intelligent adaptive SOM with recurrent neural networks (RNNs). *IEEE Trans. Cybern.* **2015**, *45*, 1134–1145. [[CrossRef](#)]
44. 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.
45. Wang, Y.W. Moving-target travelling salesman problem for a helicopter patrolling suspicious boats in antipiracy escort operations. *Expert Syst. Appl.* **2023**, *213*, 118986. [[CrossRef](#)]
46. Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In Proceedings of the NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017.
47. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
48. Abuqaddom, I.; Mahafzah, B.; Faris, H. Oriented Stochastic Loss Descent Algorithm to Train Very Deep Multi-Layer Neural Networks Without Vanishing Gradients. *Knowl.-Based Syst.* **2021**, *230*, 107391. [[CrossRef](#)]
49. Bai, X.; Yan, W.; Ge, S.S. Efficient Task Assignment for Multiple Vehicles With Partially Unreachable Target Locations. *IEEE Internet Things J.* **2021**, *8*, 3730–3742. [[CrossRef](#)]

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.