drones

MDPI

Article

Cooperative Multi-UAV Task Assignment in Cross-Regional Joint Operations Considering Ammunition Inventory

Xinyong Yu [1], Xiaohua Gao [1], Lei Wang [1], Xinwei Wang [2,*], Yu Ding [3,4,5], Chen Lu [3,4,5] and Sheng Zhang [2]

[1] School of Mathematical Science, Dalian University of Technology, Dalian 116024, China; yuxinyong@mail.dlut.edu.cn (X.Y.); gaoxiaohua@mail.dlut.edu.cn (X.G.); wanglei@dlut.edu.cn (L.W.)
[2] State Key Laboratory of Structural Analysis for Industrial Equipment, Department of Engineering Mechanics, Dalian University of Technology, Dalian 116024, China; zhangs@dlut.edu.cn
[3] Science and Technology on Reliability and Environmental Engineering Laboratory, Beijing 100191, China; dingyu@buaa.edu.cn (Y.D.); luchen@buaa.edu.cn (C.L.)
[4] Institute of Reliability Engineering, Beihang University, Beijing 100191, China
[5] School of Reliability and Systems Engineering, Beihang University, Beijing 100191, China
* Correspondence: wangxinwei@dlut.edu.cn

Abstract: As combat missions become increasingly complex in both space and time, cross-regional joint operations (CRJO) is becoming an overwhelming trend in modern air warfare. How to allocate resources and missions prior to the operation becomes a central issue to improve the combat efficiency. In this paper, we focus on the cooperative mission planning of multiple heterogeneous unmanned aerial vehicles (UAVs) in a CRJO. A multi-objective optimization problem is presented with the aim of minimizing the makespan while maximizing the value expectation obtained. Moreover, it is not mandatory for each UAV to return exactly to the base which it takes off. Furthermore, in addition to the constraints commonly found in UAV mission assignment problems, the ammunition inventory at each base is also taken into account. To solve such a problem, we developed an improved genetic algorithm (IGA) with a novel chromosome encoding format. It can determine the number of attacks on a given target based on the expectations obtained, rather than being predetermined. Specifically, an efficient logic-based unlocking mechanism is designed for the crossover and mutation operations in the algorithm. Simulation results show that the developed IGA can efficiently solve the considered problem. Through numerical experimental comparisons, the algorithm proposed in this work is superior to other existing IGA-like algorithms in terms of computational efficiency.

Keywords: heterogeneous UAVs; cross-regional joint operation; task assignment; multi-objective optimization; ammunition inventory; improved genetic algorithm

Citation: Yu, X.; Gao, X.; Wang, L.; Wang, X.; Ding, Y.; Lu, C.; Zhang, S. Cooperative Multi-UAV Task Assignment in Cross-Regional Joint Operations Considering Ammunition Inventory. *Drones* **2022**, *6*, 77. https://doi.org/10.3390/drones6030077

Academic Editor: Abdessattar Abdelkefi

Received: 20 February 2022
Accepted: 11 March 2022
Published: 16 March 2022

1. Introduction

Due to the advantages of powerful execution and low risk to human life, UAVs have been widely used for various military purposes such as reconnaissance, precision strike, surveillance, communication relay, decoy, etc. [1]. Compared with a single UAV, UAVs with different capacities can achieve more complicated mission goals through cooperation. Nowadays, there is a growing consensus that mission planning, which is generally composed of task assignment and path planning, should be taken as the core technique to improve the combat efficiency [2,3].

The task assignment problem is essentially a nondeterministic polynomial hard (NP-hard) problem [4], which suggests that the scale of the problem severely impacts the solving process. The multi-UAV task assignment problem can generally be modeled as a mixed-integer linear programming (MILP), and commonly used models include vehicle routing problem (VRP) [5], multiple traveling salesman problem (MTSP) [6], multiple processors resource allocation (MPRA) [7], etc. Complicated constraints are embedded in the models to give an accurate description of the mission. The exhaustive enumerations

of MILP and the tree search method lead to prohibitive computational complexity when handling large and complex missions [8,9]. Instead, due to the powerful optimality-seeking ability and extraordinary computational stability, swarm intelligence algorithms (SIAs) have become the mainstream. Commonly used SIAs in these fields include particle swarm optimization (PSO) [10], genetic algorithm (GA) [11,12], ant colony optimization (ACO) [13,14], reactive tabu search algorithm [15] and their variations. In order to improve the solving performance, many studies combine multiple SIAs to construct novel compound methods [16,17]. In addition to the centralized algorithms mentioned above, distributed task assignment methods such as contract net [18], auction methods [19] are also popular for their advantages in real-time implementation.

When the factors considered are different, the model of the task assignment problem and the corresponding algorithm are different. These factors may include the types of UAVs, the types of tasks, and the types of constraints, etc. For the multi-UAV collaborative task assignment problem, Jia et al. [20] proposed the two-parent genetic algorithm based on the given coding scheme and a sequence number cross method. Considering the task coupling constraints, the resource constraints and the strict task precedence constraint comprehensively, Ye et al. [21] established a cooperative multiple task assignment model to minimize the makespan, and an adaptive genetic algorithm is developed to address the problem (note that makespan is often used in scheduling problems, where it represents the total execution time of UAV tasks). Taking the benefits of UAVs, the time required to execute tasks and the task load into account, Yang et al. [22] proposed a hybrid task allocation method for the multi-UAV cooperative task allocation, where the quantum genetic algorithm, the grouping optimization strategy and the simulated annealing criterion are considered in the proposed method. Zhen et al. [23] proposed an intelligent self-organized algorithm to solve the cooperative task planning problem, which the maneuverability constraints, the collision avoidance constraints, the threat avoidance constraints and the range constraints are considered. Environmental uncertainty also affects models and algorithms. Bertuccelli et al. [24] studied the real-time multi-UAV task assignment in dynamic and uncertain environment, and analyzed task assignment for heterogeneous air vehicles using a guaranteed conflict-free assignment algorithm. Considering the stochastic velocities of UAVs, Jia et al. [25] studied the cooperative multiple task assignment problem, which is formulated as a two-stage stochastic programming model, and proposed an improved genetic algorithm. A task assignment problem that many criteria values essential to task assignment were random or fuzzy was studied in the literature [26], and a novel approach based on stochastic multicriteria acceptability analysis method was proposed in this study. There are many other related studies on the consideration of uncertain factors in the task assignment of multiple UAVs [27–30], which will not be described in detail here.

It should be noted that general crossover and mutation operations in SIAs are likely to generate chromosome with constraint violation, which results in the deadlock (or "lock" for short) phenomenon. Each algorithm has its mechanism to fix these infeasible solutions, known as the unlocking process. During the implementation, a large portion of computational time is consumed by the unlocking process. For example, unlocking process is achieved by keeping swapping two columns of the chromosome randomly until all constraints are satisfied in [31]. It results in an unacceptable computational burden as the problem scale increases. Hence, developing efficient unlock mechanisms is significantly beneficial for improving the computational efficiency.

Considering the complexity in the both spatial and time domain of modern air warfare, CRJOs are frequently executed to cover the battlefield with a vast range. It may require heterogeneous UAVs distributed at different bases to constitute a fleet or a swarm to achieve certain mission goals cooperatively. For example, aiming to attack an island, the carrier-based combat UAVs on the sea and the bomber UAVs at home should cooperate, where the former tries to win the air superiority and the latter executes bombing missions. However, task assignment problems under the multi-base scenario are rarely seen in previous research. Hence, in this paper, we address the cooperative task assignment for multiple heterogeneous UAVs in

CRJOs by an IGA. First, the task assignment problem is formulated into a multi-objective MILP. In the developed mathematical model, heterogeneous UAVs set off from different airports and they are not required to return exactly to the airport they depart. Three types of tasks (i.e., classification, attack, and verification) are expected to be executed sequentially on each target. Each target is prescribed with a fixed value, and it allows multiple attacks on a certain target. Two contradictory objectives, i.e., minimization of execution time and maximization of obtained value expectation are set. Practical constraints such as maximum ammunition load, time constraints, and so on are imposed. In addition, the ammunition inventory of each airport, which is rarely seen in the existing literature, is also taken into consideration. To solve the problem, an IGA is developed. It deals with various constraints in the formulated problem through a unique encoding of chromosomes and specifically designed resource constraint sets (RCSs). Special crossover and mutation operations can adaptively adjust the number of attacks on the target according to the fitness function. Furthermore, the logic-based unlocking method greatly reduces the time consumption of unlocking process and effectively improves the efficiency of the algorithm.

To the best of our knowledge, this is the first paper to address the problem of multi-heterogeneous UAV mission assignment with multiple airports and munitions inventory constraints. The main contributions of this paper can be summarized as follows: (i) a novel IGA with a special encoding format and genetic operations is developed to solve the problem; (ii) an efficient logic-based unlocking mechanism is designed by using RCS; and (iii) the number of attacks on the target can be determined by the IGA without the need of prior setting.

The remainder of the paper is organized as follows. Section 2 describes the multi-airport task assignment problem. Section 3 establishes the corresponding MILP model. An IGA to solve the problem is developed in Section 4 and then validated in Section 5. Finally, Section 6 concludes the paper.

2. Problem Description

Suppose that N_T detected targets with known positions in the battlefield. Let

$$T := \{T_1, T_2, \cdots, T_{N_T}\}$$

denote the set of targets. Target T_k is assigned with a value $Value_{T_k}$. For a target, a series of sub-tasks need to be performed on it until the desired goal is reached. Each target contains 3 types of tasks, that is, classification task, attack task and verification task. Let k represent the task type, $k = 1, 2, 3$, respectively, represent classification task, attack task and verification task. In particular, let $k = 0$ represents that the UAV returns to the airport. In addition, N_k represents the number of task types, thus, $N_k = 4$.

For simplicity of description, the classification, attack and verification tasks are abbreviated as C, A, V, respectively. Let S_{T_j} be the task set of target T_j,

$$S_{T_j} := \{C, A_1, A_2, \cdots, A_{N_A^j}, V\}, j = 1, 2, \cdots, N_T,$$

where N_A^j is the number of attack tasks of target T_j, which depends on the value of the target T_j. In this paper, the values of $N_A^j, j = 1, 2, \cdots, N_T$ are quantities to be optimized rather than being prescribed. Since multiple airports are considered, we suppose that

$$P := \{P_1, P_2, \cdots, P_{N_P}\}$$

denote the set of airports, where $P_m, m = 1, 2, \cdots, N_P$ is the m-th airport.

According to the actual situation, for the same target, classification, attack and verification tasks should be executed sequentially. Let t_C^{Tj}, $t_{A_l}^{Tj} (l = 1, 2, \cdots, N_A^j)$, t_V^{Tj} denote the execution time of the task C, A_l, V of target T_j, respectively. Then, we have,

$$t_C^{Tj} < t_A^{Tj}, \hat{t}_A^{Tj} < t_V^{Tj}, j = 1, 2, \cdots, N_T,$$

where $t_A^{Tj} := \min\{t_{A_l}^{Tj} | l = 1, 2, \cdots, N_A^j\}$, $\hat{t}_A^{Tj} := \max\{t_{A_l}^{Tj} | l = 1, 2, \cdots, N_A^j\}$.

Three types of UAVs, i.e., surveillance UAV, combat UAV and munition UAV cooperate to conduct the CRJO. Among them, surveillance UAVs can perform the classification and verification tasks, combat UAVs can perform all types of tasks, and munition UAVs can only perform the attack task. Let

$$U := \{U_1, U_2, \cdots, U_{N_U}\}$$

denote the set of UAVs, where U_i is the i-th UAV. The tasks assigned to a certain UAV constitute the task set of the UAV. Each UAV loads ammunition at the airport and then takes off to perform its tasks. Suppose that the task set of UAV U_i is

$$M_{U_i} := \{T_1^{U_i}, T_2^{U_i}, \cdots, T_{N_{U_i}}^{U_i}\},$$

where $T_n^{U_i}$, $n = 1, 2, \cdots, N_{U_i}$ is the n-th task of UAV U_i. Each UAV completes the tasks one by one in the order of task assignment. Let $t_n^{U_i}$ $(n = 1, 2, \cdots, N_{U_i})$ denote the execution time of the n-th task of UAV U_i. Then, we have,

$$t_1^{U_i} < t_2^{U_i} < \cdots < t_n^{U_i} < \cdots < t_{N_{U_i}}^{U_i}, i = 1, 2, \cdots, N_U.$$

There is the possibility, in practical situations, that a UAV does not completely destroy a target by one attack. In this paper, we assume that each UAV has different success probabilities when attacking different targets. Let the probability of target T_j $(j = 1, 2, \cdots, N_T)$ being destroyed by UAV U_i $(i = 1, 2, \cdots, N_U)$ is written as $PS_{i,j}$. We only consider the success probability of the attack task.In addition, for the cross-regional joint operations problem, we also make the following assumptions:

1. To reduce the complexity of the problem, the environment considered is a two-dimensional plane;
2. UAVs all fly at different altitudes, and there will be no collision even if the trajectories overlap;
3. The success rate in performing classification and verification tasks is 100%.

For the above problem description and assumptions, Figure 1 shows an example of cross-regional joint operations for 2 airports, 2 targets and 3 UAVs. Among them, U_1, U_2 and U_3 are the survival UAV, munition UAV and combat UAV, respectively.

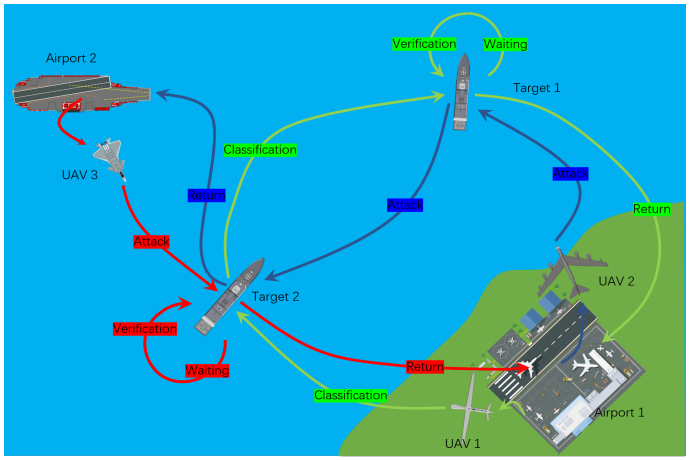

Figure 1. An illustration of cross-regional joint operations.

3. Mathematical Model

3.1. Map Discretization

The connected digraph is introduced for map discretization. The locations of targets and airports, and the values of heading angles are involved in the information of the vertices. A UAV can approach or leave an airport or a target with different orientations. In order to facilitate the calculation, the continuous heading angle is discretized. The heading angle is discretized by using the form of equal division, and the set of discretized heading angles is

$$\Psi = \{\psi_i | \psi_i = \frac{2\pi i}{N_\psi}, i = 0, 1, \cdots, N_\psi - 1\},$$

where N_ψ represents the number of possible heading angles.

Based on the above description, the set of vertices is

$$V = \{P_1, P_2, \cdots, P_l, \cdots, P_{N_P}, T_1, T_2, \cdots, T_j, \cdots, T_{N_T}\},$$

where $l = 1, 2, \cdots, N_P$, $j = 1, 2, \cdots, N_T$. Let N_V is the number of vertices. Obviously, one has $N_V = N_P + N_T$. The set of edges is

$$E = \{(V_i, V_j) | V_i, V_j \in V, i, j = 0, 1, \cdots, N_V, i \neq j\},$$

where V_i is the i-th element of V. Then, the connected digraph is given as

$$CG = \{V, E\}.$$

3.2. Objective Function

In the task assignment problem model, there are two parts are considered in the objective function. These two parts are the makespan and the expectation of value obtained, and the details are given as follows.

(1) The makespan

When evaluating a task assignment plan, minimizing the makespan is the primal indicator. The makespan depends on the maximum completion time among all UAVs, and it can be expressed as

$$J_1 = \max_{u \in U} \{ \sum_{k=1}^{N_k} \sum_{i=1}^{N_V} \sum_{j=1}^{N_V} \sum_{\hat{j}=0}^{N_\psi - 1} \sum_{\hat{i}=0}^{N_\psi - 1} X^{u,k}_{((v_i,\psi_{\hat{i}}),(v_j,\psi_{\hat{j}}))} \cdot d^u_{((v_i,\psi_{\hat{i}}),(v_j,\psi_{\hat{j}}))} / Velocity_u \}, \quad (1)$$

where $X^{u,k}_{((v_i,\psi_i),(v_j,\psi_j))} \in \{0, 1\}$, $\forall v_i, v_j \in V$, $\forall \psi_{\hat{i}}, \psi_{\hat{j}} \in \Psi$, $u \in U$ is a binary decision variable. If UAV U_u executes task type k from $(v_i, \psi_{\hat{i}})$ to $(v_j, \psi_{\hat{j}})$, then $X^{u,k}_{((v_i,\psi_{\hat{i}}),(v_j,\psi_{\hat{j}}))} = 1$; otherwise, $X^{u,k}_{((v_i,\psi_{\hat{i}}),(v_j,\psi_{\hat{j}}))} = 0$. And $d^u_{((v_i,\psi_{\hat{i}}),(v_j,\psi_{\hat{j}}))}$ is the the distance of the Dubins path of UAV U_u from $(v_j, \psi_{\hat{j}})$ to $(v_j, \psi_{\hat{j}})$. For more information about Dubins path, please refer to the literature [32]. $Velocity_u$ represents the speed of UAV U_u.

(2) The expectation of the value obtained

Maximizing the expectation of the value obtained is another important indicator to evaluate a task assignment scheme. Here, we try to minimize the opposite of the expectation of the value obtained,

$$J_2 = -\sum_{u \in U} \sum_{l=1}^{N_V} \sum_{j=1}^{N_V} \sum_{\hat{j}=0}^{N_\psi - 1} \sum_{\hat{i}=0}^{N_\psi - 1} (X^{u,2}_{((v_i,\psi_{\hat{i}}),(v_j,\psi_{\hat{j}}))} \cdot Value_{T_j} \cdot PS_{u,j}). \quad (2)$$

This is a multi-objective optimization problem. The objective function we expect is the weighted sum of J_1 and J_2. The orders of magnitude of two objectives can be of large

difference, and direct weighted sum is meaningless. In this paper, the standard values $standard_1$ and $standard_2$ are taken to make J_1 and J_2 dimensionless. The specific form is given as follows,

$$\min J = \alpha \cdot J_1/standard_1 + \beta \cdot J_2/standard_2, \tag{3}$$

where α, β are the weights that satisfy $\alpha + \beta = 1$.

Remark 1. *How to select the standard value in the algorithm is described in detail in Section 4.4.2 of this article. For some reasons, the fitness function in the actual calculation cannot directly take the objective function, but requires operations such as translation. Related details are also provided in Section 4.4.2.*

3.3. Constraints

Based on the actual situation and the multi-airport situation, the following constraints need to be considered in the task assignment problem.

(1) The task requirement constraint.

Multi-UAV cooperative constraints are common constraints in the UAV task assignment problem. Based on the above description, UAVs are required to perform all tasks in coordination, that is, each task will be assigned to one UAV to perform. The mathematical representation of this constraint is given as follows,

$$\sum_{u=1}^{N_U}\sum_{k=1}^{N_k}\sum_{i=1}^{N_V}\sum_{\hat{i}=0}^{N_\psi-1}\sum_{\hat{j}=0}^{N_\psi-1} X^{u,k}_{((v_i,\psi_{\hat{i}}),(T_t,\psi_{\hat{j}}))} = N_{T_t}, t = 1, 2, \cdots, N_T. \tag{4}$$

N_{T_t} represents the number of tasks that target t needs to perform. In some cases N_{T_t} may be 0, such as the ammunition is insufficient or the target value is not high enough.

(2) The ammunition load constraints.

Due to various reasons, the amount of ammunition carried by the UAV is limited, so the number of attack tasks of the UAV cannot exceed its ammunition load. This constraint can be modeled as follows,

$$\sum_{i=1}^{N_V}\sum_{j=1}^{N_V}\sum_{\hat{i}=0}^{N_\psi-1}\sum_{\hat{j}=0}^{N_\psi-1} X^{u,2}_{((v_i,\psi_{\hat{i}}),(v_j,\psi_{\hat{j}}))} \le n^a_u, \forall u \in \mathrm{U}, \tag{5}$$

where n^a_u represents the ammunition capacity of UAV U_u. In addition, in order to reduce the risk of attack task and increase the value expectation, we often stipulate that the same UAV will only attack the same target once.

(3) The time constraints.

In this article, each target contains three types of tasks. Based on the above description, these three types of tasks for the same target must be completed in order.

$$t^{T_t}_C < t^{T_t}_A < t^{T_t}_V, t = 1, 2, \cdots, N_T. \tag{6}$$

When hitting the same target multiple times, $t^{T_t}_A$ is not just an element. It will be a set $T^{T_t}_A = \{t^{T_t}_{A_1}, \ldots, t^{T_t}_{A_n}\}$. Under this circumstance, we only need to ensure that the following formula holds.

$$t^{T_t}_C < \min(T^{T_t}_A) \quad \text{and} \quad \max(T^{T_t}_A) < t^{T_t}_V, t = 1, 2, \cdots, N_T. \tag{7}$$

(4) The UAVs recycling constraints.

In the actual battlefield, UAVs that have completed their tasks need to return to the airport. There are multiple airports considered in this paper, such as aircraft carrier formations and island bases. Therefore, for the multi-airport problem, any UAV that departs from any airport needs to return to any airport in the end. The mathematical representation of this constraint is given as follows,

$$\begin{aligned}&\sum_{n=1}^{N_P}\sum_{k=1}^{N_k}\sum_{j=1}^{N_T N_\psi}\sum_{l=0}^{N_\psi-1} X^{u,k}_{((P_n,\psi_{P_n}),(v_j,\psi_l))}\\&+\sum_{n=1}^{N_P}\sum_{k=1}^{N_k}\sum_{j=1}^{N_T N_\psi}\sum_{l=0}^{N_\psi-1} X^{u,k}_{((v_j,\psi_l),(P_n,\psi_{P_n}))}=2,\forall u\in U.\end{aligned} \tag{8}$$

(5) The multi-airport ammunition constraints.

In the multi-airport problem, the number of ammunition at each airport is limited, so the task assignment problem involves the multi-airport ammunition constraint. Therefore, the number of tasks performed by UAVs at each airport cannot exceed the total number of ammunition at that airport. That is,

$$\sum_{i=1}^{N_V}\sum_{j=1}^{N_V}\sum_{\hat{i}=0}^{N_\psi-1}\sum_{\hat{j}=0}^{N_\psi-1} X^{u,2}_{((v_i,\psi_{\hat{i}}),(v_j,\psi_{\hat{j}}))}\le n^a_p,\quad u\in P_p \tag{9}$$

where n^a_p, $p=1,2,\cdots,N_P$ represents the number of ammunition of the p-th airport.

3.4. Formulation of the Optimization Problem

Based on the above description, the model of the task assignment problem involving multiple airports is described as follows,

$$\begin{aligned}&\min J=\alpha\cdot J_1/standard_1+\beta\cdot J_2/standard_2\\&s.t.\sum_{u=1}^{N_U}\sum_{k=1}^{N_k}\sum_{i=1}^{N_V}\sum_{\hat{i}=0}^{N_\psi-1}\sum_{\hat{j}=0}^{N_\psi-1} X^{u,k}_{((v_i,\psi_{\hat{i}}),(T_t,\psi_{\hat{j}}))}=N_{T_t},t=1,2,\cdots,N_T,\\&\sum_{i=1}^{N_V}\sum_{j=1}^{N_V}\sum_{\hat{i}=0}^{N_\psi-1}\sum_{\hat{j}=0}^{N_\psi-1} X^{u,2}_{((v_i,\psi_{\hat{i}}),(v_j,\psi_{\hat{j}}))}\le n^a_u,\forall u\in \mathrm{U},\\&t^{T_t}_{\mathrm{C}}<t^{T_t}_{A}<t^{T_t}_{V},t=1,2,\cdots,N_T,\\&\sum_{n=1}^{N_P}\sum_{k=1}^{N_k}\sum_{j=1}^{N_T N_\psi}\sum_{l=0}^{N_\psi-1} X^{u,k}_{((P_n,\psi_{P_n}),(v_j,\psi_l))}\\&+\sum_{n=1}^{N_P}\sum_{k=1}^{N_k}\sum_{j=1}^{N_T N_\psi}\sum_{l=0}^{N_\psi-1} X^{u,k}_{((v_j,\psi_l),(P_n,\psi_{P_n}))}=2,\forall u\in U,\\&\sum_{i=1}^{N_V}\sum_{j=1}^{N_V}\sum_{\hat{i}=0}^{N_\psi-1}\sum_{\hat{j}=0}^{N_\psi-1} X^{u,2}_{((v_i,\psi_{\hat{i}}),(v_j,\psi_{\hat{j}}))}\le n^a_p,\quad u\in P_p,\\&X^{u,k}_{((v_i,\psi_i),(v_j,\psi_j))}\in\{0,1\},\forall v_i,v_j\in V,\forall\psi_{\hat{i}},\psi_{\hat{j}}\in\Psi,\forall u\in U.\end{aligned} \tag{10}$$

4. An IGA with Flexible Unlock Mechanism

4.1. Chromosome Encoding

According to the above description, in order to make the genetic algorithm suitable for solving the multi-airport task assignment problem, the algorithm needs to be modified. The modified chromosome coding is given in this section.

4.1.1. Encoding Format

In the original genetic algorithm, each chromosome represents a feasible solution. Similarly, this method is also used in the improved algorithm. We know that although the task assignment problem is an NP-hard problem, it is essentially to find the corresponding relationship between tasks and UAVs and the sequence of these corresponding relationships. Therefore, each given plan contains two aspects: 1. the corresponding relationship between UAVs and tasks; 2. the order in which UAVs perform tasks. Based on this idea, the following chromosome fragment can be constructed.

Figure 2 is a chromosome fragment, and the third row represents the number of the UAV. Figure 2 represents the plan of UAV U_1 in the chromosome. A complete plan is executed by multiple UAVs collaboratively, so a complete chromosome contains several fragments similar to Figure 1. In addition, in each fragment, the first and second rows are the task number and task type, respectively. The information in the first three rows forms the above-mentioned corresponding relationship between UAVs and tasks. The order of tasks corresponding to the chromosome from left to right is the order of execution of tasks. Because the distance is calculated by using the Dubins path, the information in Dubins path is added to the chromosome. The fourth row is the angle when the UAV approaches the target. The fifth and sixth rows are the angles of the UAV when it leaves and returns to the airport. In addition, the background of multiple airports also makes the chromosome need to include the necessary airport information. The seventh row is the number of the airport where the UAV returns, and the eighth row is the number of the airport where the UAV departs, and the ninth row is the location of the UAV in the airport. For example, in Figure 2, the eighth and ninth rows in the chromosome indicate that UAV U_1 is at position 1 of airport 1. If UAV U_2 is also at Airport 1, then the values of the eighth and ninth rows corresponding to the UAV U_2 are 1 and 2, respectively. This agreement is to facilitate program calculations. Figure 3 is a complete chromosome,which consists of several fragments illustrated in Figure 2.

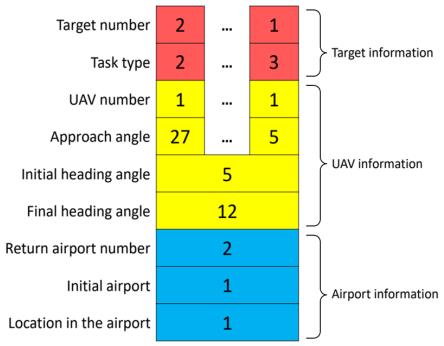

Figure 2. Chromosome fragment.

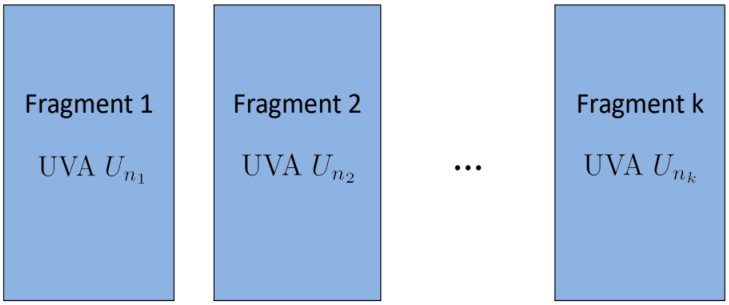

Figure 3. Chromosome.

4.1.2. Initialization Strategy

Because the CRJO problem contains many constraints, there are many infeasible solutions in our randomly generated chromosomes, and a large number of infeasible solutions

will affect the speed of algorithm convergence. Therefore, in the process of generating chromosomes, we must avoid some infeasible solutions. A resource constraint set is defined in the algorithm of this paper, which is generated according to the ammunition reserves owned by the airport and the docked UAVs, and represents the capabilities of each airport. Figure 4 shows the process of assigning task k to execute the UAV in the encoding of chromosomes. The advantage of assigning tasks through resource constraint set encoding is that the resulting chromosomes satisfy resource constraints. Therefore, the resource constraint set needs to be used in the subsequent cross-mutation process.

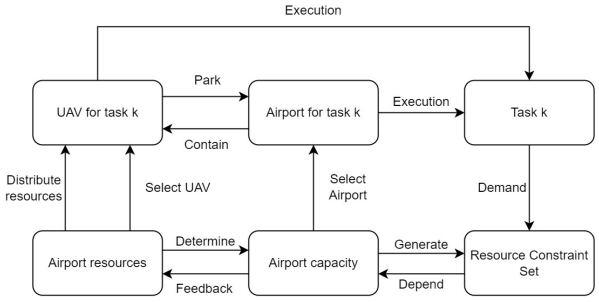

Figure 4. Chromosome initialization process.

Figure 4 completes the coding of lines 1–3 of the initial chromosome, various heading angles in lines 4–6 are randomly given, and the airport information in lines 7–9 is initially determined. The chromosomes in Figure 3 are obtained by arranging the generated chromosomes according to the UAV number. Since this article considers the situation of ammunition scarcity, first, we need to judge the size of $n_e^a = \max\{\text{airport total ammunition, sum of all UAV loads}\}$ and N_T. If the ammunition is sufficient, assign the UAV to the task of the N_T targets, otherwise, select n_e^a targets by roulette according to the target value, and assign the UAV to perform their tasks.

4.2. Crossover Operator

The crossover operator is an important part of the genetic algorithm. In essence, the crossover operation is to exchange the information of the two chromosomes. This paper, it is to exchange the correspondence between the tasks in the two schemes and the UAV. First, this paper uses the roulette algorithm to select two good parents according to the fitness function value. Second, a single-point crossover method is used to randomly select a point from parent 1 as the crossover starting point. All information from the beginning to the end of parent 1 is exchanged with the corresponding information of parent 2. Since the attack task will involve resource constraints, the crossover of attack information is different from that of classification evaluation.

Furthermore, under the background assumptions of this paper, the number of attacks against the target is no longer deterministic. In the initial process of chromosomes, the number of attacks on the target is 1, but in the subsequent calculation, the algorithm will determine the number of attacks on the target by itself according to the value of the fitness function. Therefore, this will lead to the phenomenon of crossover operation of chromosomes of different sizes during the crossover process, and changes in chromosome dimensions may also occur during the crossover process. The crossover method in this paper takes the dimension of the first chromosome as a reference, so different dimensions and changes in dimensions will not affect the crossover operation.

4.2.1. Crossover of Attack Genes

This section gives the specific operation of single-point crossover of attack genes. Since the attack task involves ammunition constraints, the crossover operation of attack genes will be divided into two cases by discussing whether the attackable UAV set is empty or not. Because the crossover process involves the crossover of chromosomes of different

dimensions, the set of attackable UAVs corresponding to one chromosome may be empty, while the other is not. In this case, the chromosome corresponding to the empty set is defined as parent 1. As long as the attackable UAV set of parent 1 is empty, the attackable UAV set is considered empty (the attackable UAV set corresponding to another chromosome may not be empty at this time). We call the constraint that the same target can only be attacked once by the same UAV as constraint A. A detailed description of the single-point crossover of the attack gene is as follows.

- Attackable set is not empty.

If the attackable set is not empty, then the tasks information remain unchanged, and the exchange of the information of the two chromosomes is achieved by exchanging the information of the UAVs performing the attack tasks in the two chromosomes. The specific steps are shown in Algorithm 1, In addition, Table 1 is a notation description of the algorithm in this section.

A simple example is given to assist readers in understanding the operation flow of the crossover operation when the set of attackable UAVs is not empty. In this example, there are 2 airports, 3 UAVs, and 2 targets. First, two parents are obtained through the roulette algorithm, as shown in the following figure.

Table 1. Parameters in the Crossover Algorithm.

Notation	Explanation
$\mathbf{chromosome_i}$	Chromosome of parent i.
$\mathbf{attackableUAV_i}$	The set of attackable UAVs for parent i.
$\mathbf{detectableUAV_i}$	The set of detectable UAVs for parent i.
$\mathbf{Hn_i}$	The set of heading angles of parent i.
T_{t_0}	In chromosome 1, the cross column target is marked as T_{t_0}.
U_{u_0}	In chromosome 1, the UAV performing the cross-column task is denoted as U_{u_0}
$\mathbf{CA_1}$	In Chromosome 1, except for T_{t_0}, the other targets that were attacked were executed.
$\mathbf{CA_2}$	$\{T_{t_0}\}$
$\mathbf{CB_1}$	In chromosome 2, except for the target of the attack task executed by U_{u_0}, other targets are executed the attack task.
$\mathbf{CB_2}$	In Chromosome 2, the target of the attack mission executed by U_{u_0}.
$\mathbf{CI_1}$	Intersection of $\mathbf{CA_1}$ and $\mathbf{CB_2}$.
$\mathbf{CI_2}$	Intersection of $\mathbf{CA_2}$ and $\mathbf{CB_1}$.

As shown in the Figure 5, the orange column in parent 1 is the column for single-point crossover operation, and the UAV performing the attack task in this column is U_2. The attack resource constraint set of parent 1 is $\mathbf{attackableUAV_1} = \{1, 2, 3\}$, it means U_1, U_2, U_3 that can also perform attack tasks under the task assignment plan of parent 1. The columns of attack tasks in Parent 2 is marked in green in the Figure 5, and it is represented by a set as $attack2 = \{1, 2\}$. The attack resource constraint set of parent 2 can also be expressed as $\mathbf{attackableUAV_2} = \{1, 2, 3\}$. Remove 2 from $\mathbf{attackableUAV_1} = \{1, 2, 3\}$, and then take the intersection with $attack2$, which is expressed as $I = \mathbf{attackableUAV_1}/\{2\} \cap attack2 = \{1\}$. Because 1 is in $\mathbf{attackableUAV_2}$, so the number in the red position of parent 1 can be exchanged with the number in the blue position of parent 2 in the Figure 5.

After swapping the UAVs of the crossed columns of the two chromosomes, it is also necessary to swap the approach angle and other information, and finally get the result shown in the Figure 6. This is the process of single-point crossover operation of attack genes when resources are sufficient.

Algorithm 1 Crossover operations where the attackable set is not empty

Input:
chromosome$_1$ $= c_1$, **attackableUAV$_1$** $= aA_1$,
chromosome$_2$ $= c_2$, **attackableUAV$_2$** $= aA_2$
Output:
c_1, aA_1, c_2, aA_2

Find out all the attack tasks of c_2, marked as $attack_2$.
Remove the UAV number a of the exchange line of c_1 from aA_1. In addition, let $I = aA_1 \cap attack_2$.
if $I \neq 0$ **then**
    Randomly select an element b from I.
    while b does not satisfy constraintA **do**
        Randomly select an element b from I.
    if a is not in aA_2 or a does not satisfy constraintA **then**
        Randomly select an element c from aA_2.
        $a = c$
    Change b into the cross column of c_1, and change a into the corresponding column of c_2.
    Update aA_1 and aA_2.
return c_1, aA_1, c_2, aA_2

1	1	2	2	2	1
1	2	1	3	2	3
1	2	3	1	2	3
22	35	27	7	23	13

Initial parent1

1	1	2	1	2	2
1	3	3	2	2	1
1	1	1	2	1	1
26	15	6	17	34	13

Initial parent2

Figure 5. The initial chromosome when the attackable set is not empty.

1	1	2	2	2	1
1	2	1	3	2	3
1	2	3	1	1	3
22	35	27	7	34	13

Parent 1

1	1	2	1	2	2
1	3	3	2	2	1
1	1	1	2	2	1
26	15	6	17	23	13

Parent 2

Figure 6. Result when the attackable set is not empty.

- Attackable set is empty.

When the attackable set is empty, then all the ammunition are allocated, so the operation of changing the UAV cannot be performed. According to the previous description, the nature of crossover operation is to exchange the corresponding relationship between the targets and the UAVs in the two chromosomes. Therefore, when the attack set is empty, the UAVs that perform the attack task can be fixed, and the targets being attacked in the two parents can be exchanged.

Table 1 is a notation description of the algorithm in this section. If **CI$_1$** is not an empty set, when the crossover operation is performed on chromosome 1 and chromosome 2, chromosome 1 can refer to the information of chromosome 2 for internal exchange. Internal

exchange is a random selection of an element of $\mathbf{CI_1}$, which is a target that the UAV U_{u_0} did not perform on chromosome 1, but according to the definition of the set, this target is chromosome 2 The target of the attack mission performed by U_{u_0}. Exchange this target with the target executed by U_{u_0} in chromosome 1, so as to indirectly achieve the purpose of exchanging information with chromosome 2. Similarly, if $\mathbf{CI_2}$ is not an empty set, chromosome 2 can also refer to the information of chromosome 1 for internal exchange. We refer to the restriction of internal exchange as Constraint B.

The specific steps are shown in Algorithm 2.

Algorithm 2 Crossover operations where the attack set is empty

Input:
chromosome$_1$ $= c_1$, **chromosome$_2$** $= c_2$,
Output:
c_1, c_2

Calculate $\mathbf{CA_1}$,$\mathbf{CA_2}$,$\mathbf{CB_1}$,$\mathbf{CB_2}$ according to c_1, c_2.
if $\mathbf{CB_2} \neq \emptyset$ **then**
    Calculate $\mathbf{CI_1}$,$\mathbf{CI_2}$.
    $Complete = 0$
    while $Complete = 0$ **do**
        if $\mathbf{CI_1} \neq \emptyset$ and $\mathbf{CI_1} \neq \emptyset$ **then**
            Randomly select an element a from $\mathbf{CI_1}$, elect an element b from $\mathbf{CI_2}$.
            if a, b satisfies constraintB **then**
                c_1 and c_2 are exchanged internally accordingly.
                $Complete = 1$
            else
                Remove the corresponding elements from $\mathbf{CI_1}$,$\mathbf{CI_2}$.
        if $\mathbf{CI_1} \neq \emptyset$ and $\mathbf{CI_2} = \emptyset$ **then**
            Randomly select an element a from $\mathbf{CI_1}$.
            if a satisfies constraintB **then**
                c_1 exchanges internally.
                Replace c_1 cross column elements into c_2, and adjust c_2.
                $Complete = 1$
            else
                Remove a from $\mathbf{CI_1}$.
        if $\mathbf{CI_1} = \emptyset$ and $\mathbf{CI_2} \neq \emptyset$ **then**
            Randomly select an element b from C_2.
            Traverse $\mathbf{CB_2}$ if c_2 can be exchanged internally, record $flag = 1$;
            if $flag = 1$ **then**
                c_2 exchanges internally.
                Replace c_2 cross column elements into c_1, and adjust c_1.
                $Complete = 1$
            else
                Remove the corresponding elements from $\mathbf{CI_2}$.
        if $\mathbf{CI_1} = \emptyset$ and $\mathbf{CI_2} = \emptyset$ **then**
            Replace c_2 cross column elements into c_1, and adjust c_1.
            Replace c_1 cross column elements into c_2, and adjust c_2.
            $Complete = 1$
return c_1, c_2

In order to facilitate readers to understand the above-mentioned crossover process, take the situation that resources are insufficient as an example to illustrate how the crossover operation is performed. The example contains 2 airports, 4 targets, 3 UAVs, and each airport has 1 ammunition. The two parents are given as follows:

The orange column of parent 1 in the Figure 7 is the column where parent 1 performs single-point crossover operation, and the target number that needs to be changed is marked in red. Obviously, $\mathbf{CA_1} = \{1\}$, $\mathbf{CA_2} = \{3\}$, $\mathbf{CB_1} = \{2\}$, $\mathbf{CB_2} = \{1\}$, $\mathbf{CI_1} = \{1\}$, $\mathbf{CI_2} = \emptyset$. The UAV that performs the attack task in the crossed column of parent 1 is U_2. In parent 2, we also use orange to mark the column that U_2 performs the attack task, and the target number is marked in blue. As shown in the Figure 7, there is only one column for U_2 to execute the attack, so this column can only be used as the crossed column of parent 2. If there are multiple columns of U_2 to perform the attack task in Parent 2, it is necessary to use the information of $\mathbf{CI_1}$, $\mathbf{CI_2}$ to select the crossed columns according to the algorithm steps. In this example, $\mathbf{CI_1} = \{1\}$ is not empty, so first determine whether parent 1 can be exchanged internally. We mark the column in parent 1 that performs attack tasks on target 1 in green, and the target is marked in blue. We found that the numbers in the red block and the blue block in parent 1 can still satisfy the constraints after being exchanged, so parent 1 can be exchanged internally. Since $\mathbf{CI_2}$ is an empty set, the number 3 can only be exchanged into the blue block of parent 2 to replace the previous number 1. After this treatment, there are no attack tasks of target 1 in parent 2, so the classification task and verification task (the gray column in parent 2) are meaningless, so they need to be deleted in the subsequent operations.

1	1	3	3	3	1
1	3	1	3	2	2
1	1	3	1	2	3
18	16	9	28	18	31

Initial parent 1

1	1	2	1	2	2
1	3	3	2	1	2
1	1	1	2	1	3
35	29	26	1	32	25

Initial parent 2

Figure 7. The initial chromosome when the attackable set is empty.

The result obtained after the two chromosomes are crossed is shown in the Figure 8. The target 1 and target 3 in parent 1 are interchanged. Change the previous target 1 in parent 2 to target 3, because there was no attack task of target 3 in the previous chromosome, so after adding the attack task of target 3, the classification task and verification task of target 3 are need to be added, such as the red columns in parent 2. The above process completes the single-point crossover operation for the attack task in the case of insufficient resources.

1	1	3	3	1	3
1	3	1	3	2	2
1	1	3	1	2	3
18	16	9	28	18	31

Parent 1

2	3	2	2	3	3
3	2	1	2	1	3
1	2	1	3	3	3
26	1	32	25	4	25

Parent 2

Figure 8. Result when the attackable set is empty.

4.2.2. Crossover Operations of the Classification Genes and the Verification Genes and the Heading Angles

Because the ammunition constraints do not need to be considered in the classification and verification tasks, so their crossover method is simple. The crossover operation is similar to the simplified version of the crossover operation where the attackable set is not empty. The specific steps are shown in Algorithm 3.

Algorithm 3 Crossover operator for classification and verification tasks

Input:
$\mathbf{chromosome_1} = c_1$, $\mathbf{detectableUAV_1} = dA_1$,
$\mathbf{chromosome_2} = c_2$, $\mathbf{detectableUAV_2} = dA_2$
Output:
c_1, dA_1, c_2, dA_2

1: Find all UAVs that perform classification and verification tasks in c_2, marked as $detect_2$.
2: Remove the UAV number a of the exchange line of c_1 from dA_1. In addition, let $I = dA_1 \cap detect2$.
3: **if** $I \neq \emptyset$ **then**
4: Randomly select an element b from I.
5: Change b into the cross column of c_1, and change a into the corresponding column of c_2.
6: Update dA_1 and dA_2.
7: **return** c_1, dA_1, c_2, dA_2

In addition, in the fifth step of the crossover operation, the initial and final heading angles of the two chromosomes need to be exchanged.The specific steps are shown in Algorithm 4.

Algorithm 4 The heading angle crossover operator

Input:
$\mathbf{chromosome_1} = c_1$, $\mathbf{chromosome_2} = c_2$
Output:
c_1, c_2

1: Count the number of heading angles of c_1 and c_2 and record them as $\mathbf{Hn_1}$ and $\mathbf{Hn_2}$, respectively.
2: $Hn_{min} = \min\{|\ \mathbf{Hn_1}\ |, |\ \mathbf{Hn_2}\ |\}$.
3: $R = random(0, Hn_{min})$.
4: Exchange the first R heading angles of c_1 and c_2.
5: **return** c_1, c_2

The above is all about performing crossover operations on a pair of chromosomes.

4.3. Mutation Operator

In this paper, mutation operation can not only prevent the algorithm from falling into a local optimum, but also change the dimension of chromosomes by adding or deleting attack tasks for a certain target. The mutation operation mainly consists of two parts. The probability P_{M_i} of the occurrence of a given mutation. When the generated random number is less than P_{M_i}, perform related operations. First of all, to make the attack times of each solution to the target not fixed, it is necessary to increase or decrease the attack times of a certain target during the mutation process. Second, to avoid falling into a local optimum, it is necessary to change the task sequence of the random UAVs in the scheme and the various heading angles of the randomly selected UAVs.

We want to increase the number of attacks on high-value targets when each target has value. In addition, we want to drop low value targets when ammo is limited. For

the selection method of high-value targets, the value of the target is used as the standard, and high-value targets are selected through roulette. In the case of insufficient resources, repeated attacks on some high-value targets meant that some low-value targets had to be abandoned. This paper takes the inverse of the target value and selects low-value targets through the roulette method. The Algorithm 5 gives the algorithm flow for the first part of the mutation.

We call the constraint A that the same UAV can only attack the same target once and the total number of attacks $\sum_{i=1}^{N_A} \min\{n_i^a, \sum_{j=1}^{N_{U_i}} n_{j,i}^a\}$. In addition, we call the attack task whose resource constraint set can guarantee the increase of the corresponding target as constraint C.

In order to facilitate the readers to understand the above process, a simple example is given here. The example contains 2 airports, 4 targets, 3 UAVs, and each airport has 1 ammunition. This example is used to illustrate how the algorithm can increase the number of attacks on a certain target when resources are insufficient. The initial chromosome for the mutation operation is given in Figure 9.

2	2	3	3	2	3
1	3	1	3	2	2
1	1	1	1	2	3
9	14	29	12	10	15

Figure 9. Initial chromosome for mutation operation.

Algorithm 5 Mutation operations

Input:
chromosome $= c$, **attackableUAV** $= aA$,
Output:
c, aA

$flag_1 = 0$
while $flag_1 = 0$ **do**
    Roulette chooses a target a that increases the number of attacks.
    if a satisfies constraintA. **then**
        if a satisfies constraintC. **then**
            Increase the number of attacks on a.
            $flag_1 = 1$
        else
            $flag_2 = 0$
            while $flag_2 = 0$ **do**
                Copy c and aA as c_c and aA_c.
                Delete an attack task of c_c and update aA_c.
                if a and aA_c satisfies constraintC. **then**
                    $flag_2 = 1$
            $c = c_c$,$aA = aA_c$.
            Increase the number of attacks on a.
            $flag_1 = 1$
        Update the heading angle of c.
return c, aA

First, the roulette method is used to select the target that needs to be increased the number of attacks. Since target 2 has the highest value, it has the greatest probability of being selected. In this example, the target that needs to be increased the number of attacks is also target 2. The second step of the algorithm needs to determine whether the target 2 can be attacked again. Although target 2 has been attacked once in the chromosome, there are two UAVs that can perform the attack task, so in theory, target 2 can still be attacked again. Because the current resource constraint set is empty, the target of some attack tasks in the chromosome needs to be replaced with target 2. Because in the chromosome, except for the orange column, only the green column is the attack task column. Therefore, replace the value in the gray position in the Figure 9 with 2. When the value of the gray position is changed to 2, the chromosome still satisfies the constraint, so this operation is reasonable.

After the replacement, the attack task of target 3 does not exist in the chromosome, so the classification task and verification task of target 3 are meaningless (the column marked in gray in the Figure 10). The original chromosome contains the classification task and verification task for target 2, so there is no need to add the classification task and verification task for target 2 after replacement.

2	2	3	3	2	2
1	3	1	3	2	2
1	1	1	1	2	3
9	14	29	12	10	15

Figure 10. Chromosome obtained after increasing the number of attacks on the target.

The Figure 11 is the final result of the mutation operation. The size of the chromosome has been changed through the mutation operation.

2	2	2	2
1	3	2	2
1	1	2	3
9	14	10	15

Figure 11. Chromosome obtained by mutation operation.

4.4. Program Initialization

The content of chromosome initialization coding has been described in detail above. At the same time, a very important part of program initialization is the calculation of fitness. Many issues in fitness calculation will be discussed in this part.

4.4.1. Unlock

This article uses a logic-based unlocking method to deal with the deadlock issue. This unlocking method can extract the program information contained in the chromosome, and then logically determine whether it is locked and use corresponding strategies for differentdeadlock scenarios.

First, we define some set symbols as shown in Table 2. Through the previous explanation of the structure of the chromosome, we know that a chromosome represents a task assignment plan. Each UAV executes a part of the overall plan. We call the part executed by UAV U_i as a sub-program (Subprogram) of the overall plan. Next, we define the Subprogram task execution logic and four strategies.

Table 2. Parameters in the unlocking process.

Notation	Explanation
Program	Program set
Classified	The set of targets that have performed the classification task
Classified_S	The set of unfinished missions of the attack mission is in **Classified**
Attacked_N	Number of target attacks
Can_V	A set of targets that can perform classification tasks
Verified	Status of investigation mission

Task execution logic: we traverse the program **Program** of the UAV U_i.

1. When encountering a reconnaissance task, we remove the task from the Subprogram, and then add it to **Classified** and **Classified_S**.

2. When encountering an attack task, we judge whether the target number is in **Classified_S**, if it is in it, execute the attack task, and remove the task from the Subprogram, and update **Attacked_N**, add 1 to the number of attacks on the target. At this time, if the number of attacks on the target is equal to the total number of attacks in the plan, the target number is added to **Can_V** and removed from **Classified_S**.

3. When encountering an evaluation task, we judge whether the target is in **Can_V**, if it is, we remove the task from the Subprogram, and add the target number to **Verified**.

Operation 1. Use task execution logic for each Subprogram in the overall program.

Operation 2. Traverse each Subprogram to find the reconnaissance task, and if found, exchange the execution order with the first task of the current Subprogram, and use task execution logic for the Subprogram of the exchange order.

Operation 3. Find the attack task of the first element in **Classified_S**, and if found, exchange the execution order with the first task of the current Subprogram, and use the task execution logic for the Subprogram of the exchange order and the subsequent Subprograms.

Operation 4. Traverse **Can_V**, Find the first attack mission of the target that did not complete the attack mission. The found target exchanges the execution order with the first task of the current sub-program. And use task execution logic for the sub-schemes of the exchange sequence.

After defining the above four strategies, We combine them through the process in Figure 12 to get a logic-based unlocking method.

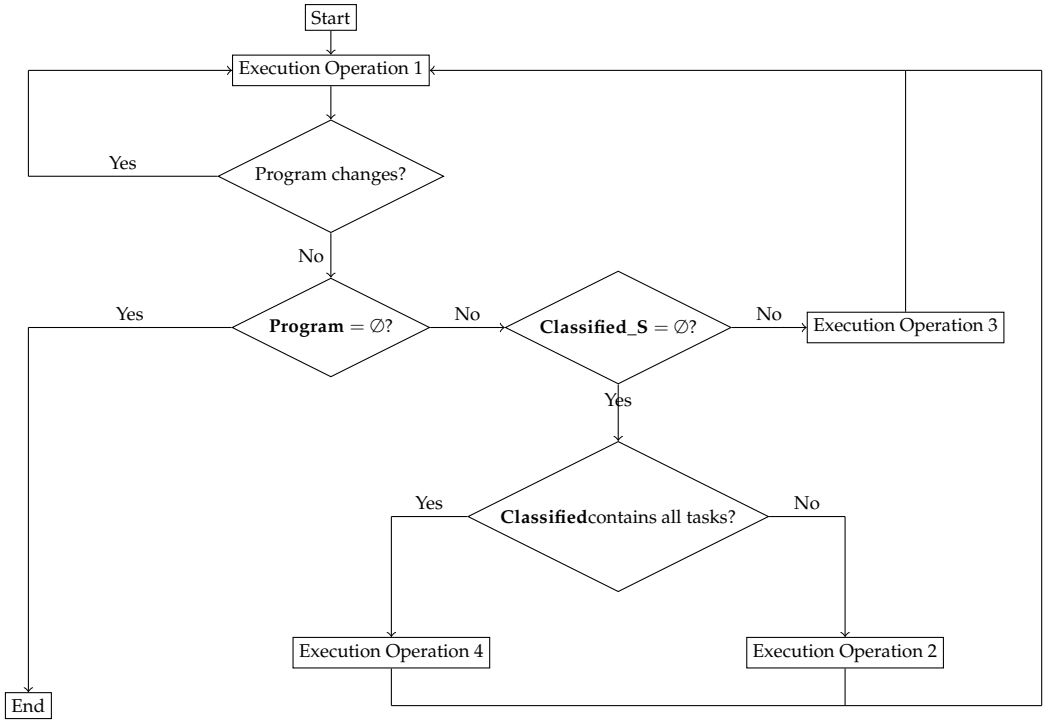

Figure 12. Logic-based unlocking process.

4.4.2. Fitness Function and Initialization

In the program initialization process, we first calculate the optimal solution in the initial population, and use J_1 of this optimal solution as the initial value of $standard_1$ and J_2 as the initial value of $standard_2$. In the optimization process, J_1 and J_2 are constantly changing and the magnitude of the change is not the same, which will actually change the fitness function. In order to ensure that the actual meaning of the fitness function does not change significantly during the iteration process, we need to iterate a certain number of times to update the standard value. Every time the specified number of iterations is reached, the standard value is replaced with J_1 and J_2 of the optimal solution of this generation.

Here is a special explanation that constantly changing the *standard* for dimensionlessness will not affect the optimization process. In other words, constantly changing the dimensionless dividend *standard* can still ensure that the algorithm is moving towards to the optimal direction. For example, we suppose $a > b > c > d > e > f > 0$. In the first dimensionless process, we choose a as the *standard*, compare $\frac{b}{a}$, $\frac{c}{a}$, we can see that $\frac{c}{a}$ is smaller. Selecting c as the *standard* for the second time, comparing $\frac{d}{c}$, $\frac{e}{c}$, $\frac{f}{c}$, obviously f is the smallest, when a is the dividend, f is also the smallest. After adding a minus sign to all values, the maximum value of this group of numbers becomes $-f$. Taking f as the first *standard*, through similar discussions, we can also know that updating *standard* will not affect the optimization. In summary, the constant replacement of *standard* can still ensure that the algorithm proceeds in the direction of optimization. Section 5.3 will discuss the impact of the replacement frequency of *standard* on the two optimization indicators.

In the actual calculation process, the value of the fitness function of Formula (4) theoretically should be [−1,1], and the roulette in the algorithm requires the value of the fitness function to be of the same sign. So we shift the fitness function value by one unit and turn it into a number between [0,2]. Finally, the fitness function will eventually become

$$F = \alpha \cdot J_1 / standard_1 + \beta \cdot J_2 / standard_2. \tag{11}$$

The overall process of initialization is shown in Figure 13.

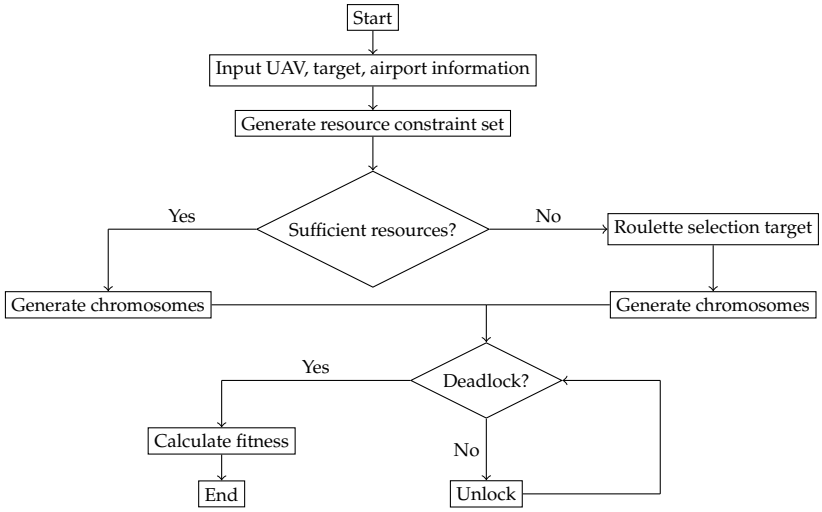

Figure 13. Initialization process.

5. Examples of Multi-Airport Task Assignment Problem

5.1. Value-Oriented Multi-Airport Task Assignment

In this section, we will give several multi-airport task assignment examples to test the effectiveness of the algorithm in this paper.

Parameter $updaterate$ is the update rate of Parameter $standard$. The meaning of the parameter $standard$ and how to take the value have been described in detail in Section 4.4.2. In Table 3, $updaterate = 5$ means that $standard$ is updated every 5 iterations.

Table 3. Parameters used in the IGA.

Parameter	Variable	Number
Population size	$population$	100
Number of iterations	$Iter_{max}$	50
Elitism Number	N_e	2
Crossover Number	N_{cr}	66
Mutation Number	N_m	32
Fitness function coefficient 1	α	0.2
Fitness function coefficient 2	β	0.8
update rate of $standard$	$updaterate$	5

Table 4 shows the location of each airport, and the ammunition quantity of each airport will be given in a specific calculation example.

For convenience, in this section we use the product of ω_{1_i} and ω_{2_j} to replace PS_{ij} in Section 3.1, $PS_{ij} = \omega_{1_i} \times \omega_{2_j} \quad 1 \leq i \leq N_U \quad 1 \leq j \leq N_T$. The variable ω_{1_i} represents the execution capability of the UAV U_i, and the closer to 1, the higher the success rate of UAV U_i executing the attack task. The variable ω_{2_j} represents the difficulty of the T_j, and the closer to 1, the easier it is to complete the T_j. Through the above explanation, we know that this substitution is reasonable.

This article considers heterogeneous UAVs, and there are three types considered in this article. C represents a combat UAV, which can perform reconnaissance and attack evaluation tasks, M represents an ammunition UAV that is specifically responsible for attack tasks, and S represents a reconnaissance UAV, which can perform reconnaissance and evaluation tasks.

Table 4. Airport information.

Airport Number	Position
1	[800,0]
2	[−2000,2000]
3	[−1000,5000]

The following calculation examples will use the data in the above Tables 3–6. If the number of devices in the calculation example is less than the number in the table, the first few data in the table will be taken. For example, 7 targets take the first 7 data of target information. The following multi-airport calculation example contains 3 airports, 5 UAVs, and 7 targets. The number of ammunition corresponding to the airport is $\{5, 2, 3\}$, and the docking status of UAVs at each airport is $A_1 = \{U_1, U_2\}$, $A_2 = \{U_3\}$, $A_3 = \{U_4, U_5\}$.

The specific task assignment scheme obtained by the algorithm is shown in Table 7.

Table 5. UAV information.

UAV Number	Type	Speed (m/s)	Max Ammunition	Min Turning Radius (m)	ω_1 (Success Rate)
1	S	250	0	80	-
2	M	210	5	60	0.9
3	C	230	2	70	0.7
4	S	250	0	80	-
5	C	200	3	90	0.9

Table 6. Target information.

Target Number	Position	Initial Task Types	Value	ω_2 (Difficulty of Execution)
1	[2200,4000]	[C,A,V]	60	0.9
2	[3200,1700]	[C,A,V]	90	0.6
3	[500,1200]	[C,A,V]	80	0.7
4	[−300,−1800]	[C,A,V]	53	0.98
5	[1100,2000]	[C,A,V]	100	0.7
6	[1100,1000]	[C,A,V]	70	0.8
7	[−500,1200]	[C,A,V]	70	0.8
8	[−2000,4800]	[C,A,V]	80	0.6
9	[−500,3200]	[C,A,V]	56	0.92

Table 7. Task assignment result.

UAV Number	UAV U_1	UAV U_2	UAV U_3	UAV U_4	UAV U_5
	$\{T_1$,V$\}$	$\{T_1$,A$\}$	$\{T_5$,C$\}$	$\{T_1$,C$\}$	$\{T_4$,A$\}$
	$\{T_3$,C$\}$	$\{T_5$,A$\}$	$\{T_5$,A$\}$	$\{T_5$,V$\}$	$\{T_3$,A$\}$
	$\{T_4$,V$\}$	$\{T_7$,A$\}$	$\{T_4$,C$\}$	$\{T_7$,C$\}$	$\{T_6$,A$\}$
	$\{T_3$,V$\}$	$\{T_3$,A$\}$	$\{T_2$,C$\}$	$\{T_7$,V$\}$	
	$\{T_6$,C$\}$	$\{T_2$,A$\}$	$\{T_2$,A$\}$		
	$\{T_6$,V$\}$		$\{T_2$,V$\}$		
The makespan: 88.62 s Total number of tasks: 24 Runtime: 31.14 s					

Figure 14 is a trajectory diagram corresponding to the above task assignment scheme.

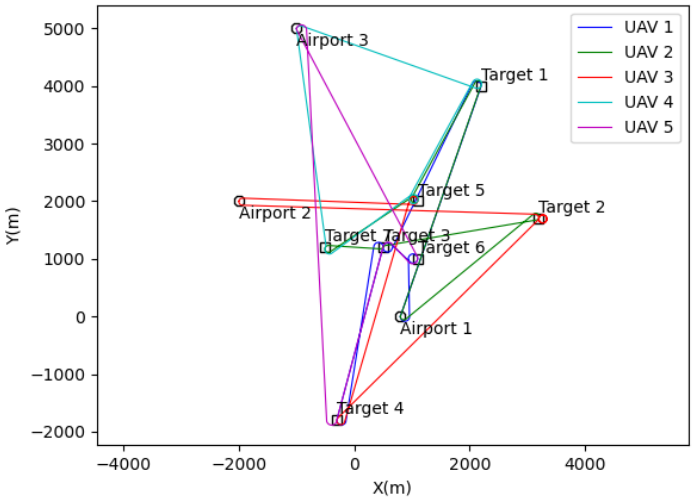

Figure 14. Task assignment and the path planning result.

The task types included in all the initial targets are $\{C, A, V\}$. In the calculation example, the number of ammunition is greater than the number of targets, so the algorithm will decide to strike certain targets multiple times based on factors such as the value and distance of the target. We can find from the task assignment plan that the plan has two attacks on targets 2, 3, and 5. It is easy to see that the value of these goals is higher than the remaining goals. If the value is set appropriately, there may be more than two attacks on a target. Under the value set of this calculation example, there will be no more than two attacks on the same target.

In addition, the algorithm in this paper can also solve the situation of insufficient ammunition. Table 8 lists the calculation results of various calculation examples.

Table 8. Algorithm Program.

	Airport-UAV	Effective Ammunition	Targets	J_1	J_2	J	Runtime (s)
Lack of resources	$A_1 - \{U_1, U_2\}$ $A_2 - \{U_3\}$	{1,1}	3	28.4092	87.8000	0.3999	4.4582
	$A_1 - \{U_1, U_2\}$ $A_2 - \{U_3, U_4\}$	{2,1}	5	27.6828	150.4000	0.3995	7.0692
	$A_1 - \{U_1, U_2\}$ $A_2 - \{U_3\}$ $A_3 - \{U_4, U_5\}$	{2,1,2}	7	47.7950	251.32	0.3999	13.1846
	$A_1 - \{U_1, U_2\}$ $A_2 - \{U_3\}$ $A_3 - \{U_4, U_5\}$	{3,2,2}	9	60.9314	332.3504	0.3995	26.2494
Sufficient resources	$A_1 - \{U_1, U_2\}$ $A_2 - \{U_3\}$	{3,2}	3	55.1529	215.0	0.3989	6.9584
	$A_1 - \{U_1, U_2\}$ $A_2 - \{U_3, U_4\}$	{5,3}	5	79.4810	356.3872	0.3992	16.3244
	$A_1 - \{U_1, U_2\}$ $A_2 - \{U_3\}$ $A_3 - \{U_4, U_5\}$	{5,2,3}	7	89.7176	455.2938	0.3988	28.7234
	$A_1 - \{U_1, U_2\}$ $A_2 - \{U_3\}$ $A_3 - \{U_4, U_5\}$	{5,3,4}	9	97.3252	460.8563	0.3979	44.0693

5.2. Logic-Based Unlocking

The logic-based unlocking method in this article is much faster than the random unlocking method. The following will use the data of some examples in Section 5.1 to test and compare. In the case of multiple airports, the parameters are still consistent with Table 3. We consider the situation of 2 airports with 3 UAVs and 3 targets, that is, the first situation in Table 8 when the ammunition is sufficient. We let the program iterate ten times to generate a new set of unlocked chromosomes with a population of 100. Calculate the fitness function of the group of chromosomes through two different unlocking methods. Since random unlocking is not stable, the time gap required each time will be large, so each different unlocking method is used to calculate the fitness function of the group of chromosomes 10 times, and the box plot drawn by the obtained data is shown in Figure 15.

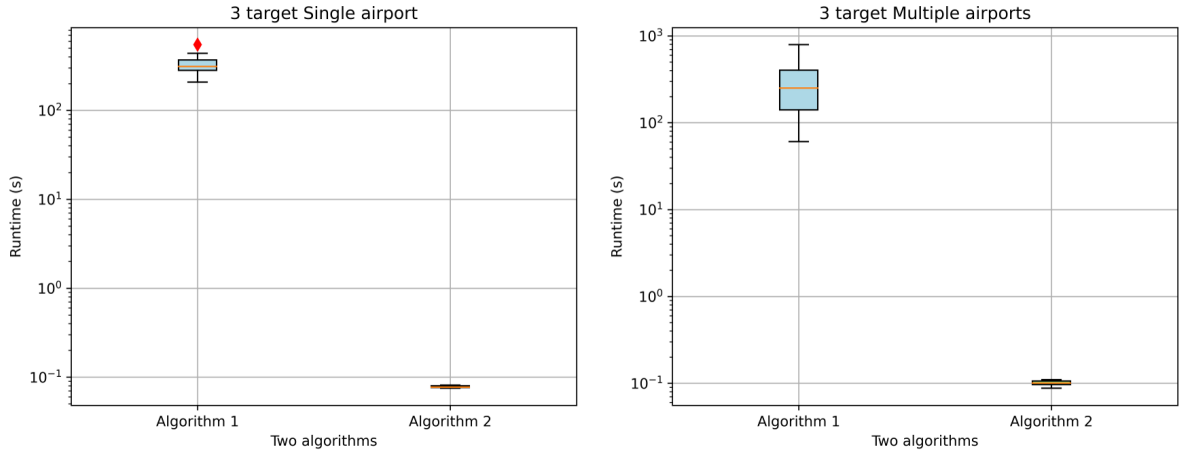

Figure 15. Time comparison of unlocking methods.

We can see from the figure that the speed of logic-based unlocking is much faster than the random exchange algorithm. In addition, when the scale becomes larger, the speed of the random algorithm becomes slower and slower. In addition, from the excellent speed of logic-based unlocking, we see the possibility of using this unlocking method to solve larger-scale problems. Figure 16 shows the changing trend of the calculation time of the algorithm based on the logic unlocking method for the task assignment of a single airport with the change of the target number. Figure 17 is a trajectory diagram of 8 UAVs in a single airport and 23 targets.

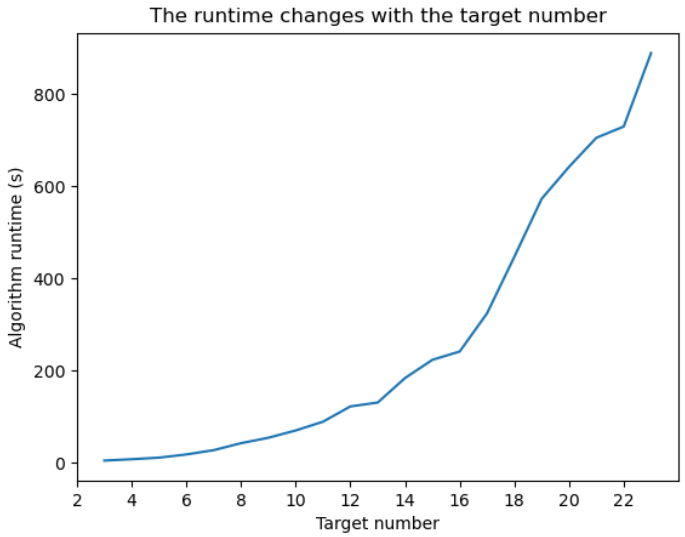

Figure 16. The runtime changes with the target number.

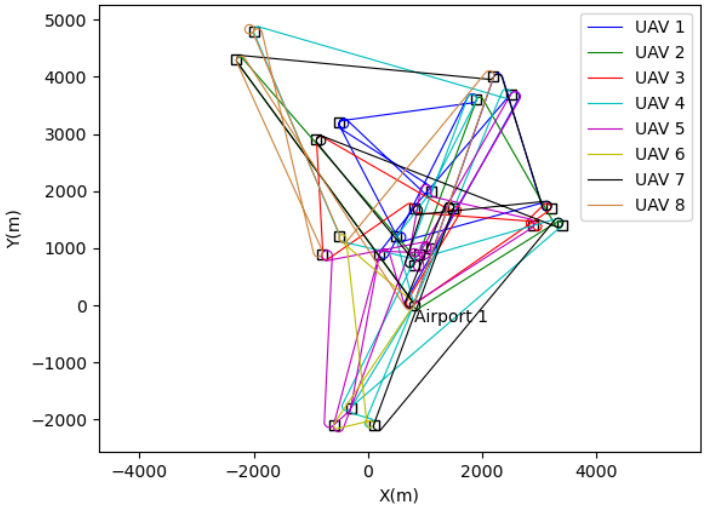

Figure 17. The assignment result for 8 UAVs and 23 targets.

5.3. Influence of Parameters on Results

For the weight coefficient, β of the fitness function, the change of the indicator is not unexpected. When β increases, the second indicator value expectation also increases, while the negative value of the first indicator's makespan decreases, as shown in Figure 18. This shows that the weight does have the effect of controlling the size of the two indicators.

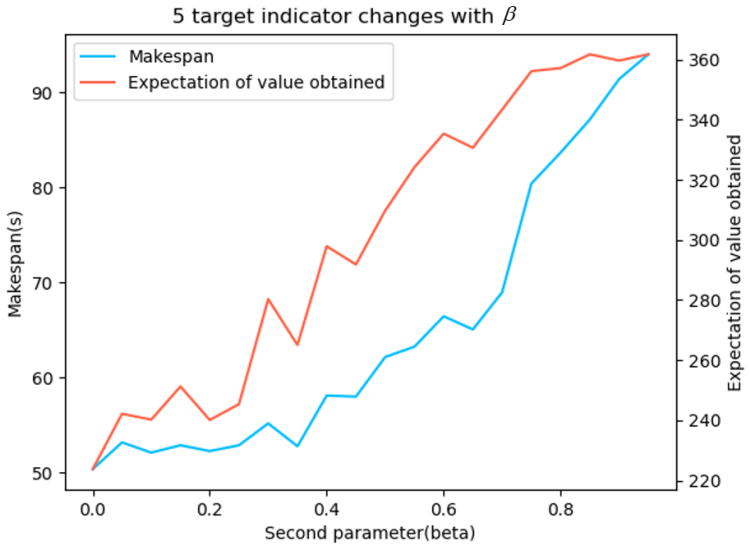

Figure 18. Indicator changes with β.

For the update rate of the standard value, Figure 19 is an image drawn by using updaterate to change from 1 to 10, calculating 10 times for each parameter and averaging the data. As can be seen from Figure 19, both indicators show an upward trend with the increase of updaterate. We know that the smaller the indicator makespan, the better the solution, and the larger the value expectation, the better the solution. When the update is frequent, there will be a smaller makespan, and in fact, the value expectation will subsequently become smaller in the process of continuously increasing the updaterate, but overall it does not change much, and the difference between the maximum and minimum values will not exceed 10. For a number of 2 orders of magnitude, this change is very small. Therefore, when calculating, we try to choose a smaller updaterate to ensure that the obtained makespan is better.

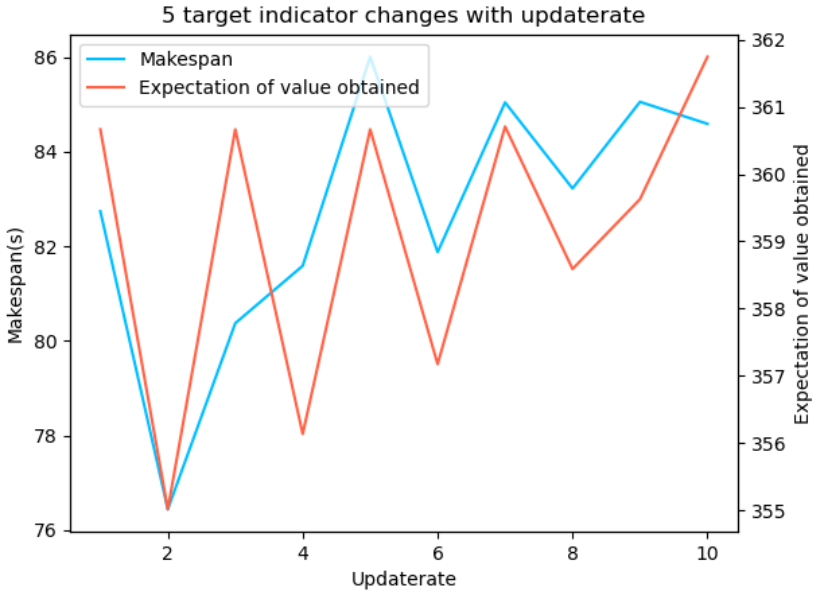

Figure 19. Indicator changes with updaterate.

6. Conclusions

In this paper, the task assignment problem involving multiple airports is studied. We construct a model for the task assignment problem, where two objectives (i.e., minimizing the makespan of the plan and maximizing the expectation of obtained value) are considered. In addition to the conventional constraints, two types of constraints (i.e., the UAVs recycling constraints and the multi-airport ammunition constraints) are proposed based on the idea of multiple airports. For solving the problem, an IGA with novel encoding format and genetic operators is developed. We creatively introduce a mutation method that changes the size of the chromosome, so that the algorithm adaptively selects the number of attacks based on the target value. In addition, a specially designed logic-based unlocking mechanism greatly improves computational efficiency. It makes the developed IGA a promising method to solve large-scale UAV task assignment problems. Our future research direction is to deal with obstacles and moving targets.

Author Contributions: Conceptualization, X.Y. and X.W.; Methodology, X.Y.; Software, X.Y.; Validation, Y.D., C.L. and S.Z.; Formal Analysis, L.W.; Investigation, X.W. and X.Y.; Resources, X.W.; Data Curation, X.Y.; Writing—Original Draft Preparation, X.Y. and X.G.; Writing—Review and Editing, L.W., X.W., Y.D., C.L. and S.Z.; Visualization, X.Y.; Supervision, L.W. and X.W.; Project Administration, X.W.; Funding Acquisition, L.W. and X.W. All authors have read and agreed to the published version of the manuscript.

Funding: The authors are grateful for the National Key Research and Development Plan (2021YFB3302501); the National Natural Science Foundation of China (12102077); the Fundamental Research Funds for the Central Universities (DUT20YG125).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Contact the first/corresponding author please.

Conflicts of Interest: The authors declare that they have no conflict of interest.

References

1. Bethke, B.; Valenti, M.; How, J.P. UAV Task Assignment: An Experimental Demonstration with Integrated Health Monitoring. *IEEE Robot. Autom. Mag.* **2008**, *15*, 39–44. [CrossRef]
2. Wang, J.; Jia, G.; Lin, J.; Hou, Z. Cooperative task allocation for heterogeneous multi-UAV using multi-objective optimization algorithm. *J. Cent. South Univ.* **2020**, *27*, 432–448. [CrossRef]

3. Gao, Y.; Zhang, Y.; Zhu, S.; Sun, Y. Multi-UAV Task Allocation Based on Improved Algorithm of Multi-objective Particle Swarm Optimization. In Proceedings of the 2018 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, Zhengzhou, China, 18–20 October 2018.
4. 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]
5. O'Rourke, K.P.; Carlton, W.B.; Bailey, T.G.; Hill, R.R. Dynamic routing of unmanned aerial vehicles using reactive tabu search. *Mil. Oper. Res.* **2001**, *6*, 5–30. [CrossRef]
6. Secreat, B.R. *Traveling Salesman Problem for Surveillance Mission Using Particle Swarm Optimization*; Wright-Patter-son, Air Force Ibstitute of Technology: Kaduna, Nigeria, 2001.
7. Zang, B.Y. *Optimize Virtual Machine on Multi-Core Processor*; Fudan University: Shanghai, China, 2014.
8. Darrah, M.A.; Niland, W.; Stolarik, B.M. Multiple UAV Dynamic Task Allocation Using Mixed Integer Linear Programming in a SEAD Mission. *Infotech* **2006**, *2005*, 7164.
9. Rasmussen, S.J.; Shima, T. Tree search algorithm for assigning cooperating UAVs to multiple tasks. *Int. Robust. Nonlinear Contr.* **2008**, *18*, 135–153. [CrossRef]
10. Kennedy, J.; Eberhart, R. Particle swarm optimization. *Neural Netw.* **2002**, *4*, 1942–1948.
11. Souza, G.d.M.; Toledo, C.F.M. Genetic Algorithm Applied in UAV's Path Planning. In Proceedings of the 2020 IEEE Congress on Evolutionary Computation, Glasgow, UK, 19–24 July 2020; pp. 1–8.
12. Wang, Z.; Liu, L.; Long, T.; Yun, W. Multi-uav reconnaissance task allocation for heterogeneous targets using an opposition-based genetic algorithm with double-chromosome encoding. *Chin. J. Aeronaut.* **2018**, *31*, 339–350. [CrossRef]
13. Zaza, T.; Richards, A. Ant Colony Optimization for routing and tasking problems for teams of UAVs. In Proceedings of the 2014 UKACC International Conference on Control, Loughborough, UK, 9–11 July 2014; pp. 652–655.
14. Zhen, Z.; Zhu, P.; Xue, Y.; Ji, Y. Distributed intelligent self-organized mission planning of multi-uav for dynamic targets cooperative search-attack. *Chin. J. Aeronaut.* **2019**, *32*, 2706–2716. [CrossRef]
15. Battiti, R.; Tecchiolli, G. The Reactive Tabu Search. *Informs J. Comput.* **1994**, *6*, 126–140. [CrossRef]
16. Duan, H.; Luo, Q.; Shi, Y.; Ma, G. Hybrid Particle Swarm Optimization and Genetic Algorithm for Multi-UAV Formation Reconfiguration. *IEEE Comput. Intell. Mag.* **2013**, *8*, 16–27. [CrossRef]
17. Arantes, M.d.S.; Arantes, J.d.S.; Toledo, C.F.M.; Williams, B.C. A Hybrid Multi-Population Genetic Algorithm for UAV Path Planning. In Proceedings of the Genetic and Evolutionary Computation Conference, Denver, CO, USA, 20–24 July 2016; pp. 853–860.
18. Smith, R.G. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Trans. Comput.* **1980**, *29*, 1104–1113. [CrossRef]
19. Fu, X.; Feng, P.; Gao, X. Swarm uavs task and resource dynamic assignment algorithm based on task sequence mechanism. *IEEE Access* **2019**, *7*, 41090–41100. [CrossRef]
20. Jia, Y. Research on UAV Task Assignment Method Based on Parental Genetic Algorithm. In *International Conference on Swarm Intelligence*; Springer: Cham, Switzerland, 2019; pp. 439–446.
21. Ye, F.; Chen, J.; Tian, Y.; Jiang, T. Cooperative Task Assignment of a Heterogeneous Multi-UAV System Using an Adaptive Genetic Algorithm. *Electronics* **2019**, *9*, 687. [CrossRef]
22. Yang, W.Z.; Xin, Y. Multi-UAV Task Assignment Based on Quantum Genetic Algorithm. *J. Phys. Conf. Ser.* **2021**, *1824*, 12010. [CrossRef]
23. 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]
24. Bertuccelli, L.; Choi, H.-L.; Cho, P.; How, J. Real-Time Multi-UAV Task Assignment in Dynamic and Uncertain Environments. In Proceedings of the AIAA Guidance, Navigation, and Control Conference, Chicago, IL, USA, 10–13 August 2009.
25. 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]
26. Hu, X.; Cheng, J.; Luo, H. Task assignment for multi-uav under severe uncertainty by using stochastic multicriteria acceptability analysis. *Math. Probl. Eng.* **2015**, *14*, 249825.1–249825.10. [CrossRef]
27. Wang, Z.; Zheng, M.; Guo, J.; Huang, H. Uncertain UAV ISR mission planning problem with multiple correlated objectives. *J. Intell. Fuzzy Syst.* **2017**, *32*, 321–335. [CrossRef]
28. Jia, X.; Wu, S.; Sun, J. UAV formation collaborative search and dynamic task allocation method under uncertain environment. In Proceedings of the 33rd Chinese Control Conference, Nanjing, China, 28–30 July 2014; pp. 947–953.
29. Zhen, Z.; Chen, Y.; Wen, L.; Han, B. An intelligent cooperative mission planning scheme of UAV swarm in uncertain dynamic environment. *Aerosp. Sci. Technol.* **2020**, *100*, 105826. [CrossRef]
30. Chen, Y.; Yang, D.; Yu, J. Multi-UAV Task Assignment With Parameter and Time-Sensitive Uncertainties Using Modified Two-Part Wolf Pack Search Algorithm. *IEEE Trans. Aerosp. Electron. Syst.* **2018**, *54*, 2853–2872. [CrossRef]
31. Deng, Q.; Yu, J.; Mei, Y. Deadlock-free consecutive task assignment of multiple heterogeneous unmanned aerial vehicles. *J. Aircr.* **2014**, *51*, 596–605. [CrossRef]
32. Edison, E.; Shima, T. Integrated task assignment and path optimization for cooperating uninhabited aerial vehicles using genetic algorithms. *Comput. Oper. Res.* **2011**, *38*, 340–356. [CrossRef]