



Article

Multi-Conflict-Based Optimal Algorithm for Multi-UAV Cooperative Path Planning

Xiaoxiong Liu ^{1,*} , Yuzhan Su ¹, Yan Wu ² and Yicong Guo ¹ ¹ School of Automation, Northwestern Polytechnical University, Xi'an 710129, China² School of Mathematics and Statistics, Xidian University, Xi'an 710071, China

* Correspondence: liuxiaoxiong@nwpu.edu.cn

Abstract: Multi-UAV cooperative path planning can improve the efficiency of task completion. To deal with the space and time conflicts of multi-UAVs in complex environments, a multi-collision-based multi-UAV cooperative path planning algorithm, multi-conflict-based search (MCBS), is proposed. First, the flight and cooperative constraints of UAV are analyzed, and a three-dimensional environment model is established that incorporates geographical information. Then, hierarchical optimization is used to design collaborative algorithms. In the low-level path design, UAV flight constraints are combined with a sparse A* algorithm, and by improving the cost function, the search space is reduced, and the search time is shortened. In high-level cooperation, the priorities of different conflicts are set, heuristic information is introduced to guide the constraint tree to grow in the direction of satisfying the constraints, and the optimal path set is searched by the best priority search algorithm to reduce the convergence time. Finally, the planning results of the proposed algorithm, the traditional CBS algorithm, and the sparse A* algorithm for different UAV tasks are compared, and the influence of the optimization parameters on the calculation results is discussed. The simulation results show that the proposed algorithm can solve cooperative conflict between UAVs, improve the efficiency of path searches, and quickly find the optimal safe cooperative path that satisfies flight and cooperative constraints.

Keywords: multi-UAV cooperative; path planning; conflict-based search; sparse A*



Citation: Liu, X.; Su, Y.; Wu, Y.; Guo, Y. Multi-Conflict-Based Optimal Algorithm for Multi-UAV Cooperative Path Planning. *Drones* **2023**, *7*, 217. <https://doi.org/10.3390/drones7030217>

Academic Editor: Oleg Yakimenko

Received: 20 January 2023

Revised: 17 March 2023

Accepted: 20 March 2023

Published: 21 March 2023



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

1. Introduction

In recent years, as the flight environment and mission of UAVs have become more complex, a single UAV can no longer meet the mission requirements. Multi-UAV cooperative systems have attracted much attention from researchers due to their advantages such as scalability and robustness [1]. Multi-purpose UAVs are widely used in both military and civilian applications such as search and rescue, detection, and surprise defense [2–4]. Avoiding various threats in the environment is a key problem in achieving cooperative operations with multi-UAVs [5,6], which is usually defined as finding the optimal set of paths that satisfy various constraints of multi-UAVs [7,8].

Many multi-UAV path planning algorithms have been recently proposed by research scholars, and can be divided into two categories: reactive collision avoidance methods and active cooperative path generation methods. In reactive collision avoidance methods [9–11], any UAV that detects a risk of collision with another UAV performs a local cooperative replanning algorithm to avoid collision. Many algorithms based on reactive collision avoidance methods have been proposed, such as optimal reciprocal collision avoidance (ORCA) [12] and distributed reactive collision avoidance (DRCA) [13,14]. Reactive collision avoidance methods have been widely used in practice by virtue of their efficiency [15]. However, such methods only consider local cooperative collision avoidance, and thus tend to fall into a local optimum. In addition, such algorithms are difficult to deal with path planning problems that require satisfying both time and space cooperative constraints on UAVs. Among the active cooperative path generation

methods, decoupling methods and coupling methods are mainly included. The decoupling method uses a hierarchical architecture to deal with the multi-UAV path planning problem, i.e., after coordinating the planning order of all UAVs through the path coordination method, the cooperative paths are generated for each UAV in turn independently. The multi-UAV path planning algorithms based on decoupling methods are the hierarchical cooperative A* algorithm (HCA*) [16], the prioritized planning algorithm (PP) [17], and the ADPP [18]. The decoupling algorithm can transform the multi-UAV cooperative path planning problem into a single-UAV path planning problem so it can obtain the cooperative path quickly and effectively. However, such methods usually can only obtain feasible paths and have some limitations in finding optimal solutions. On the other hand, coupled methods usually design only one cooperative path planner for multi-UAVs and are able to plan the set of safe paths, meeting the constraints for all UAVs simultaneously. Some coupling-based algorithms usually convert the cooperative path planning problem into problems that have been well studied to obtain cooperative paths, such as linear programming (LP) [19] and the constraint satisfaction problem (CSP) [20]. In addition, some search-based coupling methods such as CBS have been proposed. Coupled methods are able to obtain find the globally optimal set of collaborative paths, and they have gained extensive research this year.

This paper is devoted to the research of the cooperative path planning problem of multiple fixed-wing UAVs under the scenario of a surprise defense. The multi-UAV cooperative path planning problem is essentially a multi-agent problem. Sharon [21] proposed the conflict-based search algorithm, a recent achievement in multi-agent path planning that has been applied in warehouse robot systems [22]. Conflict-based search (CBS) uses a low-level heuristic search algorithm to find the optimal path for the agent to satisfy the constraints of the high level where a binary tree is constructed to set constraints according to the conflict between the agent paths. The best-first search algorithm is adopted to ensure optimal planning results. CBS converts the multi-agent path planning problem into multiple single-agent problems and searches for the optimal solution. Experimental results showed that CBS reduced the search time by up to an order of magnitude. Therefore, it has great potential for solving the multi-UAV path planning problem.

Based on the CBS algorithm, many improved methods have been proposed. Fatih Semiz [23] proposed an incremental algorithm by replacing the low-level A* algorithm of CBS with D*-lite for the multi-agent path planning problem in a dynamic environment. Bare [24] proposed an enhanced CBS (ECBS) algorithm that replaces the best-first search algorithm in the high and low levels of CBS with a focused search. Li Jiaoyang [25] replaced the focused search with an explicit estimation search based on ECBS and proposed a new bounded suboptimal variant of CBS called explicit estimation CBS (EECBS).

It is not difficult to see that researchers' improvements for CBS focused on improving algorithm solution efficiency. CBS and its improvement methods are not suitable for multi-UAV cooperative path planning because they do not take into account how a complex 3D environment affects UAVs, their flight constraints or the cooperative constraints between UAVs. In particular, how to use CBS to solve the multi-agent path planning problem with time constraints is a current research blind spot. Therefore, when solving it in a complex 3D environment, the flight, space and time constraints of UAVs and environmental threats should be fully considered [26]. For example, in the sparse A* algorithm, Zhe Zhang [27] introduced the flight constraints of the UAV into the sparse A* algorithm to realize 3D UAV path planning. In some swarm agent algorithms, Cheng Xu [28] comprehensively considered the time and space constraints between UAVs and realized multi-UAV cooperative path planning in four-dimensional space.

Therefore, we propose a multi-conflict-based optimal algorithm for the multi-UAV cooperative path problem in complex environments by combining the flight constraints of UAVs and the cooperative constraints between them. At the low level of the algorithm, we designed a sparse A* algorithm that satisfied the flight constraints of UAVs in a 3D environment; at the high level of the algorithm, we defined the cooperative conflicts between multi-UAVs, set the priorities of different conflicts, changed the growth mode of

the constraint tree and designed a heuristic function to guide the algorithm search. The proposed algorithm has been proven effective in simulation experiments.

The main contributions of our work are as follows:

- (1) Considering the constraint information of UAVs, the complex environment of multiple UAVs was modeled.
- (2) A sparse A * algorithm was designed to meet the flight constraints of UAVs, which reduces the search space and shortens the search time.
- (3) The collaborative conflict between multiple UAVs was defined; the priority of different collaborative conflicts is set; and the heuristic information was designed to guide the constraint tree to grow in the direction of conflict resolution to improve the algorithm's convergence time.

The arrangement of this article is as follows: in Section 2, we establish a constrained optimization model for multi-UAV cooperative path planning, analyze the model's flight and cooperative constraints, and then model the battlefield environment; in Section 3, we present in detail the multi-UAV cooperative path planning algorithm based on CBS; in Section 4, we evaluate the performance of the algorithm through simulation experiments; the conclusion is given in Section 5.

2. Problem Description

The problem studied in this paper is the cooperative path planning problem when a multi-UAV performs tasks in a complex environment. The multi-UAV cooperative path planning algorithm not only needs to find the path with the smallest cost for each aircraft, but also needs to satisfy certain constraints, such as those imposed by the physical characteristics of UAVs, environment threats and obstacles, and the cooperative constraints between the UAVs. Therefore, the multi-UAV cooperative path planning problem can be regarded as a constrained optimization problem that can be described as the following formula:

$$\min J(P/E) \text{ s.t. } \begin{cases} g(P) \leq 0 \\ h(P) = 0 \end{cases} \quad (1)$$

where E is the environment for path planning; $g(P)$ and $h(P)$ are inequality constraints that comprehensively consider UAV flight and cooperative constraints; P is a set of paths that satisfy the constraints, which can be expressed as $P = \{p_1, p_2, \dots, p_N\}$.

2.1. Path Cost Analysis

The total path cost J is the sum of all UAV path costs:

$$J(P/E) = \sum_{i=1}^N J_i(p_i/E) \quad (2)$$

where p_i is the path of the i th UAV; N is the number of UAVs; $J(P/E)$ is the total path cost of the path set P in the current environment E . The path cost J_i of the i th UAV is composed of two parts: the path length cost $J_{length}(p_i)$ and the path threat cost $J_{threat}(p_i)$:

$$J_i(p_i/E) = J_{length}(p_i) + J_{threat}(p_i/E) \quad (3)$$

2.1.1. Path Length Cost Function

When planning the path of the i th UAV, the path p_i is the set of waypoints where the i th UAV is located at different times. When evaluating the length cost of the UAV path p_i , the path length cost function of the n th waypoint v_i^n is as follows:

$$j_l(v_i^n) = \sqrt{(x_n - x_{n-1})^2 + (y_n - y_{n-1})^2 + (z_n - z_{n-1})^2} \quad (4)$$

where (x_n, y_n, z_n) is the 3D coordinates of the i th waypoint in the path.

The path length cost is the sum of all waypoint length costs except for the starting point. The calculation formula is as follows:

$$J_{length}(p_i) = \sum_{n=2}^{l_i} j_l(v_i^n) \quad (5)$$

where l_i is the number of waypoints in the path.

2.1.2. Path Threat Cost Function

The path threat cost mainly comes from five aspects: terrain, air defense radar, surface-to-air missile, anti-aircraft artillery and no-fly zones. The path threat cost is the sum of the threat costs of all waypoints in the path. The path threat cost function is as follows:

$$J_{threat}(p_i/E) = \sum_{n=1}^{l_i} j_t(v_i^n) \quad (6)$$

where the threat cost j_t of the n th waypoint v_i^n is the weighted sum of five threat costs, as shown in the following formula:

$$j_t = k_g \cdot U_g + k_r \cdot U_r + k_m \cdot U_m + k_a \cdot U_a + k_n \cdot U_n \quad (7)$$

where k_g , k_r , k_m , k_a , and k_n are weighting factors.

The path threat cost is affected by the environment E , so it is crucial to model environmental threats accurately.

(1) Terrain threat model and cost analysis

To make the simulation environment based on path planning closer to the real environment, this paper selected a 500×500 km digital elevation model as the environmental model for path planning, as shown in Figure 1. The digital elevation model is located roughly between 107° – 119° E and 34° – 35° N. The size of the digital elevation model is 1000×1000 and 10^6 data points exist.

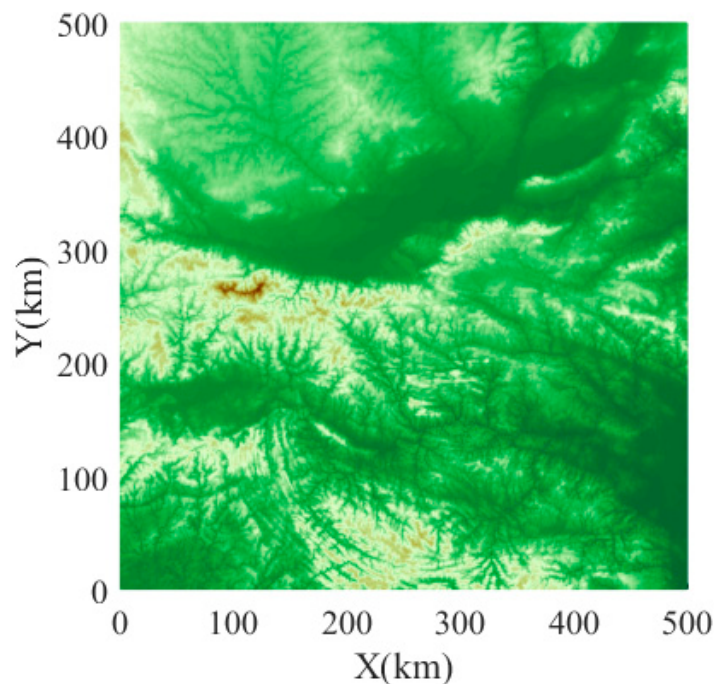


Figure 1. Environmental model.

For waypoint v_i^n , the terrain threat cost is calculated as follows:

$$U_g(v_i^n) = \begin{cases} q_g & \text{if } (z_n - h_g) < H_{\min} \\ 0 & \text{if } (z_n - h_g) > H_{\min} \end{cases} \quad (8)$$

where z_n is the height of the waypoint v_i^n ; h_g is the ground height; and H_{\min} is the minimum height above ground.

(2) Air defense radar model and analysis

When modeling the radar, we took into account the radar blind spot created by terrain obscuring. The calculation steps of radar terrain masking a blind area are as follows:

(a) Calculate the radar ray equation

Assuming that the height of the radar station is h_r , the schematic diagram of the radar detecting the terrain obscuring blind area in a certain azimuth is as shown in Figure 2.

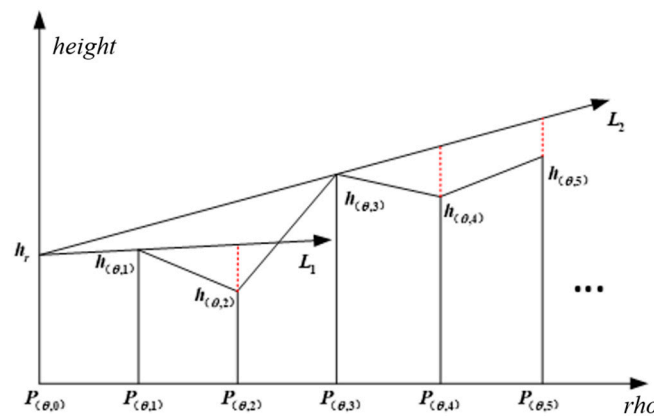


Figure 2. Schematic diagram of the blind area of the radar in a certain azimuth.

Based on known polar coordinates $P_{(\theta,k)}$ of different radar detection azimuths and digital elevation models $h_{(\theta,k)}$, the radar detection ray L_1 equation is:

$$L_1(P_{(\theta,k)}) = h_r + \frac{h_{(\theta,1)} - h_r}{P_{(\theta,1)} - P_{(\theta,0)}}(P_{(\theta,k)} - P_{(\theta,0)}) \quad (9)$$

(b) Calculate the height of the radar blind spot

Expand outward in turn from $P_{(\theta,0)}$. If the terrain height $h_{(\theta,k)}$ is below the radar detection ray, there is a blind spot. If the terrain height $h_{(\theta,k)}$ is above the radar detection ray, update ray, and continue expanding outward to calculate the blind spot height. If the terrain height at $P_{(\theta,2)}$ is below the radar detection ray and there is a blind spot, its height at this point can be represented by the dotted line in Figure 2, and the height value is $L_1(P_{(\theta,2)}) - h_{(\theta,2)}$. The threat range of the radar in the final environment is shown in Figure 3, and the radar detection area is above the curved surface.

When the UAV flies over the radar detection area above the radar detection blind spot surface, it will face the danger of being detected by the enemy. We represent the radar threat cost of waypoint v_i^n using the radar detection probability:

$$U_r(v_i^n) = \begin{cases} \left(1 + \frac{1}{1 + \frac{2 \times d^4}{R_r^4}}\right)^{-2}, & v_i^n \subseteq P_r \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

where P_r is the point set of the radar detection coverage area; R_r is the radar detection radius; and d is the horizontal distance between the waypoint v_i^n and the radar center point.

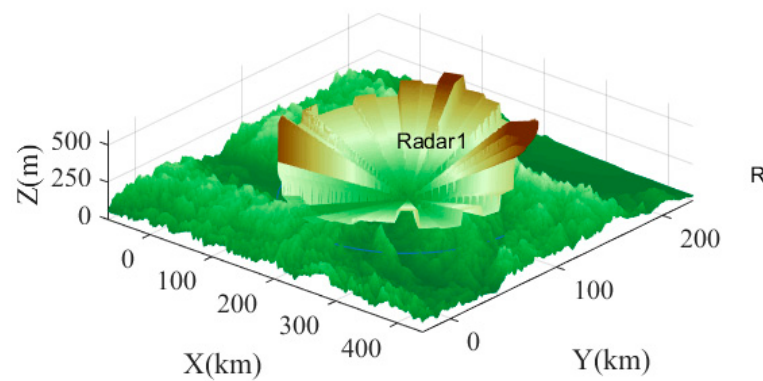


Figure 3. Radar detection area.

(3) Surface-to-air missile threat model and analysis

For general surface-to-air missiles, the interception coverage area can be approximated by a hemisphere. The path planning space we studied was mainly aimed at an airspace of 6000 m and below for the flight of midair UAVs, so the surface-to-air missile interception coverage area at this height is represented as the remaining hemisphere with the top part removed. The threat range of the surface-to-air missile in the environment is shown in Figure 4.

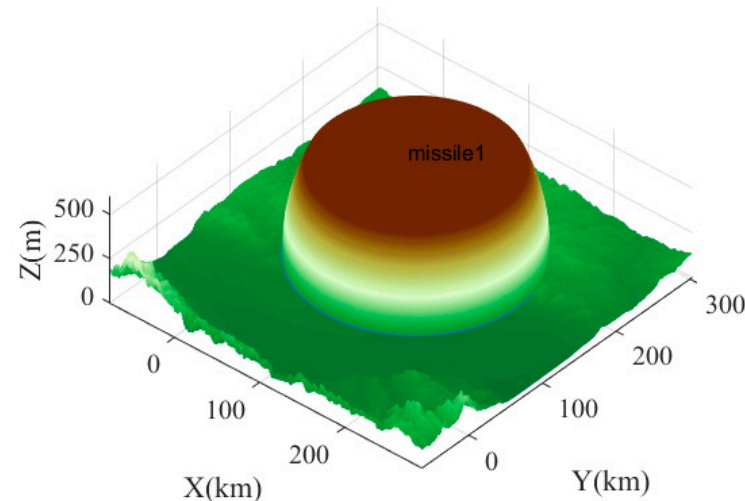


Figure 4. The threat range of surface-to-air missile.

The threat cost of the surface-to-air missile to the target within the attack range can be expressed by the following formula:

$$U_m(v_i^n) = \begin{cases} \frac{(r_m - R_{mmin}) \times (R_{mmax} - r_m)}{(R_{mmin} + R_{mmax})^2 / 4} & , R_{min} \leq r_m \leq R_{max} \text{ and } v_i^n \subseteq P_m \\ 0 & , otherwise \end{cases} \quad (11)$$

where r_m is the distance between the missile launcher and the attack target; R_{mmax} is the maximum attack distance of the missile; R_{mmin} is the minimum attack distance of the missile; and P_m is the point set of the surface-to-air missile interception coverage area.

(4) Anti-aircraft artillery threat model and analysis

Similar to the surface-to-air missile, the threat space of anti-aircraft artillery can also be represented as the remaining hemisphere with the upper part removed. The threat range of anti-aircraft artillery in the environment is shown in Figure 5.

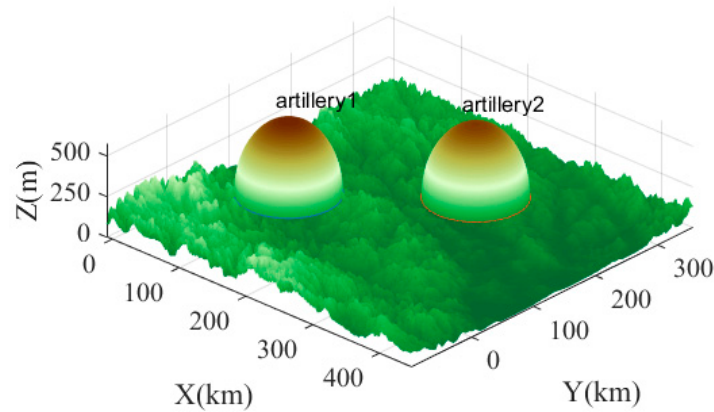


Figure 5. The threat range of surface-to-air missile.

Within the scope of the anti-aircraft artillery, the closer the target is to the firing point of the artillery, the greater the probability of its being shot down; the farther the target is from the firing point, the smaller the probability, and the probability is zero when it exceeds the artillery range. The air defense artillery threat cost of the path waypoint v_i^n at a certain point can be represented by U_a , which can be determined using the following formula:

$$U_a(v_i^n) = \begin{cases} \exp\left(-\frac{(x_n-x_0)^2+(y_n-y_0)^2+(z_n-z_0)^2}{R_a^2/9}\right) & , v_i^n \subseteq P_a \\ 0 & , \text{otherwise} \end{cases} \quad (12)$$

where (x_0, y_0, z_0) is the position coordinate of the anti-aircraft artillery; (x_n, y_n, z_n) is the waypoint coordinate; R_a is the maximum firing radius of the anti-aircraft artillery; and P_a is the effective point set of the anti-aircraft artillery.

(5) No-fly zone threat model and analysis

Besides avoiding entering enemy threat areas, UAVs also need to avoid no-fly areas set by meteorological, political or other criteria. Generally speaking, their design is mostly quadrilateral, so we use quadrangular prisms to represent the no-fly zone for UAVs. The threat range is shown in Figure 6.

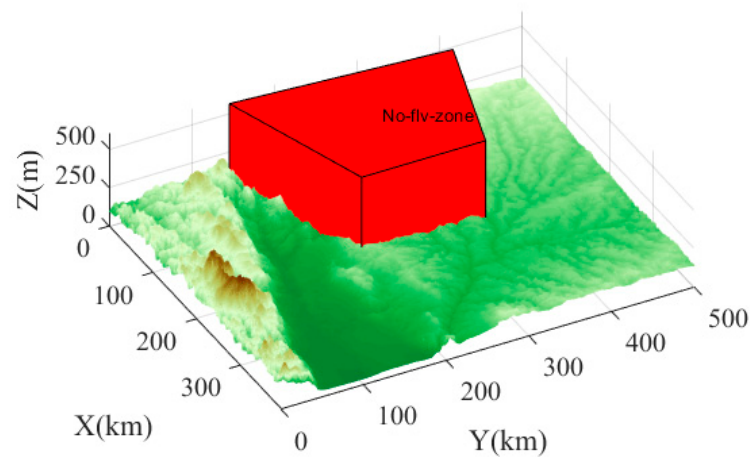


Figure 6. The threat range of the no-fly zone.

For waypoint v_i^n , define P_n as the point set in the no-fly zone space, and the calculation method of the no-fly zone threat cost is as follows:

$$U_n(v_i^n) = \begin{cases} q_n, v_i^n \subset P_n \\ 0, v_i^n \not\subset P_n \end{cases} \quad (13)$$

2.2. Constraint Analysis

The different sources of constraints that UAVs must satisfy in flight can be divided into two categories: UAV flight constraints derived from its physical characteristics, and the cooperative constraints between UAVs.

2.2.1. UAV Flight Constraint Analysis

For the multi-UAV path planning problem studied in this paper, we select and derive a set of flight constraints suitable for this paper based on the representative constraints in the literature [29,30] to consider the key performance constraints and complex environmental requirements in UAV path planning. We mainly consider the following four UAV flight constraints:

(1) Minimum turning radius

Due to maneuvering constraints, the UAV turning radius needs to be larger than the minimum. Therefore, the planned path needs to impose a minimum turning radius constraint to be flyable, and this can be expressed as:

$$r_i > r_{\min} (i = 1, 2, \dots, n) \quad (14)$$

where r_i is the turning radius of the planned path in the first path segment, and r_{\min} is the minimum turning radius of the UAV.

(2) Minimum path segment length

The minimum path segment length is the minimum distance that the UAV must fly before changing its attitude. The planned path unit step size we set needed to meet the minimum path segment length constraint, which can be expressed as

$$dS > dS_{\min} \quad (15)$$

where dS is the path planning unit step size, and dS_{\min} is the minimum path length.

(3) Maximum path slope angle

Since the UAV is constrained by engine performance and flight safety, the vertical ascent and dive angles of the path are also limited. Therefore, within a unit path step, the planned flight path of the UAV must meet the maximum path slope angle constraint. Assuming that the horizontal projection of the i th path is expressed as $a_i = (x_i - x_{i-1}, y_i - y_{i-1})^T$, the difference in the vertical direction is $b_i = |z_i - z_{i-1}|$, and the maximum path slope angle is assumed to be θ_{\max} , then the maximum path slope angle constraint can be expressed as:

$$\frac{b_i}{a_i} \leq \tan \theta_{\max} (i = 1, 2, \dots, n) \quad (16)$$

(4) Minimum ground height

To avoid collision with terrain threats such as mountains, it is necessary to set the minimum height above the ground, so that the flying height of the UAV is greater than the minimum height above the ground. The minimum ground height constraint can be expressed as:

$$z_i \geq H_{\min} (i = 1, 2, \dots, n) \quad (17)$$

where z_i is the height value of the i th path planning point, and H_{\min} is the minimum height above the ground.

2.2.2. Constraint Analysis

In addition to satisfying the flight constraints of a single aircraft, multi-UAV path planning also needs to meet the cooperative constraints that require UAVs to cooperate to complete the task within a specified time without any space conflicts. According to the task requirements, UAVs need to keep consistent or fly in a certain order. Therefore,

the cooperative constraints between UAVs are divided into space and time cooperative constraints.

(1) Time cooperative constraint analysis

A time cooperative constraint requires that each UAV flies from different starting points according to the planned cooperative path and reaches the target point according to a certain time sequence or the same time interval. Assuming that the planned track length of the i th UAV is L_i and the speed range is $[V_{min}, V_{max}]$, the time range T_i for the UAV to reach the target point is $[L_i/V_{max}, L_i/V_{min}]$. To enable all N UAVs to reach the target point at the same time through speed allocation, it is necessary that the intersection of the expected arrival time windows T_i of UAVs not be empty, that is, the cooperative arrival time window of the UAV group should be $T = T_1 \cap T_2 \cap \dots \cap T_N$. Although the time cooperation between UAVs can be achieved by adjusting speed, the adjustable range is limited. Therefore, as shown in Figure 7, we mainly considered adjusting the arrival time by adjusting the UAV path length.

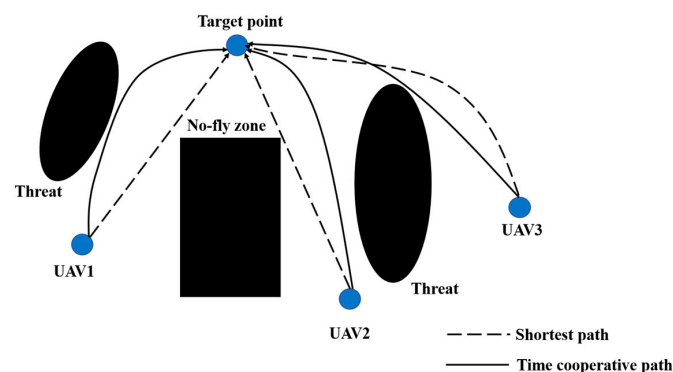


Figure 7. Time cooperative path.

(2) Space cooperative constraint analysis

During multi-UAV cooperative path planning, in addition to the time cooperation of UAVs, it is also necessary to consider the space cooperation of different UAVs. As shown in Figure 8, the space cooperative constraint is that the distance between any two UAVs in flight at the same time is always greater than the safety distance D_{safe} . Suppose the distance between the i th UAV and j th UAV at the time t is $D_{ij}(t)$. If all N UAVs are required to maintain a safe distance, the following formula must be satisfied for any time t :

$$z_i \geq H_{\min}(i = 1, 2, \dots, n) \quad (18)$$

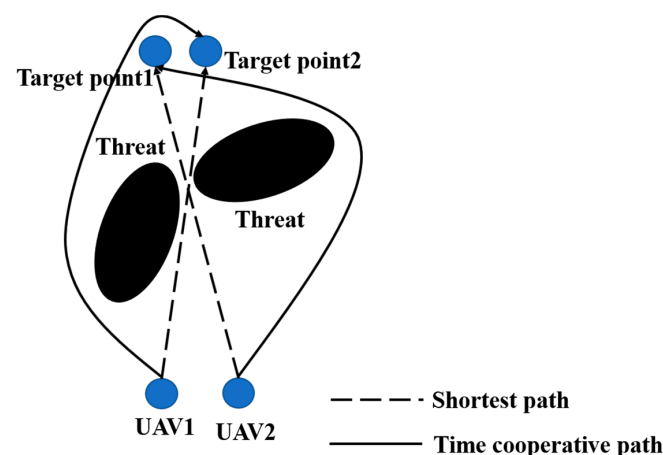


Figure 8. Space cooperative path.

3. Multi-Conflict-Based Optimal Algorithm

Compared with single-UAV path planning, multi-UAV path planning has much greater complexity because it needs to consider not only the optimal path of the single-UAV but also the multi-UAV cooperative path problem. We noticed that the CBS algorithm solved the space cooperative problem of planar multi-agent path planning excellently, so we tried to solve the multi-UAV path planning problem using the CBS algorithm; however, new problems were encountered. For example, the low-level A* algorithm had difficulty carrying out 3D space path planning; the UAV had many constraints; and the path conflict between UAVs was lacking. The specific definition and coordination problems included both space and time conflicts. In response to these problems, we made a new algorithm called multi-conflict-based search (MCBS).

3.1. Conflict Based Search

Conflict-based search (CBS) is a multi-agent path planning algorithm chiefly designed to create a constraint set for each agent according to the conflicts between the paths of agents and break down the multi-agent path planning problem into a large number of constrained combinations of single-agent path planning problems. The CBS algorithm is divided into two levels. The high level is responsible for discovering the conflicts between paths of agents and adding constraints; the low level is responsible for planning paths that meet the constraints of the single agent.

The pseudo-code for CBS is shown in Algorithm 1. The high level uses the best-first search algorithm. The algorithm will generate a root tree node *Root* with empty constraints set at first. It consists of four parts: the set of paths, the solution *Root.solution*, the cost of the node *Root.cost*, and the constraints set *Root.constraints*. The search tree then begins to expand. All the tree nodes that have not been selected yet are put into the $OPEN_{cbs}$ table and then the node with the lowest growth cost is selected. Each tree node contains a set of constraints to avoid path conflicts and a set of paths that meet the constraints. The cost of the node is the sum of the lengths of the paths. When expanding the constraint tree node, the algorithm will first check the conflict between paths, that is, whether multiple agents are occupying the same vertex or the same edge at the same time. If there is no conflict, the path is the target path, and the algorithm terminates. Otherwise, if there is a conflict between the i th agent a_i and the j th agent a_j at vertex v , CBS will generate two child nodes, with the current constraint tree node as the parent. Each child node inherits a path set and constraint information from its parent, which resolves the conflict by adding new constraints c_s to the child nodes and then replanning the path of the agent through the low-level A* algorithm. By the growth of the constraint tree, the CBS algorithm analyzes the two solutions to a conflict to ensure the completeness of the algorithm and the optimality of the algorithm through the best-first search at both high and low levels.

Algorithm 1 Pseudocode of the CBS algorithm.

```

Input: MAPF instance
Root.constraints  $\leftarrow \emptyset$ ;
Root.solution  $\leftarrow$  Find path for each agent through A*;
Root.cost  $\leftarrow \text{cost}(\text{Root.solution})$ ;
Insert Root into OPENcbs;
While OPENcbs  $\neq \emptyset$ 
    Find the tree node N with the minimal cost in OPENcbs;
    if N.solution has conflicts,
        find the conflict  $(a_i, a_j, v, t)$  which occurs first in N.solution;
        Remove N from OPENcbs;
        for  $a_x$  in  $[a_i, a_j]$ :
            Create a new constraint set  $c_s = (a_x, v_x, t)$ ;
             $P \leftarrow \text{CreatNode}()$ ;
            P.constraints  $\leftarrow$  N.constraints +  $c_s$ ;
            P.solution  $\leftarrow$  N.solution;
            P.solution( $a_x$ )  $\leftarrow$  Find path for agent  $a_x$  through A*;
            P.cost  $\leftarrow \text{cost}(P.\text{solution})$ ;
            Insert P into OPENcbs;
        if N.solution does not have conflicts,
            return N.solution.

```

3.2. Algorithm Design of MCBS

Corresponding to the two-level design of the CBS algorithm, MCBS divides the constraints of multi-UAV path planning into two categories: constraints that must be satisfied in single aircraft planning, such as physical constraints in flight, and the other is space and time constraints between UAVs. We dealt with these two kinds of constraints in the low level and the high level of the algorithm, respectively, to find the optimal solution to satisfy the constraints. For low-level path planning, we designed a sparse A* algorithm based on the flight constraints of a UAV, which solved the problem of the A* algorithm adopted by the CBS algorithm having difficulty dealing with UAV path planning in 3D space. For high-level design, the CBS algorithm only considered the space conflict, whereas MCBS considers both space and time conflicts and sets the priority of conflict resolution. The constraint tree always grows first because of the space conflict, and when there is no space conflict it grows according to the time conflict. The constraint tree will continue to grow until it finds a path combination solution that has neither a time nor space conflict. When searching for the optimal path combination solution in the constraint tree, the search is carried out according to the best priority of the path cost. However, in extreme cases, the algorithm will traverse all path combinations, making the convergence of the algorithm slow. Therefore, we designed a heuristic function to make the algorithm search in the direction of fewer conflicts.

3.2.1. High Level Search

The purpose of the high level is to search different path combinations to find the optimal one through the growth of the constraint tree. We defined the path conflict between UAVs and improved the growth mode of the constraint tree and designed a conflict resolution framework to solve the time and space cooperative problems, as well as a new heuristic function to guide the algorithm search.

(1) Conflict detection and constraint generation

Unlike the grid environment, the path planning space in the 3D environment is larger, so the conflict is more difficult to define. We divided the conflict in the path planning problem of a multi-UAV into two categories: space and time.

(a) Space conflict

When UAVs fly, it is necessary to maintain a safe distance between UAVs at the same time. The two space conflicts we considered were point and edge. Point conflict is shown in Figure 9. Conflict occurs when two aircraft are at the same path point at the same time or when the distance between two points at the same time is less than the safe distance. As shown in Figure 10, the edge is formed by connecting the front and rear path points. When the distance between the points on the edges of two aircraft at the same time is less than the safe distance, edge conflict occurs.

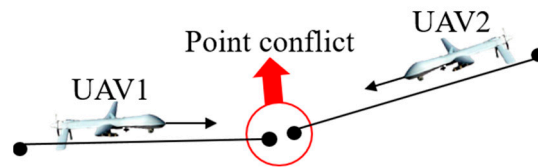


Figure 9. Point conflict.

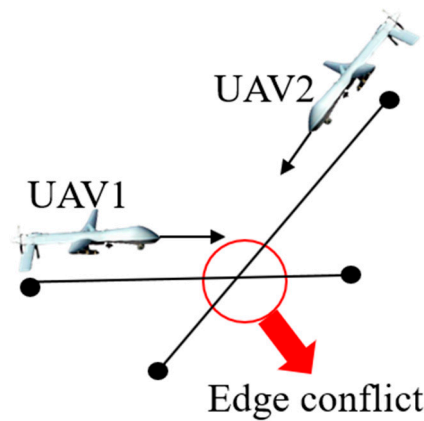


Figure 10. Edge conflict.

Space conflict can be generated as (a_i, a_j, t, D_{safe}) , which means that the mutual distance between UAV a_i and UAV a_j at the path point at time t was less than the safe distance D_{safe} . Space conflict c_s can be generated as (a_i, v_i, t) , where v_i is the waypoint where UAV a_i is in the event of a conflict, which means that UAV a_i is forbidden to be at the waypoint v_i at time t .

(b) Time conflict

Due to mission constraints, multi-UAVs need to reach a target point at the same time or in a certain order. When the speed is given, the UAV arrival time is determined by the path length, so the time can be expressed by the number of waypoints in the UAV path. When the difference between the number of path nodes in the path set is greater than the maximum time difference required for time cooperation, a time conflict occurs.

This conflict can be generated as (a_x, T_{tar}) , which means that the difference between the path waypoints of UAV a_x and the maximum waypoints in the path set is greater than the target time difference T_{tar} . The time constraint c_t can be expressed as (a_x, v_t, t, dS) , and v_t is the path point of the UAV a_x at the time t . The time constraint means that UAV a_x is prohibited from occupying the sphere with path point v_t as its center and radius dS at time t , making this sphere a time-limited area.

(2) The constraint tree

A high-level search uses the best-first search in the nodes of the constraint tree. The growth of the constraint tree represents the increase in the searchable path combination solutions. Compared with CBS, MCBS needs to deal with two conflicts because its constraint tree designs different growth methods for different conflicts.

Each tree node in the constraint tree includes three parts: constraint set, path set, and tree node cost. However, compared with CBS, MCBS subdivides the constraint set into a time set C_{Time} and space set C_{Space} according to the two conflicts.

(3) Create tree nodes

When the constraint tree generates the initial tree node, it does not consider space and time cooperation with other UAVs; instead, it plans a path set that only meets the flight constraints of UAVs as the initial tree node. Each time the constraint tree grows, the tree node with the lowest cost is selected as the current node. When the conflicts of the current node are different, the growth methods of the constraint tree are different. The growth process is shown in Figure 11. If the current node has a time conflict, then the current node is taken as the parent node to generate child nodes. As shown in Figure 11a, the number of child nodes is equal to the number of waypoints l in the path of the UAV with time conflict. Then, a new time constraint c_{ti} is added to the i th child node's time set C_{Time} to solve the conflict. If the current node has a space conflict, the growth mode of the constraint tree is the same as that of the CBS. As shown in Figure 11b, the first conflict in the path set is selected, and then the current node is taken as the parent node to generate two child nodes and add new space constraint c_s for each child node. The child nodes inherit all the information from the parent and update their own path set through a low-level algorithm to provide a feasible solution to the conflict. If the current node has both time and space conflict, the constraint tree of MCBS defines which has priority. It always grows according to the space conflict first. When the current node has no space conflict, it grows according to the time conflict.

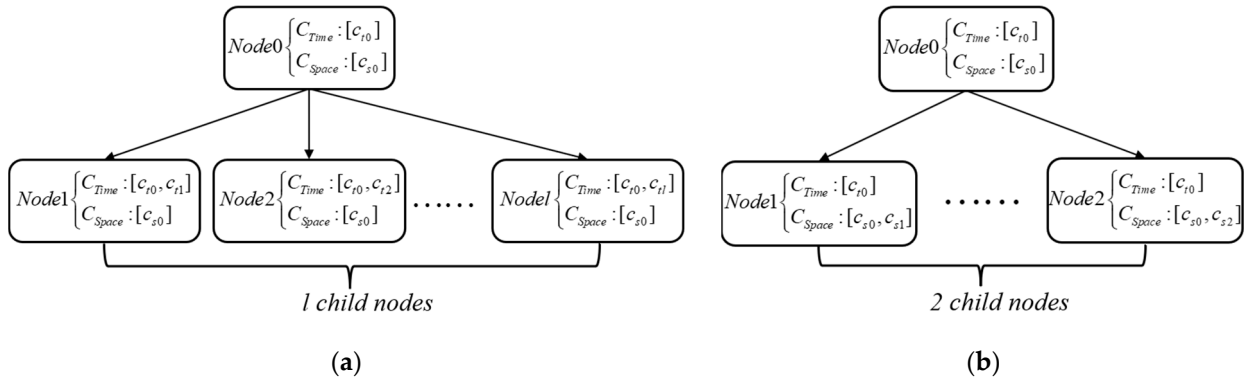


Figure 11. The growth methods of the constraint tree. (a) Tree grows due to time conflict; (b) tree grows due to space conflict.

(4) Tree node cost

At the high level, the algorithm will maintain an open set, select the tree node with the lowest path cost in the open set as the parent node for the next growth, then put the new growth node into the open set and remove the parent node. If the path cost is calculated according to Formula (2), the algorithm will traverse all tree nodes with conflicting connections in extreme cases, which will lead to a huge search space. To reduce both the search space and the time to find the optimal solution, we added the number of space conflicts of paths and the time cooperative violation of paths as heuristic information to the path cost and designed a new cost function to guide the growth of the constraint tree. The new node cost function is defined as follows:

$$F = J + \omega_s \times n_s + \omega_t \times \sum_{i=1}^n (l_{\max} - l_i) \quad (19)$$

where n_s is the number of space cooperative conflicts between UAVs; l_{\max} is the maximum number of waypoints in the path set; l_i is the number of waypoints in the path of the i th

UAV; ω_s and ω_t are space and time cooperative penalty factors; and J is the current total path cost. When the algorithm finds out the target path set, the number of space conflicts between UAVs is equal to zero, and the difference in the number of voyages in the path set n_s is 0. At this time, $F = J$.

(5) Conflict resolution

The flow chart of the conflict resolution framework is shown in Figure 12. The conflict resolution framework can be divided into space conflict and time conflict resolution frameworks, each with different conflict resolution methods. The process is as follows:

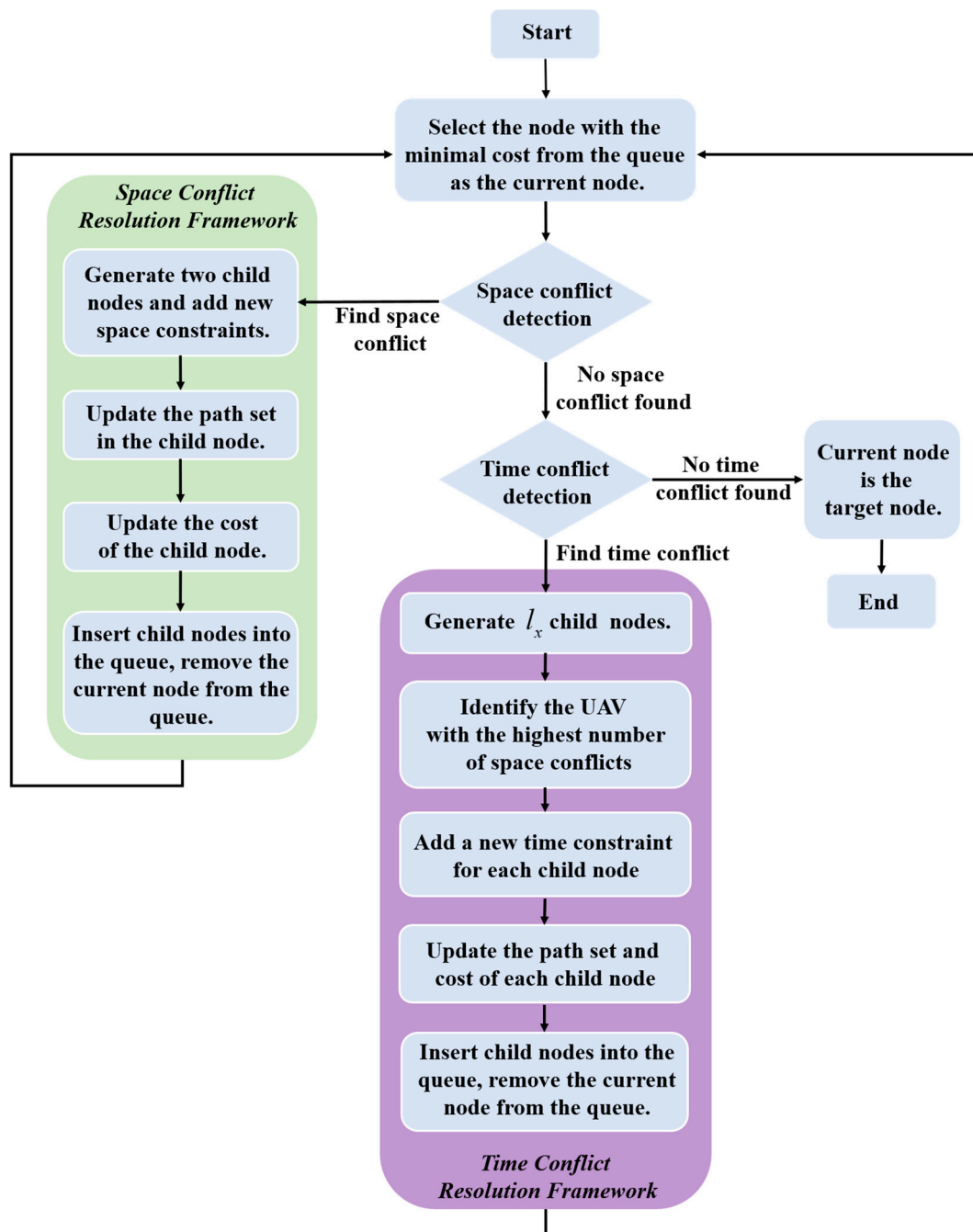


Figure 12. The flow chart of the conflict resolution framework.

For the current tree node, the space conflicts in the path sets are detected first. When space conflict (a_i, a_j, t, D_{safe}) occurs, two child nodes are grown with the current tree node

as the parent node, and the child nodes inherit the original constraint set and add space constraints (a_i, v_i, t) and (a_j, v_j, t) , respectively. Thus, the two solutions to the current space conflict are discussed through the child nodes.

Time conflict is not caused by a fixed path point or an edge but by a certain path in the path set. As shown in Figure 13, the time conflict can be solved by using the time constraint set to make the low-level algorithm bypass the point where the cost is too small and prolong the path length, which is too short. When there is no space conflict in the current node, we look for a time conflict. When the time conflict (a_x, T_{tar}) occurs at the current node, it is used as the parent node to grow l_x child nodes, and each child node randomly selects path point v_t at a time t in the path of UAV a_x as the time constraint (a_x, v_t, t, dS) . Then, the time constraint is added to the original time constraint set of child nodes. l_x is the number of waypoints in the path of UAV a_x , which means that all solutions to the current time conflict are discussed and analyzed through subsequent subnodes to ensure the completeness of the algorithm.

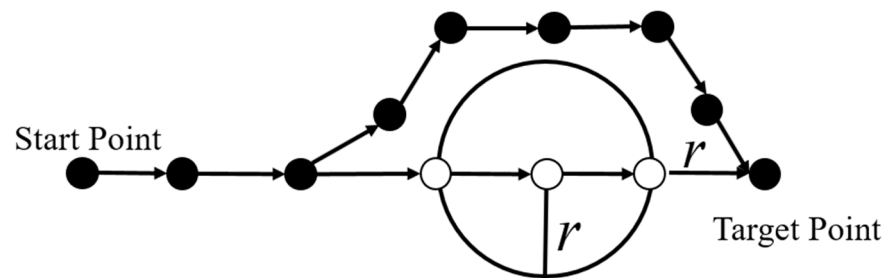


Figure 13. Path extension.

3.2.2. Low Level Search

The algorithm sets the constraint set for each UAV according to the conflict at the high level and finds a path satisfying the constraint set for the UAV at the low level. We adopted the sparse A* algorithm at the low level and, based on it, we eliminated invalid points by combining the flight constraints of the UAV, thus reducing the search space, shortening the search time, and meeting the physical characteristics of UAV, which can be directly applied to its flight.

(1) Waypoint cost function

When the sparse A* algorithm was expanded, the cost function of the n th waypoint could be determined according to the following formula:

$$f(v^n) = f_l(v^i) + f_t(v^i) + h(v^n) \quad (20)$$

where $f_l(v^i)$ is the cumulative length cost; $f_t(v^i)$ is the path threat cost; and $h(v^n)$ is heuristic information. As shown in Formula (21), the cumulative length cost $f_l(v^i)$ is the sum of the distances from the starting point to the current waypoint, and the cumulative threat cost $f_t(v^i)$ is the sum of the threat costs of all waypoints.

$$\begin{cases} f_l(v^i) = \sum_{i=2}^n j_l(v^i) \\ f_t(v^i) = \sum_{i=1}^n j_t(v^i) \end{cases} \quad (21)$$

The heuristic information $h(v^n)$ is the Manhattan distance from the current waypoint to the target waypoint. When the waypoint is identical to the target waypoint, $h(v^n)$ is zero, and the waypoint cost $f_t(v^i)$ is equal to the path cost J_i of the UAV.

(2) Waypoint extension

Successor waypoints can be divided into upper, horizontal, and lower layers according to the different heights. Each layer contains three subsequent waypoints at the same

altitude. When the waypoint is expanded, subsequent waypoints on the horizontal plane are calculated according to the current yaw angle and the minimum turning radius of the UAV; then the maximum climb and maximum dive altitudes that correspond to the aircraft are calculated based on the maximum path slope angle. Finally, the successor waypoints of the upper and lower layers are calculated.

The expansion of the waypoint in the horizontal layer is shown in Figure 14. On the horizontal plane, each waypoint expands three subsequent waypoints at one time. S_j^i indicates that the parent waypoint of the current waypoint S_j is S_i .

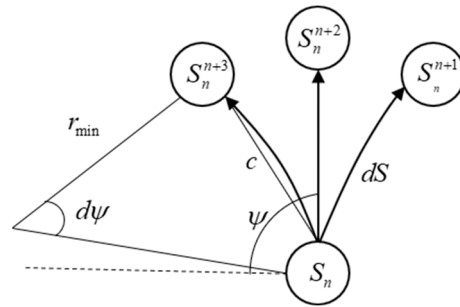


Figure 14. The expansion of the waypoint in the horizontal layer.

In Figure 14, dS is the unit step size of the waypoint and r_{\min} is the minimum turning radius of the UAV. The coordinates of the next generation extended waypoints ($S_n^{n+1}, S_n^{n+2}, S_n^{n+3}$) can be calculated as

$$d\psi = dS / R_{\min} \quad (22)$$

$$c = R_{\min} \sqrt{2 \cdot (1 - \cos(d\psi))} \quad (23)$$

$$S_n^{n+1} = S_n + \begin{bmatrix} -c \cdot \cos(\psi + 0.5d\psi) \\ c \cdot \sin(\psi + 0.5d\psi) \\ d\psi \end{bmatrix} \quad (24)$$

$$S_n^{n+2} = S_n + \begin{bmatrix} -dS \cdot \cos(\psi) \\ dS \cdot \sin(\psi) \\ 0 \end{bmatrix} \quad (25)$$

$$S_n^{n+3} = S_n + \begin{bmatrix} -c \cdot \cos(\psi - 0.5d\psi) \\ c \cdot \sin(\psi - 0.5d\psi) \\ -d\psi \end{bmatrix} \quad (26)$$

3.3. Algorithm Procedure

The multi-UAV cooperative path planning problem is essentially a constrained optimization problem. The MCBS algorithm adopts a two-level design; the low level is used to find the optimal path of UAV, and the high level is used to find the optimal combination path of a multi-UAV. MCBS divides the UAV constraints into flight and cooperative constraints. Flight constraints are resolved at the low level, and collaborative constraints at the high level.

The high level solves the time and space conflicts separately by setting different priorities for the UAV constraints. Corresponding to lines 8 to 19, the algorithm first solves the space conflict through the space constraint set. According to lines 20 to 30, when the current node has no space conflicts, time conflicts are resolved through the time constraint set. Lines 31 to 33 state that when there are no space or time conflicts at the current node, a solution that satisfies the constraints is obtained. To obtain the solution that satisfies the constraints as the optimal solution, the best-first algorithm was carried out in the constraint tree to ensure the optimality of the algorithm, which corresponds to line 26.

The lower level of the algorithm combines the flight constraints of the UAV to limit the waypoint selection so that the final path meets the physical characteristics of the UAV. At the high level, when resolving a space conflict, the constraint tree generates two child nodes, and when resolving a time conflict, the constraint tree generates child nodes with the same number of waypoints in the path where the time conflict occurred. In this way, all solutions to different conflicts are analyzed and evaluated in detail, ensuring the completeness of the algorithm. At the high and low levels, we introduce heuristic information to accelerate convergence and ensure that the results were optimal. The algorithm pseudocode is shown in Algorithm 2.

Algorithm 2 Pseudocode of proposed algorithms.

Input: number of UAV, starting points, target points, D_{safe} , T_{tar} .
Output: optimal solution that satisfies flight constraints and cooperative constraints.

```

Root.time_constraints  $\leftarrow \emptyset$ ;
Root.space_constraints  $\leftarrow \emptyset$ ;
Root.solution  $\leftarrow$  Find path for UAV through low level search;
Root.cost  $\leftarrow cost(Root.solution)$ ;
insert Root into  $OPEN_{cbs}$ ;
while  $OPEN_{cbs} \neq \emptyset$ 
    find the tree node  $N$  with the minimal cost in  $OPEN_{cbs}$ ;
    if  $N.solution$  has space conflicts,
        find the space conflict  $(a_i, a_j, t, D_{safe})$  which occurs first in  $N.solution$ ;
        remove  $N$  from  $OPEN_{cbs}$ ;
        for  $a_x$  in  $[a_i, a_j]$ 
            create a new space constraint set  $c_s = (a_x, v_x, t)$ ;
             $P \leftarrow CreatNode()$ ;
             $P.time\_constraints \leftarrow N.time\_constraints$ ;
             $P.space\_constraints \leftarrow N.space\_constraints + c_s$ ;
             $P.solution \leftarrow N.solution$ ;
             $P.solution(a_x) \leftarrow$  Find path for UAV  $a_x$  through low level search;
             $P.cost \leftarrow cost(P.solution)$ ;
            insert  $P$  into  $OPEN_{cbs}$ ;
    if  $N.solution$  does not have space conflicts but has time conflicts  $(a_x, T_{tar})$ ,
        remove  $N$  from  $OPEN_{cbs}$ ;
        for  $i = 1 : l_x$ 
            Create a new time constraint set  $c_{ti} = (a_x, v_t, t, dS)$ ;
             $P \leftarrow CreatNode()$ ;
             $P.time\_constraints \leftarrow N.time\_constraints + c_{ti}$ ;
             $P.space\_constraints \leftarrow N.space\_constraints$ ;
             $P.solution \leftarrow N.solution$ ;
             $P.solution(a_x) \leftarrow$  Find path for UAV  $a_x$  through low level search;
             $P.cost \leftarrow cost(P.solution)$ ;
            insert  $P$  into  $OPEN_{cbs}$ ;
    if  $N.solution$  has neither space conflicts nor time conflicts,
        break;
    then, return  $N.solution$ .
```

4. Simulation Studies

We simulated and analyzed MCBS in a complex environment. It mainly verified the superiority of a heuristic search at the high-level and the applicability of solving temporal and space cooperation problems in multi-UAV cooperative path planning.

4.1. Environmental and Parametric

In order to be close to the real battlefield environment, we set the parameters of various threats, as shown in Table 1, based on the real environment space in a certain western region as the battlefield setting area.

Table 1. Parameters related to various threats.

Type of Threat		Attributes	
Air defense radar	radar number	coordinates	radius
	radar 0	[300, 400, 20]	80 km
	radar 1	[700, 600, 25]	70 km
Surface-to-air missile	missile number	coordinates	radius
	missile 0	[400, 750, 20]	60 km
Anti-aircraft artillery	artillery number	coordinates	radius
	artillery 0	[600, 180, 25]	30 km
	artillery 1	[750, 320, 20]	30 km
No-fly zone	vertex coordinates: p1 = [520, 320]; p2 = [620, 340]; p3 = [600, 430]; p4 = [520, 430];		

The cooperate constraints imposed on UAVs are affected by the type of mission. We selected the most common allocation and rendezvous task for the multi-UAV. The allocation task was to require the UAV to avoid various threats and reach the airspace near the target point from an initial point, which required high space cooperation. The rendezvous tasks required all UAVs to arrive at the same target point at the same time, which required higher space and time coordination. The parameter settings of the algorithm are shown in Table 2.

Table 2. Parameters related to various tasks.

	Allocation Task	Rendezvous Task
Safe distance D_{safe}	7.5 km	7.5 km
Maximum node difference T_{tar}	0	0
Minimum unit step dS_{min}	25 km	25 km
Minimum turning radius r_{min}	25 km	25 km
Minimum ground height H_{min}	2.5 km	2.5 km

4.2. Algorithm Comparative Analysis

(1) Allocation task

Figures 15 and 16 show the multi-UAV cooperative path planning results of the MCBS algorithm when the number of UAVs performing the allocation tasks was different. As can be seen in Figures 15 and 16, the MCBS algorithm is able to find a safe path for each UAV.

However, it is difficult to see from Figures 15 and 16 whether the set of paths satisfies the cooperation constraints. Therefore, we set the velocity of all UAVs to 600 km/h and calculate the shortest distance between UAVs and the time tolerance of the path set. The time tolerance of the i th UAV is calculated as follows,

$$\Delta t = \frac{l_i - l_m}{v_i}, \quad (27)$$

where l_i is the length of the path of the i th UAV; l_m is the median of all UAV path lengths; and v_i is the velocity of the i th UAV. If the time tolerance of all UAVs is less than 1.5 min, the time cooperative constraints are met. Figure 17 shows shortest distance results between UAVs when five and ten UAVs performed allocation tasks by different algorithms. Figure 18 shows time tolerance results of different algorithms when five and ten UAVs performed allocation tasks by different algorithms.

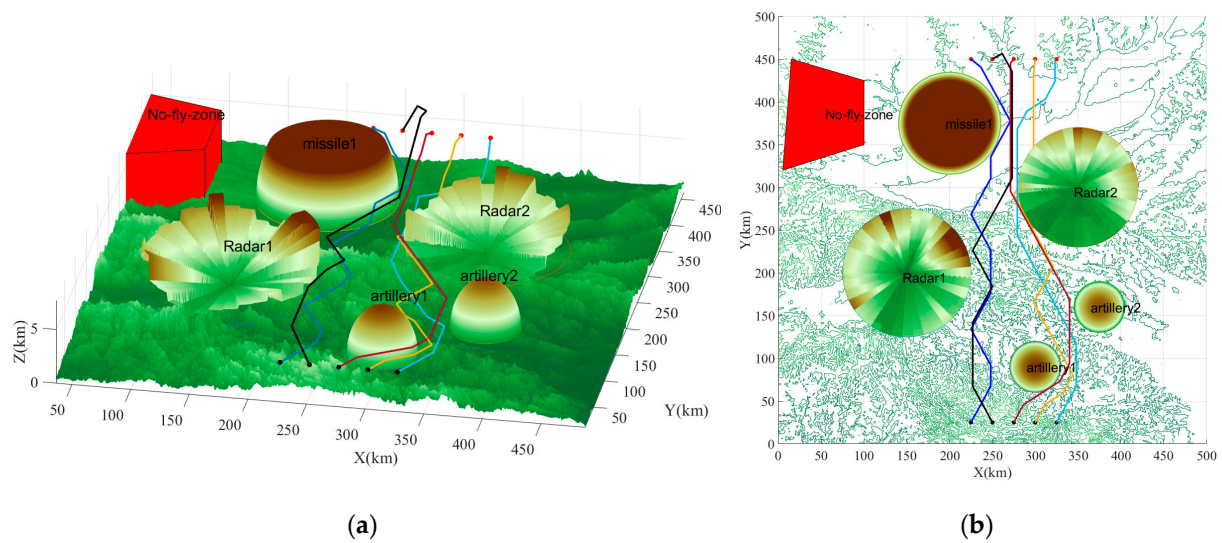


Figure 15. Algorithm simulation results when 5 UAVs perform allocation tasks. The black dots indicate the starting location of UAVs, while the red dots indicate the target location of UAVs. (a) 3D view. (b) Top view.

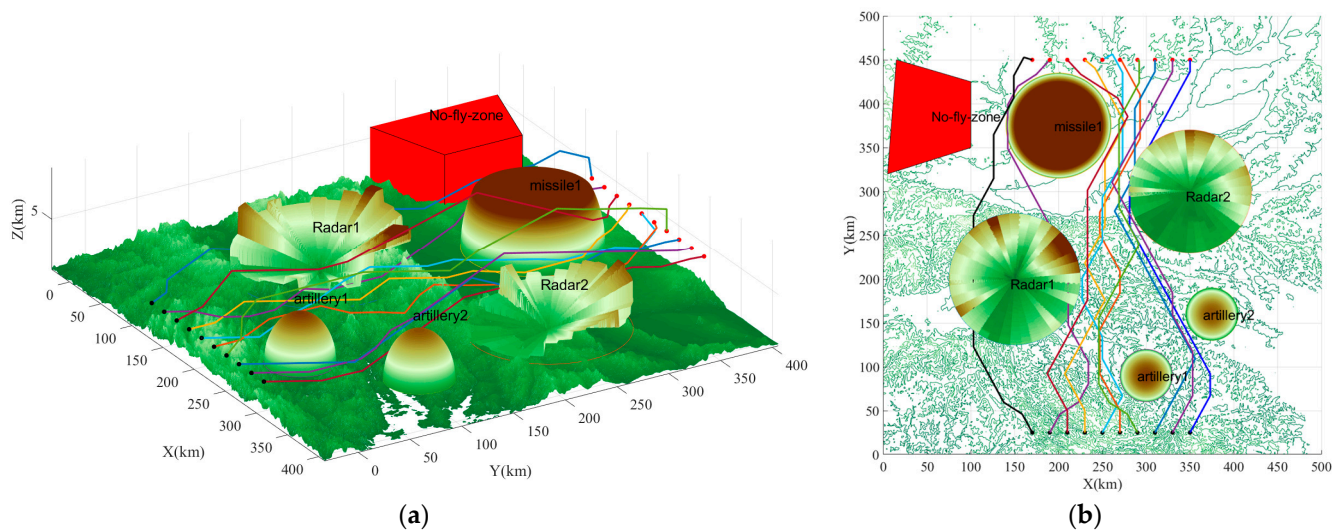


Figure 16. Algorithm simulation results when 10 UAVs perform allocation tasks. The black dots indicate the starting location of UAVs, while the red dots indicate the target location of UAVs. (a) 3D view. (b) TOP view.

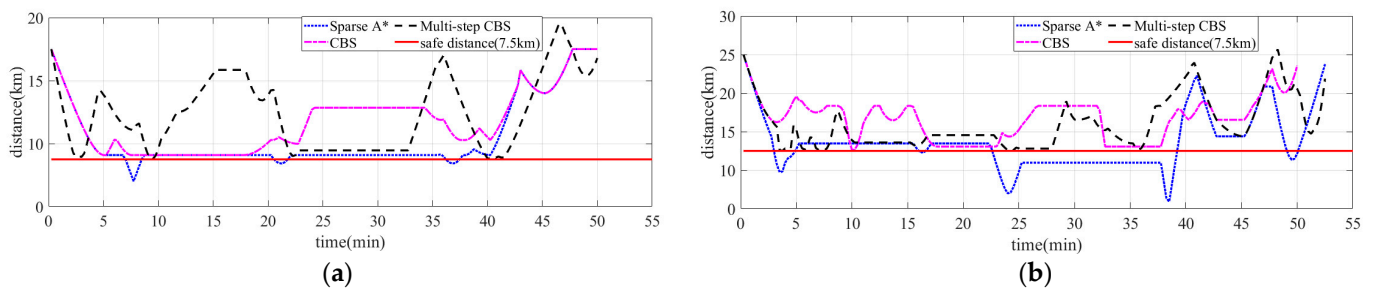


Figure 17. Shortest distance results of different algorithms. (a) Shortest distance between 5 UAVs. (b) Shortest distance between 10 UAVs.

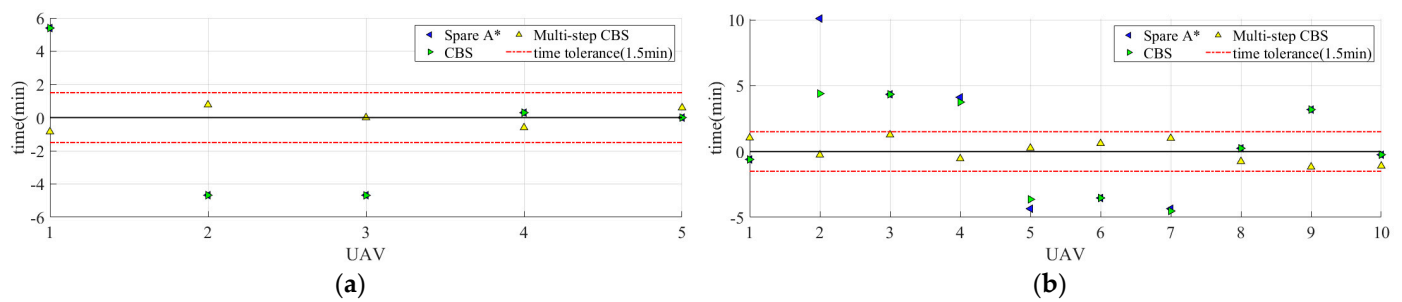


Figure 18. Time tolerance results of different algorithms when UAVs perform allocation tasks. (a) Time tolerance of 5 UAVs. (b) Time tolerance of 10 UAVs.

To further verify that the paths satisfy the flight constraints and to evaluate the set of paths planned by different algorithms, we present the constraints satisfaction results for different algorithms in Table 3. From Table 3, it can be seen that the paths planned by all three algorithms can satisfy the flight constraints of UAVs. However, the results of sparse A* cannot handle the cooperative constraints, and CBS handled the space constraints well, but not the time constraints. Only the set of paths planned by the MCBS algorithm can meet all the constraints.

Table 3. Paths' constraints evaluation by different algorithms when 10 UAVs perform allocation tasks.

	Sparse A*	CBS	MCBS
Minimum unit step of the paths	25.57 km	25.36 km	25.11 km
Minimum turning radius of the paths	25.54 km	25.38 km	25.21 km
Minimum ground height of the paths	3.62 km	3.62 km	2.97 km
Maximum node difference of the paths	4	4	0
Shortest distance of the paths	0.93 km	7.60 km	7.51 km
Maximum time tolerance of the paths	10.10 min	4.52 min	1.26 min

(2) Rendezvous task

Figures 19 and 20 shows the multi-UAV cooperative path planning results of the MCBS algorithm when the number of UAVs was different when performing rendezvous tasks. Figure 21 shows the path length and shortest distance results at the same moment, planned by different algorithms when five and ten UAVs performed a rendezvous task. Figure 22 shows the time tolerance results of different algorithms when five and ten UAVs performed rendezvous tasks by different algorithms. Table 4 presents the constraints meet results for different algorithms. As can be seen from Figures 19 and 20, MCBS is capable of planning safe paths for UAVs performing the rendezvous task. From Figures 21 and 22 and Table 4, MCBS can solve the time and space cooperative constraints and found the optimal path set for multi-UAVs to fly safely and meet the flight constraints.

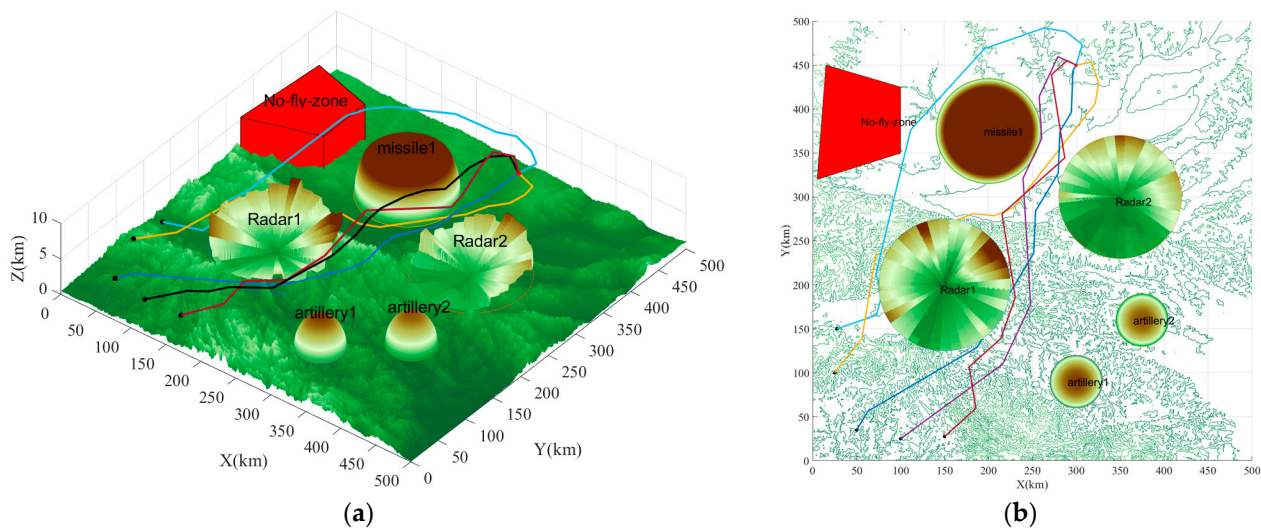


Figure 19. Algorithm simulation results when 5 UAVs perform rendezvous tasks. (a) 3D view. (b) Top view.

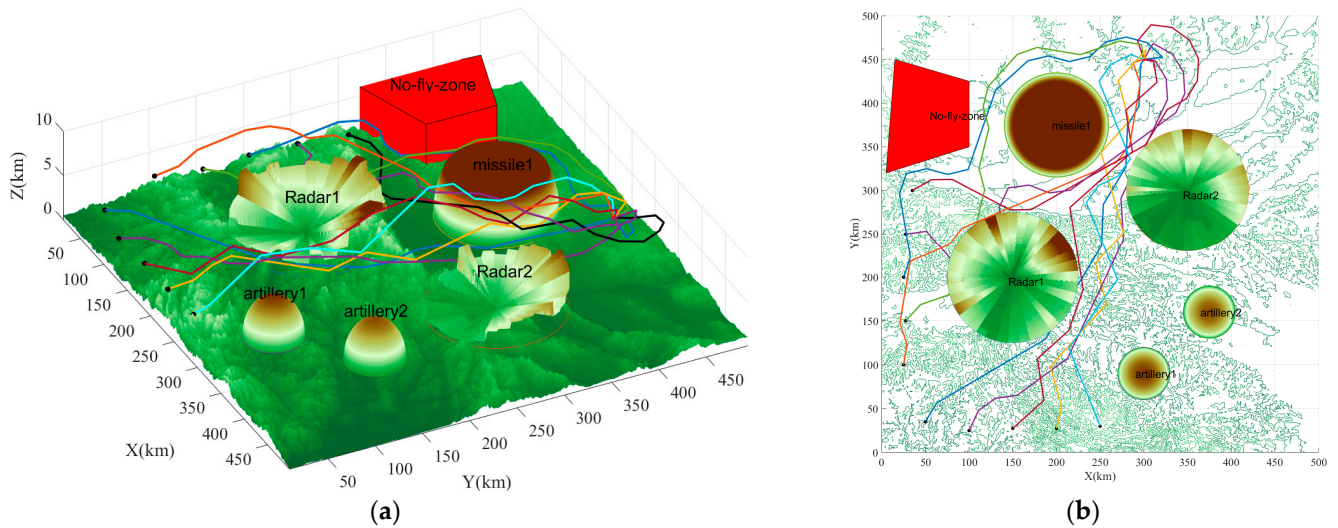


Figure 20. Algorithm simulation results when 10 UAVs perform rendezvous tasks. (a) 3D view. (b) Top view.

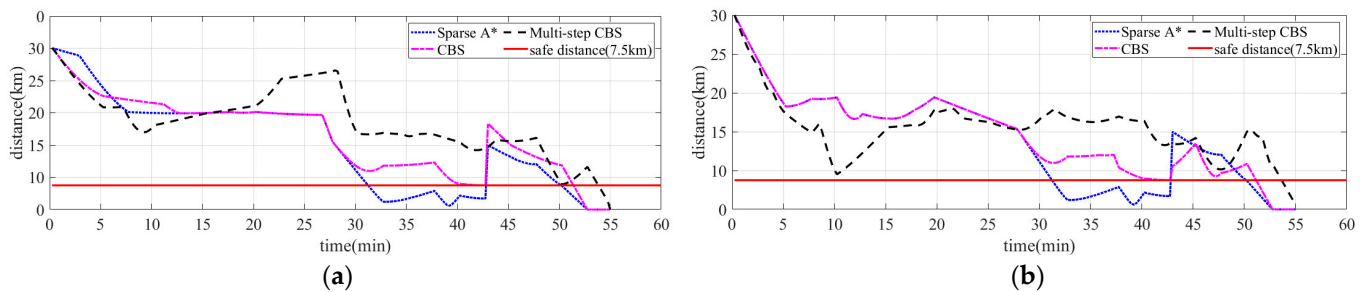


Figure 21. Shortest distance results of different algorithms. Note that the experimental scenario is a rendezvous task scenario, so collision checking ends when it reaches the path nodes before the target node (i.e., assuming there are no collisions in the target region). (a) Shortest distance between 5 UAVs. (b) Shortest distance between 10 UAVs.

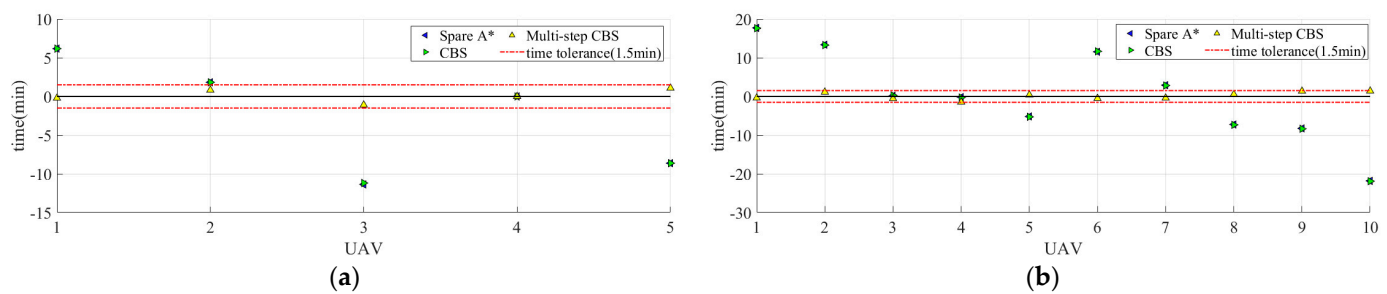


Figure 22. Time tolerance results of different algorithms when UAVs perform rendezvous tasks. (a) Time tolerance of 5 UAVs. (b) Time tolerance of 10 UAVs.

Table 4. Paths' constraints evaluation by different algorithms when 10 UAVs perform rendezvous tasks. Note that the target point is removed when calculating the shortest distance.

	Sparse A*	CBS	MCBS
Minimum unit step of the paths	25.31 km	25.31 km	25.15 km
Minimum turning radius of the paths	25.85 km	25.85 km	25.31 km
Minimum ground height of the paths	3.86 km	3.86 km	3.1 km
Maximum node difference of the paths	4	4	0
Shortest distance of the paths	1.20 km	7.52 km	9.02 km
Maximum time tolerance of the paths	21.90 min	21.90 min	1.40 min

4.3. Parametric Analysis

4.3.1. Task Parameter Analysis

Based on the above environment, we selected different starting and ending points many times to test the runtime of our algorithm under different parameters. The number of UAVs varied from 3 to 23; the maximum time difference T_{tar} from 0 to 2 min; and safe distance D_{safe} from 0 to 7.5 km in increments of 2.5 km. The test results are shown in Figure 23.

4.3.2. Penalty Factor Parameter Analysis

Previously, we introduced heuristic information including penalty factors in the cost function of tree nodes. To verify the effectiveness of this scheme, we analyzed the impact of the penalty factor on the success rate of the algorithm when performing different tasks based on the environment in 4.1. We stipulated that the algorithm failed when it could not give a valid solution within 5 min. The simulation results are shown in Figure 24.

The size of the penalty factor influenced the importance of space and time cooperation when the constraint tree grew. As it did, we prioritized resolving space conflicts. Then, on the basis that the space conflicts have been resolved, the constraint tree was made to grow in the direction of resolving time conflicts without generating a new space conflict. Therefore, as can be seen from Figure 24, when $\omega_t = 5$ and $\omega_s = 10$ such as that $\omega_s \times \Delta n_s > \omega_t \times \Delta \sum_{i=1}^n (l_{\max} - l_i) > \Delta J$, the algorithm was the most efficient.

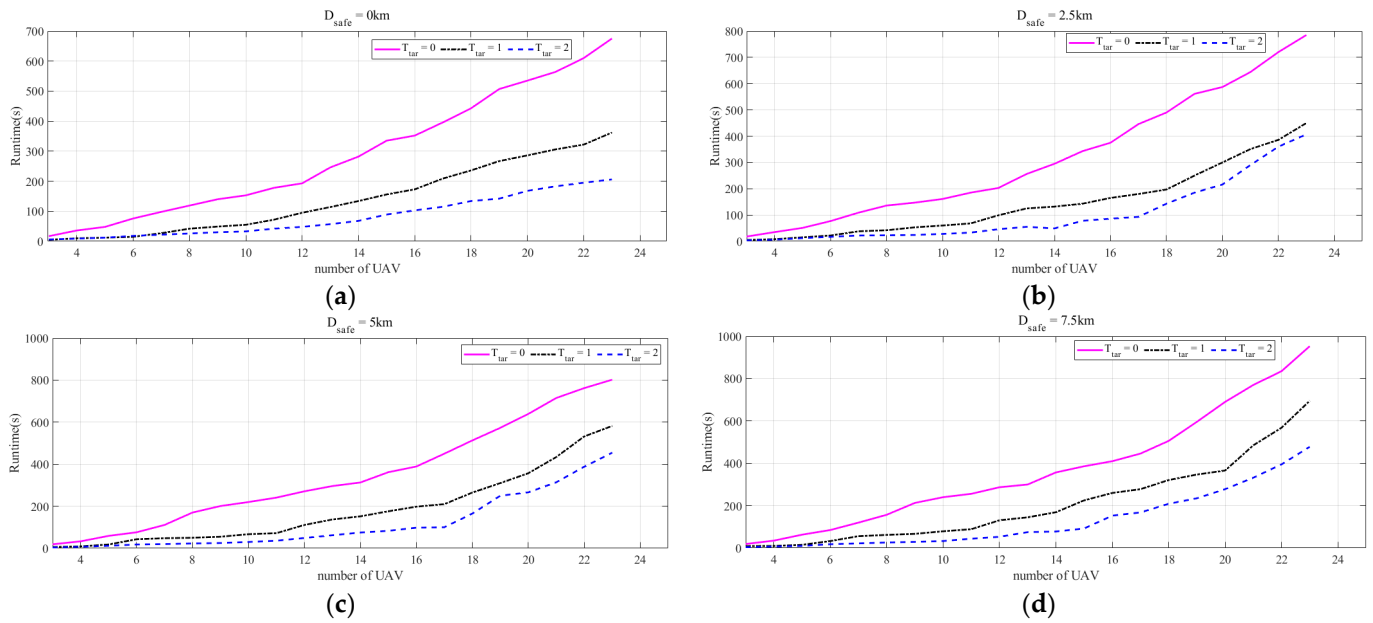


Figure 23. The runtime of MCBS under different parameters. (a) $D_{safe} = 0$ km. (b) $D_{safe} = 2.5$ km. (c) $D_{safe} = 5$ km. (d) $D_{safe} = 7.5$ km. The runtime related positively to the number of conflicts it needed to deal with. When the number of UAVs was fewer than 15, the number of conflicts was relatively small, and the optimal solution to the problem was found within 5 min. When the time and space cooperative conditions became more stringent or the number of UAVs increased, the number of collaborative conflicts increased rapidly, as did the time used by the MCBS.

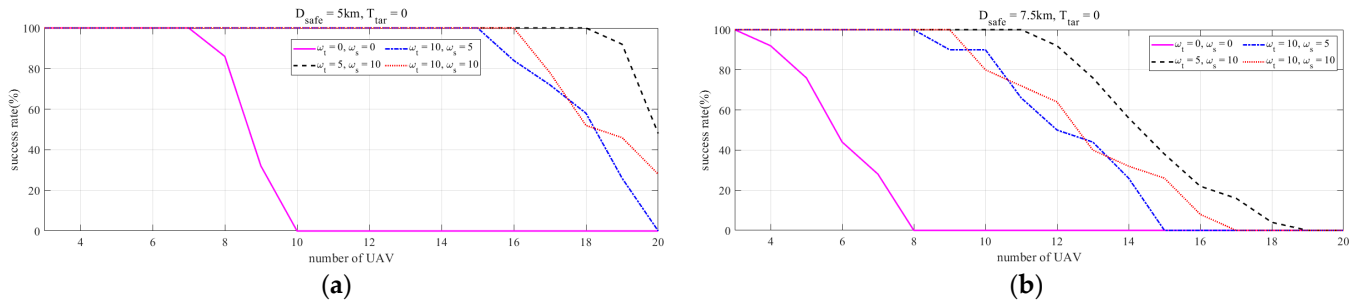


Figure 24. The success rate of MCBS under different penalty factors. (a) $D_{safe} = 10$, $T_{tar} = 1$. (b) $D_{safe} = 10$, $T_{tar} = 0$.

5. Conclusions

In this study, a cooperative path planning algorithm for UAVs based on multi-collision detection was proposed. A complex mission environment model was established based on UAV flight and cooperative constraints, and a spatial and temporal collaborative solution framework was designed for multi-UAV collaborative problems. A three-dimensional environment path planning method based on an improved sparse A* was designed for a single aircraft. For multi-UAV cooperation, the collaborative conflict between multiple UAVs was defined; the priority of different collaborative conflicts was set; the growth mode of the constraint tree was changed; and a new heuristic function was designed to guide the search to reduce the convergence time of the algorithm. A comparison was made among the planning results of this algorithm, the traditional CBS, and the sparse A* algorithm for different UAV group tasks in a complex mission environment. The simulation results showed that the proposed algorithm could deal with the coupling problem of time and space cooperation in complex environments and find the optimal safe cooperative path that satisfied the flight and cooperation constraints for multiple UAVs.

In the multi-UAV cooperative track planning problem, this paper mainly focused on the static threat before the start of the mission; in fact, the threat in the mission environment was often highly dynamic. Therefore, on the basis of cooperative track planning research, more dynamic scenarios such as mobile threats and random mobile threats can be considered to improve the adaptability of cooperative track planning to a complex mission environment.

In the future work, we plan to apply the MCBS algorithm to a multi-UAVs system with local communication for real outdoor experiments. In the planned experiments, the proposed algorithm requires each UAV to have an independent communication unit to achieve real-time communication. Therefore, ensuring the reliability and real-time of communication between UAVs is also a challenge to be faced during the implementation of this algorithm. In addition, we plan to further evaluate the algorithm through real experiments to study how to handle dynamic threats.

Author Contributions: Conceptualization, X.L. and Y.S.; methodology, X.L.; software, X.L.; validation, X.L.; formal analysis, X.L.; investigation, X.L. and Y.S.; resources, X.L. and Y.W.; data curation, X.L. and Y.G.; writing—original draft preparation, X.L.; writing—review and editing, X.L.; visualization, X.L.; supervision, X.L., Y.S., Y.W. and Y.G.; project administration, X.L.; funding acquisition, X.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, grant number No. 62073266, and the Aeronautical Science Foundation of China, grant number No. 201905053003.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: Gratitude is extended to the Shaanxi Province Key Laboratory of Flight Control and Simulation Technology.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Fu, B.; Chen, L.; Zhou, Y.; Zheng, D.; Wei, Z.; Dai, J.; Pan, H. An improved A* algorithm for the industrial robot path planning with high success rate and short length. *Rob. Auton. Syst.* **2018**, *106*, 26–37. [\[CrossRef\]](#)
2. Gupta, L.; Jain, R.; Vaszkun, G. Survey of Important Issues in UAV Communication Networks. *IEEE Commun. Surv. Tutorials* **2016**, *18*, 1123–1152. [\[CrossRef\]](#)
3. Patle, B.K.; Babu, L.G.; Pandey, A.; Parhi, D.R.K.; Jagadeesh, A. A review: On path planning strategies for navigation of mobile robot. *Def. Technol.* **2019**, *15*, 582–606. [\[CrossRef\]](#)
4. Alpdemir, M.N. Tactical UAV path optimization under radar threat using deep reinforcement learning. *Neural Comput. Appl.* **2022**, *34*, 5649–5664. [\[CrossRef\]](#)
5. Pan, Y.; Yang, Y.; Li, W. A Deep Learning Trained by Genetic Algorithm to Improve the Efficiency of Path Planning for Data Collection with Multi-UAV. *IEEE Access* **2021**, *9*, 7994–8005. [\[CrossRef\]](#)
6. Ahmed, F.; Mohanta, J.C.; Keshari, A.; Yadav, P.S. Recent Advances in Unmanned Aerial Vehicles: A Review. *Arab. J. Sci. Eng.* **2022**, *47*, 7963–7984. [\[CrossRef\]](#) [\[PubMed\]](#)
7. Roberge, V.; Tarbouchi, M.; Labonte, G. Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning. *IEEE Trans. Ind. Inform.* **2013**, *9*, 132–141. [\[CrossRef\]](#)
8. Majumder, S.; Prasad, M.S. Three dimensional D* algorithm for incremental path planning in uncooperative environment. In Proceedings of the 2016 3rd International Conference on Signal Processing and Integrated Networks (SPIN), Noida, India, 11–12 February 2016; pp. 431–435. [\[CrossRef\]](#)
9. Dhawale, A.; Yang, X.; Michael, N. Reactive Collision Avoidance Using Real-Time Local Gaussian Mixture Model Maps. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 3545–3550. [\[CrossRef\]](#)
10. Cap, M.; Novak, P.; Kleiner, A.; Selecky, M. Prioritized Planning Algorithms for Trajectory Coordination of Multiple Mobile Robots. *IEEE Trans. Autom. Sci. Eng.* **2015**, *12*, 835–849. [\[CrossRef\]](#)
11. Damani, M.; Luo, Z.; Wenzel, E.; Sartoretti, G. PRIMAL2: Pathfinding Via Reinforcement and Imitation Multi-Agent Learning-Lifelong. *IEEE Robot. Autom. Lett.* **2021**, *6*, 2666–2673. [\[CrossRef\]](#)

12. Van Den Berg, J.; Guy, S.J.; Lin, M.; Manocha, D. Reciprocal n-body collision avoidance. In *Springer Tracts in Advanced Robotics*; Springer: Berlin/Heidelberg, Germany, 2011.
13. D'Amato, E.; Mattei, M.; Notaro, I. Distributed Reactive Model Predictive Control for Collision Avoidance of Unmanned Aerial Vehicles in Civil Airspace. *J. Intell. Robot. Syst. Theory Appl.* **2020**, *97*, 185–203. [[CrossRef](#)]
14. Lalish, E.; Morgansen, K.A. Distributed reactive collision avoidance. *Auton. Robots* **2012**, *32*, 207–226. [[CrossRef](#)]
15. Zu, C.; Yang, C.; Wang, J.; Gao, W.; Cao, D.; Wang, F.Y. Simulation and field testing of multiple vehicles collision avoidance algorithms. *IEEE/CAA J. Autom. Sin.* **2020**, *7*, 1045–1063. [[CrossRef](#)]
16. Silver, D. Cooperative Pathfinding.pdf. In Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, Marina Del Rey, CA, USA, 1–2 June 2005; pp. 117–122.
17. Tai, R.; Wang, J.; Chen, W. A prioritized planning algorithm of trajectory coordination based on time windows for multiple AGVs with delay disturbance. *Assem. Autom.* **2019**, *39*, 753–768. [[CrossRef](#)]
18. Cap, M.; Novak, P.; Selecky, M.; Faigl, J.; Vokffnek, J. Asynchronous decentralized prioritized planning for coordination in multi-robot system. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 3822–3829. [[CrossRef](#)]
19. Liu, Z.; Wei, H.; Wang, H.; Li, H.; Wang, H. Integrated Task Allocation and Path Coordination for Large-Scale Robot Networks With Uncertainties. *IEEE Trans. Autom. Sci. Eng.* **2022**, *19*, 2750–2761. [[CrossRef](#)]
20. Panescu, D.; Pascal, C. A constraint satisfaction approach for planning of multi-robot systems. In Proceedings of the 2014 18th Int. Conf. Syst. Theory, Control Comput. ICSTCC 2014, Sinaia, Romania, 17–19 October 2014; pp. 157–162. [[CrossRef](#)]
21. Sharon, G.; Stern, R.; Felner, A.; Sturtevant, N.R. Conflict-based search for optimal multi-agent pathfinding. *Artif. Intell.* **2015**, *219*, 40–66. [[CrossRef](#)]
22. Tinka, A.; Durham, J.W.; Koenig, S. Lifelong Multi-Agent Path Finding in Large-Scale Warehouses Extended Abstract. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; pp. 1–3.
23. Semiz, F.; Polat, F. Incremental multi-agent path finding. *Futur. Gener. Comput. Syst.* **2021**, *116*, 220–233. [[CrossRef](#)]
24. Barer, M.; Sharon, G.; Stern, R.; Felner, A. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In Proceedings of the International Symposium on Combinatorial Search, Prague, Czech Republic, 15–17 August 2014; pp. 19–27. [[CrossRef](#)]
25. Li, J.; Ruml, W.; Koenig, S. EECBS: A Bounded-Suboptimal Search for Multi-Agent Path Finding. *AAAI Conf. Artif. Intell. AAAI* **2021**, *14A*, 12353–12362. [[CrossRef](#)]
26. Bayerlein, H.; Theile, M.; Caccamo, M.; Gesbert, D. Multi-UAV Path Planning for Wireless Data Harvesting with Deep Reinforcement Learning. *IEEE Open J. Commun. Soc.* **2021**, *2*, 1171–1187. [[CrossRef](#)]
27. Zhang, Z.; Wu, J.; Dai, J.; He, C. A Novel Real-Time Penetration Path Planning Algorithm for Stealth UAV in 3D Complex Dynamic Environment. *IEEE Access* **2020**, *8*, 122757–122771. [[CrossRef](#)]
28. Xu, C.; Xu, M.; Yin, C. Optimized multi-UAV cooperative path planning under the complex confrontation environment. *Comput. Commun.* **2020**, *162*, 196–203. [[CrossRef](#)]
29. Yang, P.; Tang, K.; Lozano, J.A.; Cao, X. Path Planning for Single Unmanned Aerial Vehicle by Separately Evolving Waypoints. *IEEE Trans. Robot.* **2015**, *31*, 1130–1146. [[CrossRef](#)]
30. Besada-Portas, E.; De La Torre, L.; De La Cruz, J.M.; De Andrés-Toro, B. Evolutionary trajectory planner for multiple UAVs in realistic scenarios. *IEEE Trans. Robot.* **2010**, *26*, 619–634. [[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.