drones

MDPI

Article

PPSwarm: Multi-UAV Path Planning Based on Hybrid PSO in Complex Scenarios

Qicheng Meng †, Kai Chen † and Qingjun Qu *

College of Systems Engineering, National University of Defense Technology, Changsha 410073, China; mengqicheng18@nudt.edu.cn (Q.M.); chenkai@nudt.edu.cn (K.C.)
* Correspondence: quqingjun24@nudt.edu.cn
† These authors contributed equally to this work.

Abstract: Evolutionary algorithms exhibit flexibility and global search advantages in multi-UAV path planning, effectively addressing complex constraints. However, when there are numerous obstacles in the environment, especially narrow passageways, the algorithm often struggles to quickly find a viable path. Additionally, collaborative constraints among multiple UAVs complicate the search space, making algorithm convergence challenging. To address these issues, we propose a novel hybrid particle swarm optimization algorithm called PPSwarm. This approach initially employs the RRT* algorithm to generate an initial path, rapidly identifying a feasible solution in complex environments. Subsequently, we adopt a priority planning method to assign priorities to UAVs, simplifying collaboration among them. Furthermore, by introducing a path randomization strategy, we enhance the diversity of the particle swarm, thereby avoiding local optimum solutions. The experimental results show that, in comparison to algorithms such as DE, PSO, ABC, GWO, and SPSO, the PPSwarm algorithm demonstrates significant advantages in terms of path quality, convergence speed, and runtime when addressing path planning issues for 40 UAVs across four different scenarios. In larger-scale experiments involving 500 UAVs, the proposed algorithm also exhibits excellent processing capability and scalability.

Keywords: unmanned aerial vehicle (UAV); path planning; particle swarm optimization; prioritized planning methods; RRT*

Citation: Meng, Q.; Chen, K.; Qu, Q. PPSwarm: Multi-UAV Path Planning Based on Hybrid PSO in Complex Scenarios. *Drones* **2024**, *8*, 192. https://doi.org/10.3390/drones8050192

Academic Editor: Francesco Nex

Received: 17 April 2024
Revised: 5 May 2024
Accepted: 8 May 2024
Published: 11 May 2024

1. Introduction

With the continuous advancement of intelligent technology, unmanned aerial vehicle (UAV) technology has become one of the hotspots of technological innovation today. Especially in scenarios that require the collaborative work of numerous UAVs, ensuring their efficient and safe completion of tasks is particularly crucial. Path planning is a key technology in the application of UAV swarms. It involves the coordination and cooperation of multi-UAVs during task execution, directly affecting the efficiency of task completion and the safety of the UAVs. An excellent path planning scheme needs to comprehensively consider various factors such as collisions between UAVs, environmental factors, and mission requirements, to ensure that UAVs can complete tasks with minimal time and space consumption while guaranteeing their own safety.

After years of development, significant progress has been made in multi-UAV path planning algorithms. Existing path planning algorithms are mainly divided into two categories: traditional algorithms and heuristic algorithms. Excellent traditional algorithms include the A* algorithm [1,2], the Rapidly-exploring Random Tree (RRT) algorithm [3,4], artificial potential field [5], etc. As the complexity of problems increases, traditional methods often cannot solve NP problems such as 3D path planning, so heuristic algorithms have gradually become the mainstream approach for solving such problems. Heuristic algorithms mainly include Differential Evolution (DE) [6,7], Ant Colony Optimization (ACO) [8,9], Particle Swarm Optimization (PSO) [10–12], the Genetic Algorithm

(GA) [13,14], the Artificial Bee Colony (ABC) algorithm [15], the Firefly Algorithm (FA) [16], the Teaching–Learning-Based Optimization (TLBO) algorithm [17], and other heuristic algorithms that are currently widely used in multi-UAV path planning. Based on different principles and strategies, these algorithms can find effective flight paths in complex environments.

Evolutionary algorithms exhibit strong flexibility and adaptability in dealing with multi-UAV path planning problems. They can easily handle various complex constraints (such as threat zones, flight altitude restrictions, turning requirements, etc.) and optimization objectives (such as flight time, safety, energy consumption, etc.). Hui et al. [18] proposed an asynchronous Ant Colony Optimization algorithm that solves the problem of detecting large and complex buildings through an asynchronous forward strategy. Nafis Ahmed [19] derived distributed full-coverage optimal path planning using the PSO algorithm, while Wang et al. [20] designed a method based on the Lévy flight search strategy and the improved velocity-dependent Bat algorithm. These methods demonstrate the effectiveness of swarm intelligence in solving complex path planning problems. Using evolutionary algorithms alone has issues such as slow convergence speed and parameter sensitivity, leading to the emergence of various hybrid algorithms. For example, the combination of evolutionary algorithms and reinforcement learning, such as the multi-strategy Cuckoo Search algorithm based on reinforcement learning proposed by Yu et al. [21], improves the convergence speed of optimization methods. Additionally, Mickey et al. [22] used Genetic Algorithm optimization methods to find the RA-MCPP path planning that maximizes PoC, while Chen et al. [23] calculated the quasi-optimal trajectory of a rotorcraft using an improved Wolf Pack Search algorithm. To improve the overall performance of the algorithm, researchers have also proposed hybrid algorithms and hierarchical strategies. For instance, Qu et al. [24] combined the Simplified Grey Wolf Optimizer (SGWO) and the Modified Symbiotic Organisms Search (MSOS) to propose a new hybrid algorithm, HSGWO-MSOS, aimed at solving complex domain problems. Yang et al. [25] proposed a Hierarchical Recursive Multi-Agent Genetic Algorithm (HR-MAGA) to achieve real-time path planning.

Among these algorithms, swarm intelligence algorithms such as Particle Swarm Optimization (PSO) have shown particularly impressive performance in multi-UAV path planning. Known for its simplicity and efficiency, the PSO algorithm has achieved remarkable results. In recent years, many researchers have attempted to leverage the PSO algorithm to tackle path planning problems. Phung et al. [26] proposed an enhanced Discrete Particle Swarm Optimization (DPSO) algorithm for solving the Traveling Salesman Problem (TSP). A distributed PSO-based exploration algorithm to aid in disaster scenarios was introduced by [27]. Xiande et al. [28] presented a method that utilizes the Rauch–Tung–Striebel (RTS) smoothing algorithm to optimize the parameters affecting the performance of the PSO algorithm, aiming to reduce efficiency losses and the occurrence of suboptimal solutions when using PSO for path planning. Das et al. [29] suggested a hybrid approach combining Improved Particle Swarm Optimization (IPSO) with the Improved Gravitational Search Algorithm (IGSA) to determine optimal trajectories for multi-robot paths in cluttered environments. P.K. Das et al. [30] proposed a method that combines Improved Particle Swarm Optimization (IPSO) with a Differential Perturbation Velocity (DV) algorithm to determine optimal trajectories for multi-robot paths in cluttered environments. This approach adjusts the robots' velocities by incorporating differential evolution (DE) vector differential operators inherited from IPSO. Yu et al. [31] introduced a new hybrid Particle Swarm Optimization (PSO) algorithm called SDPSO, which avoids local optima by incorporating a simulated annealing algorithm. Ji et al. [32] proposed a novel Dual Dynamic Biogeography-Based Learning Particle Swarm Optimization (DDBLPSO) algorithm to optimize convergence efficiency. He et al. [33] adopted a Timestamp Segmentation (TSS) model to simplify the handling of UAV coordination costs. They then combined Improved Particle Swarm Optimization (IPSO) with Modified Symbiotic Organisms Search (MSOS)

to propose a new hybrid algorithm called HIPSOMSOS. Therefore, this article chooses to use a PSO hybrid algorithm for multi-UAV path planning.

However, using the PSO method for multi-UAV path planning still faces the following challenges: (1) When there are a large number of obstacles in the environment, especially in narrow passages, it is difficult for evolutionary algorithms to find feasible solutions. (2) It is challenging to handle the collaborative constraints of multi-UAVs. Due to the numerous collision avoidance constraints that need to be satisfied among the paths of multi-UAVs, it is difficult for the particles to update their paths towards the current optimal direction, which leads to difficulties in convergence during the optimization process. (3) It is hard to ensure the diversity of the particle swarm, which leads to the problem of converging too quickly and falling into a local optimal solution.

To address the first challenge, we use the RRT* algorithm to generate initial paths. The RRT* algorithm possesses the advantages of asymptotic optimality and rapid solution finding in solving path planning problems. As the number of sampling points increases, it gradually discovers paths closer to the optimal one, showcasing good adaptability and flexibility. To solve the second challenge, we utilize a problem decoupling approach to convert multi-UAV path planning into single-UAV path planning, thereby diminishing problem complexity. Specifically, we adopt the Prioritized Planning (PP) method [34], assigning a priority to each agent and planning in descending order of priority. Each agent must avert collisions with higher-priority agents and obstacles. This method effectively resolves collisions among multi-UAVs, diminishes computational costs by reducing the number of evaluations, and consequently lowers the computational complexity of the algorithm. To tackle the third challenge, we introduce random path generation into the initial paths and integrate path randomization during subsequent iterations to augment particle diversity.

By combining the advantages of the Priority Planning method and the RRT* algorithm, this paper proposes a novel hybrid Particle Swarm Optimization (PSO) algorithm named PPSwarm to address the multi-UAV path planning problem. Additionally, to enhance the flexibility of the algorithm and reduce the difficulty of problem-solving, a two-level path planning strategy consisting of a high-level and a low-level strategy is introduced. At the high level, the Rapidly-exploring Random Tree (RRT*) and the Priority Planning (PP) algorithms are utilized to initialize the flight paths of UAVs and assign priorities to the UAVs. At the low level, high-priority individual UAVs employ the PSO algorithm for path planning, while the initial particle swarm selectively inherits the results of the RRT* initialization. In the proposed algorithm, we fully integrate the local optimization capabilities of the PSO algorithm with the global search capabilities of the RRT* algorithm, improving the efficiency of finding feasible solutions. Moreover, the Priority Planning algorithm is used to assign priorities to the UAVs, resolving potential collisions between them. Experimental results in four scenarios with 40 UAVs demonstrate that the proposed PPSwarm algorithm exhibits better performance in terms of path quality and convergence speed. Furthermore, experimental outcomes on a problem instance with 500 UAVs demonstrate the algorithm's ability to solve large-scale problems.

The remainder of this paper is organized as follows: Section 2 designs the cost function for the multi-UAV path planning problem. Section 3 explains the basic principles of the PSO, RRT*, and priority planning algorithms. Section 4 details the proposed PPSwarm algorithm. Section 5 presents the comparative experiments with the algorithms, as well as the analysis of algorithmic schemes and parameters. Finally, Sections 6 and 7 are the discussion and conclusions, respectively.

2. Problem Description

In this paper, our core objective is to determine a set of optimal or sub-optimal flight paths from the starting point to the destination while ensuring safety requirements. These paths are designed for multiple UAVs. Safety refers to the ability of UAVs to successfully avoid obstacles in complex environments and prevent collisions among themselves.

Let the flight environment be denoted by E and the number of UAVs be denoted by M, with each UAV traveling at a speed v_{uav}. We define the set of all UAV tasks as T. For each UAV $m \in \{1, \ldots, M\}$, its specific task T_m includes the starting point x_m^s and the destination point x_m^g. Assuming that the path from the starting point to the destination point consists of N_w waypoints, the path of UAV m is denoted by:

$$P^m = \{P_1^m, P_2^m, \ldots, P_{N_w}^m\}, \tag{1}$$

where P_i^m represents the i-th waypoint of the m-th UAV, with the definitions $P_1^m = x_m^s$ and $P_{N_w}^m = x_m^g$.

Our core objective is to determine a set of paths $P = \{P^1, P^2, \ldots, P^M\}$ such that the total cost of the paths in the set is minimized. It is important to note that we also penalize various constraints in the form of costs; when a constraint is not satisfied, we penalize it with a cost of infinity.

2.1. Path Cost

We divide the flight path of the UAV into multiple nodes and sum the lengths of the flight path by calculating the distances between the nodes. The formula for calculating the path cost of the UAV is as follows:

$$F_1^m = \sum_{i=1}^{n-1} ||\overrightarrow{P_i^m P_{i+1}^m}|| \tag{2}$$

$$||\overrightarrow{P_i^m P_{i+1}^m}|| = \sqrt{(x_{i+1}^m - x_i^m)^2 + (y_{i+1}^m - y_i^m)^2 + (z_{i+1}^m - z_i^m)^2}, \tag{3}$$

where $\overrightarrow{P_i^m P_{i+1}^m}$represents the vectors between two nodes, $(x_{k+1}^m, y_{k+1}^m, z_{k+1}^m)$ denotes the coordinates of the m-th UAV node, and n is the number of nodes in the path.

2.2. Threat Cost

During flight, a UAV must account for the hazards presented by obstacles to ensure flight safety. In the analysis of flight constraints, obstacles are commonly modeled as cylinders, and the hazard risk escalates with decreasing distance to the cylinder's center. The threat cost for a UAV can be calculated using the following formula:

$$F_2^m = \sum_{i=1}^{N_w-1} \sum_{j=1}^{J} Tr_j(\overrightarrow{P_i^m P_{i+1}^m}), \tag{4}$$

where Tr_j represents the threat cost of the j-th obstacle to the path segment $\overrightarrow{P_i^m P_{i+1}^m}$. The calculation method is based on the relative distance d_j between the UAV and the obstacle as follows:

$$Tr_j(\overrightarrow{P_i^m P_{i+1}^m}) = \begin{cases} 0 & \text{if } d_j > S_1 + R_j \\ (S_1 + R_j) - d_j & \text{if } R_j < d_j \leq S_1 + R_j \\ \infty & \text{if } d_j \leq R_j, \end{cases} \tag{5}$$

where S_1 denotes the safety distance (the collision avoidance zone for the UAV), R_j is the cylindrical radius of the j-th obstacle, and J represents the total number of obstacles.

2.3. Altitude Constraint Cost

The altitude of a UAV during flight is typically bounded by a lower and upper limit. The formula for calculating the altitude cost is as follows:

$$F_3^m = \sum_{i=1}^{n} H_i^m \tag{6}$$

$$H_i^m = \begin{cases} |h_i - \frac{h_{\max}+h_{\min}}{2}|, & \text{if } h_{\min} \le h_i \le h_{\max} \\ \infty, & \text{otherwise,} \end{cases} \tag{7}$$

where $H_i$represents the height constraint cost, h_i is the altitude at node i of the path, and h_{max} and h_{min} are the maximum and minimum altitude constraints, respectively.

2.4. Path Smoothing and Turn Cost

To calculate the trajectory smoothing cost for a UAV, we need to determine the turning angles. The formula for the turning angles is as follows:

$$\phi_{ij}^m = \arctan\left(\frac{||\overrightarrow{P_i^m P_{i+1}^m} \times \overrightarrow{P_{i+1}^m P_{i+2}^m}||}{|\overrightarrow{P_i^m P_{i+1}^m} \cdot \overrightarrow{P_{i+1}^m P_{i+2}^m}|}\right). \tag{8}$$

The formula for the climb angle of the UAV is:

$$\varphi_{ij}^m = \arctan\left(\frac{z_{i+1} - z_i}{||\overrightarrow{P_i^m P_{i+1}^m}||}\right). \tag{9}$$

Finally, the formula for the smoothing cost is:

$$F_4^m = a_1 \sum_{j=1}^{n-2} \phi_{ij}^m + a_2 \sum_{j=1}^{n-1} |\varphi_{ij}^m - \varphi_{i,j-1}^m|, \tag{10}$$

where a_1 and a_2 are penalty coefficients for the turning angle and climb angle, respectively.

2.5. Collision Constraint Cost between UAVs

Let $d(m_1, m_2)$ represent the distance between UAVs m_1 and m_2, and let S_2 denote the safe distance between UAVs. Then, the collision constraint is as follows:

$$F_5 = \sum_{m_1=1}^{M} F_5^{m_1} \tag{11}$$

$$F_5^{m_1} = \begin{cases} 0, & \text{if } d_{\min}(m_1, m_2) > S_2, \forall m_2 \in \{1, \ldots, M\} \\ \infty, & \text{otherwise.} \end{cases} \tag{12}$$

2.6. Total Cost

Converting the cost constraints of UAVs into a cost function allows us to quantify the quality of their paths based on the value of this function. The cost function is as follows:

$$F = \sum_{c=1}^{5} b_c F_c, \tag{13}$$

where b_c represents the weight coefficient of the cost for the c-th constraint, and F denotes the total path cost for multi-UAVs.

3. Preliminary Knowledge of the PPSwarm Algorithm

A heuristic Prioritized Planning method, based on the PPSwarm algorithm, employs RRT* to generate the initial population of the Particle Swarm Optimization (PSO) algorithm, thereby enhancing the search efficiency of finding feasible solutions. This method leverages the Prioritized Planning algorithm to assign priorities to individual UAVs based on the mission context and environmental factors. The PSO algorithm is utilized to complete the path planning for individual UAVs with specified priorities. At the same time, low-priority UAVs are programmed to avoid collisions with high-priority UAVs, effectively solving potential collision issues among UAVs.

3.1. Particle Swarm Optimization (PSO)

The Particle Swarm Optimization (PSO) algorithm is an intelligent optimization technique that simulates the behavior of a natural biological swarm, representing an efficient search strategy. In PSO, each "particle" signifies a potential solution in the search space, updating its velocity and position based on the historical best positions of both the individual and the swarm. Specifically, every particle keeps track of its own historical best position (individual optimal solution, denoted as *pbest*) and the historical best position of the entire swarm (global optimal solution, denoted as *gbest*). The velocity and position updates of the particles are based on these two optimal solutions, ensuring that the search process moves in a better direction. The updated formulas are as follows:

$$\begin{aligned} v_i^{k+1} &= wv_i^k + c_1 r_1 (pbest_i^k - x_i^k) + c_2 r_2 (gbest^k - x_i^k) \\ x_i^{k+1} &= x_i^k + v_i^{k+1}, \end{aligned} \tag{14}$$

where $x_i \in [x_{\min}, x_{\max}]$ and $v_i \in [v_{\min}, v_{\max}]$ represent the position and velocity of the i-th particle, respectively. The parameters c_1 and c_2 are acceleration coefficients, which influence the degree to which individual and social experiences affect the movement of particles. The variables r_1 and r_2 are uniformly distributed random numbers in the range of $[0, 1]$. The parameter w is the inertia weight, which is used to balance the local and global search capabilities of the particles. The variable $pbest_i^k$ denotes the individual optimal position of the i-th particle at iteration k, while $gbest^k$ represents the global optimal position of the entire particle swarm at iteration k.

3.2. Rapidly-Exploring Random Tree Star (RRT)*

The Rapidly-exploring Random Tree (RRT) algorithm achieves fast exploration of non-convex high-dimensional spaces by randomly constructing a space-filling tree. This algorithm demonstrates its efficient search capability in complex spaces, as shown in Algorithm 1. After initializing the start and goal points, the algorithm begins by randomly selecting a sample point V_{random} from the given set E. Next, the algorithm searches within the constructed tree for the node closest to V_{random}, referred to as V_{nearest}. The algorithm then proceeds in the direction from V_{random} to V_{nearest} by a predefined step size, yielding a new node V_{new}. After generating V_{new}, the algorithm performs a collision check to ensure that the new node is not located within an obstacle area. If the collision check is passed, V_{nearest} is set as the parent node of V_{new}.

Subsequently, the algorithm rediscovers all nodes that could potentially be the parent of V_{new} within the range centered at V_{random}, with the predefined step size δ_{rrt} as the radius. If a node exists that is a better parent than the current one, meaning that the path to this node is shorter than the path to the current parent node, then the parent of V_{new} will be changed.

Then, RRT* undergoes a process called rewiring. During this process, the algorithm checks all nodes within the *radius* range. If an indirect path through V_{new} is shorter than the path to the current parent node, then the parent of that node is set to V_{new}.

Algorithm 1 RRT* for Single UAV m

Input: Environment E, Task T_m with start T_m^s and goal T_m^g
Output: Path P^m

$tree \leftarrow \{T_m^s\}$; $P \leftarrow \emptyset$
for $i \leftarrow 1$ to $Iterations$ **do**
  $x_{\text{random}} \leftarrow \text{Randsample}(E)$
  $x_{\text{nearest}} \leftarrow \text{Nearest}(tree, x_{\text{random}}, E)$
  $x_{\text{new}} \leftarrow \text{Steer}(x_{\text{nearest}}, x_{\text{random}}, stepsize)$
  if $\text{IsCollisionFree}(x_{\text{nearest}}, x_{\text{new}}, E)$ **then**
    $x_{\text{near}} \leftarrow \text{Near}(tree, x_{\text{new}}, E)$
    $x_{\text{min}} \leftarrow \text{ChooseParent}(x_{\text{near}}, x_{\text{nearest}}, x_{\text{new}})$
    $tree \leftarrow \text{InsertNodeAndRewire}(tree, x_{\text{min}}, x_{\text{new}}, x_{\text{near}})$
    if $\text{InGoalRegion}(x_{\text{new}}, T_m^g)$ **then**
      $P^m \leftarrow \text{ExtractPathToGoal}(tree, T_m^g)$
      break ▷ Success
    end if
  end if
end for

3.3. Prioritized Planning (PP) Method

The Prioritized Planning (PP) method performs path planning according to the priority among UAVs. When generating the UAV paths, the paths of higher priority are set as obstacles to avoid collisions among UAVs.

The framework of the classical PP is shown in Algorithm 2. UAVs are assigned different priorities, and path planning is conducted in descending order of priority, starting with the UAV with the highest priority. When planning the task T_m for the m-th UAV, the path P^m is calculated using the function $\text{SingleUAVPlan}(E, P, T_m)$. This function involves not only avoiding threats in the environment E but also the consideration of potential collisions with the preceding $m-1$ UAVs that have higher priority.

Algorithm 2 Classical Prioritized Planning Method

Input: Environment E, Task T with M UAVs
Output: Solution P

$P \leftarrow \emptyset$
for $m \leftarrow 1$ to M **do**
  $P_m \leftarrow \text{SingleUAVPlan}(E, P, T_m)$
  if $P_m = \emptyset$ **then**
    return
  end if
  $P \leftarrow P \cup P_m$
end for

4. Proposed Method

4.1. Algorithm Framework

Aiming at the complexity of the multi-UAV path problem, the PPSwarm algorithm employs a two-level path planning strategy to effectively decouple the problem and reduce its difficulty. The overall structure of this strategy consists of two main parts: high-level strategy and low-level strategy. The main process is illustrated in Figure 1.

4.1.1. High-Level Strategy

The high-level strategy first initializes UAV paths using the RRT* algorithm, providing an initial reference for subsequent planning. Then, priority assignment is performed, and a restart strategy is introduced to adapt to complex environments and avoid local optima. After each path planning iteration, priorities are reassigned, and planning is repeated to

find a better solution. The UAV with the highest priority enters the low level for more precise path planning.

Algorithm 3 describes the steps of the high-level strategy in detail. Firstly, the RRT* algorithm is used to initialize a path for each UAV (lines 3–5). Next, priorities are assigned based on path cost, with higher costs receiving higher priorities (line 6). In subsequent iterations, priorities are randomly assigned to increase search diversity (line 16). During each restart, the obstacle list is cleared (line 9), and then paths are optimized using PSO according to priority, with the obstacle list updated to avoid conflicts (lines 11–14). The restart strategy randomly reassigns priorities and repeats path planning until the preset number of restarts is reached (lines 8–17). Finally, the algorithm outputs a path planning solution that can adapt to the complex environment optimization and ensure the efficient and safe operation of the UAV.

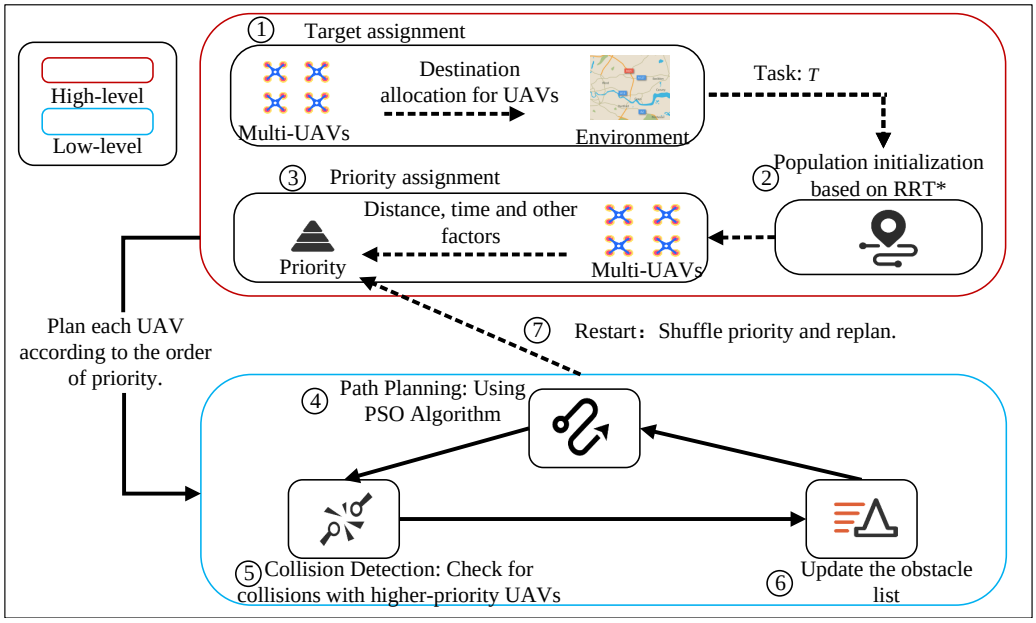


Figure 1. PPSwarm algorithm flow.

Algorithm 3 High-Level Search

Input: Number of UAVs M, number of restarts N_{restart}, environment E, task T, dynamic obstacle list $obst$

Output: Solution P

1: **for** $m \leftarrow 1$ **to** M **do**
2: $P^m \leftarrow \text{RRT}^*(E, T_m)$ ▷ Algorithm 1
3: **end for**
4: $Priority \leftarrow \text{SortbyCost}(P)$
5: **for** $i_{\text{restart}} \leftarrow 1$ **to** N_{restart} **do**
6: $obst \leftarrow \emptyset$
7: **for** $index \leftarrow 1$ **to** M **do**
8: $m \leftarrow Priority[index]$
9: $P^m \leftarrow \text{PSO}(P^m, T_m, E, obst)$ ▷ Algorithm 4
10: $D \leftarrow \text{DiscretizePath}(P^m, v_{\text{uav}})$ ▷ Algorithm 5
11: $obst \leftarrow obst \cup D$
12: **end for**
13: Shuffle $Priority$
14: **end for**

4.1.2. Low-Level Strategy

At the low level, a series of operations are performed during each iteration cycle, including path planning, collision detection, and updating the obstacle list. Path planning is achieved by inheriting the paths initialized by the RRT* algorithm and the previous generation of "restart" particle swarms, and then optimizing them using the PSO algorithm. The proposed method effectively integrates the advantages of both the RRT* and PSO algorithms. It leverages the excellent global search capability of the RRT* algorithm to identify potential locations of the overall optimal solution, while leveraging the powerful local optimization capabilities of the PSO algorithm to fine-tune the solutions. This fusion results in efficient and high-quality path planning solutions. During the path planning process, the low level performs collision detection against the global obstacle list to ensure that the generated paths are safe and reliable. Once the path planning is complete, the optimized path is updated in the global obstacle list. This allows for the efficient utilization of this information in subsequent planning processes, thereby improving the efficiency of both path planning and collision detection.

4.1.3. Encoding Selection

The choice of encoding is crucial for evolutionary algorithms. In the PSO algorithm, the encoding of particles is represented by multiple points on a path. In existing evolutionary algorithms, these path points can usually be represented as three-dimensional coordinates in Cartesian coordinates [10], polar coordinates [35], and SpherePSO [36]. These codes can better improve the convergence speed of PSO. In this article, we used RRT* to find the initial path and then randomized it. Using Cartesian coordinates to represent path points enables better randomization operations. Therefore, we chose Cartesian coordinates.

4.2. Single-UAV Path Planning Based on Particle Swarm Optimization

In our approach, Particle Swarm Optimization (PSO) is used to solve the path for a single UAV, where each particle represents a potential solution for the UAV. To fully utilize the results obtained from the RRT* algorithm and before the restart, we have decided to implement a strategy to accelerate the convergence speed of the particles.

4.2.1. Population Initialization Based on RRT*

When considering the initialization of UAV paths, the RRT* algorithm is capable of generating preliminary flight paths for individual UAVs. This algorithm explores and finds a feasible path by constructing a tree structure that starts from the starting point and gradually expands toward the target area. RRT* not only focuses on the rapid generation of paths but also optimizes the quality of the paths by rewiring existing path segments to ensure that the generated paths are relatively optimal.

Using the RRT* algorithm for UAV path initialization can provide a preliminary and effective path reference for subsequent priority allocation. Furthermore, assigning the initialized path obtained from the RRT* algorithm to the particle swarm for subsequent path planning leverages the advantages of RRT* in path exploration. This approach enhances the search efficiency and quality of the PSO algorithm while reducing the risk of getting trapped in local optima. This method exhibits significant flexibility and applicability when dealing with complex environments and multiple constraints.

For UAV m, after obtaining a feasible solution P^m using the RRT* algorithm, we assign this path to the first particle, and the other particles are randomized around this path. Specifically, the initialization formula for the particle swarm is as follows:

$$x_i = \begin{cases} P^m, & \text{if} \quad i = 1 \\ (1-\alpha) * P^m + \alpha * \text{randpath}(T_m), & \text{if} \quad i > 1 \end{cases}. \tag{15}$$

Figure 2 provides an illustration of the population initialization based on RRT*. In the image, the blue represents the search tree randomly generated by RRT*, the thick red line

indicates the obtained shortest path, and the thin red lines are the initial population paths sampled randomly around the shortest path 100 times.

4.2.2. Diversification of the Population after Restart

To better adapt to complex environments and avoid UAV paths falling into local optima, we employ a restart strategy (Algorithm 3) in UAV priority assignment, with different priorities assigned in each restart. After each restart iteration, to fully utilize the results of the previous solution, we retain the states of some particles from the previous iteration to facilitate rapid convergence. On the other hand, to maintain particle diversity, the algorithm also introduces some random particles on this basis. These random particles can increase the exploration space and help the algorithm escape from local optimal solutions. We use a parameter ρ to control the proportion of random paths in the population, which is formalized as follows:

$$x_i = \begin{cases} \text{randpath}(T_m), & \text{if } i \leq \rho \cdot N_{pop} \\ x_i^{last}, & \text{if } i > \rho \cdot N_{pop} \end{cases}, \tag{16}$$

where randpath represents a random path, x_i denotes the initialization path of particle i after a restart, and x_i^{last} refers to the flight path of particle i before the restart.

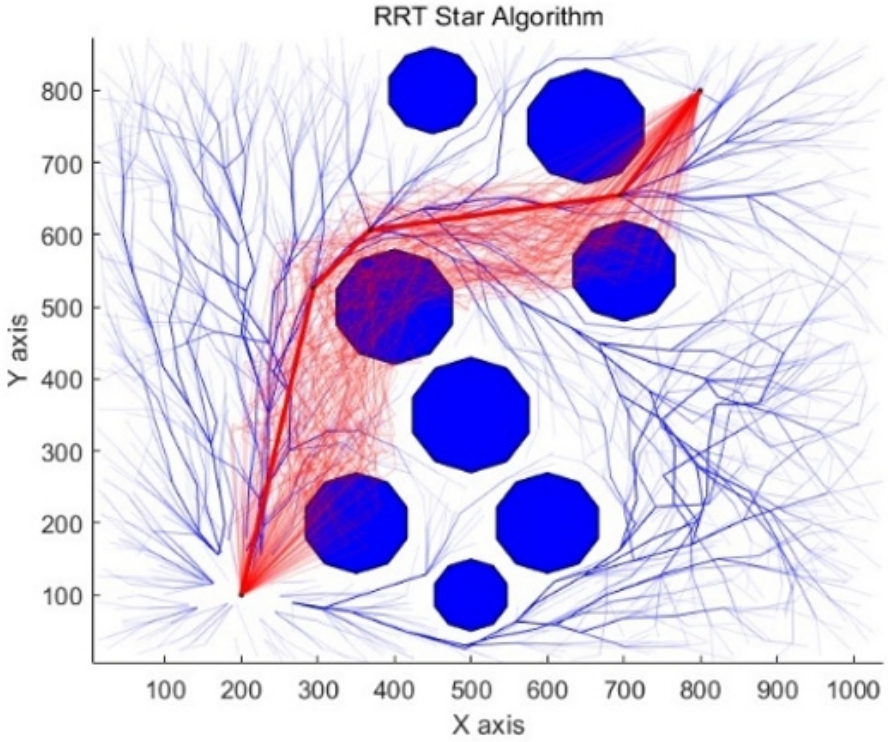


Figure 2. Schematic diagram of population initialization based on RRT*.

4.2.3. PSO Pseudocode

Algorithm 4 describes in detail the process of particle swarm optimization for UAV m in a dynamic environment. The algorithm first initializes the position and velocity of each particle based on previous solutions and task requirements. It then calculates the fitness of each position, considering both environmental conditions and the presence of dynamic obstacles. Through a series of iterations, the optimal positions (personal best and global best) are continuously updated. The state of each particle (position and velocity) is updated based on environmental interactions, and fitness evaluation guides the optimization process. Finally, the algorithm selects the global best position as the solution path for the UAV, effectively adapting to the dynamic conditions in the environment and ensuring optimized task execution under changing conditions. This PSO variant is

specifically designed to handle dynamic obstacles and adjust agent paths accordingly to achieve optimal task execution.

Algorithm 4 PSO for UAV m

Input: Iterations N_{iter}, environment E, dynamic obstacle list $obst$, last solution P^m, task T_m, restart count $i_{restart}$
Output: Solution P^m

Initialize the position of each particle i based on P^m and T_m:

$$x_i^0 \leftarrow \begin{cases} \text{Using Equation (15)}, & \text{if } i_{restart} = 1 \\ \text{Using Equation (16)}, & \text{otherwise} \end{cases}$$

Initialize the velocity v_i^0 for each particle i
Compute fitness $F(x_i^0)$ for each particle i considering E and $obst$
Set initial personal best $pbest_i \leftarrow x_i^0$ for each particle i
$gbest \leftarrow \arg\min_{x_i^0} F(x_i^0)$
for $k = 1$ to N_{iter} **do**
    for each particle i **do**
        $(x_i^k, v_i^k) \leftarrow \text{UpdateState}(x_i^{k-1}, v_i^{k-1}, E)$ ▷ Equation (14)
        Evaluate fitness $F(x_i^k)$ considering E and $obst$
        if $F(x_i^k) < F(pbest_i)$ **then**
            $pbest_i \leftarrow x_i^k$
            if $F(x_i^k) < F(gbest)$ **then**
                $gbest \leftarrow x_i^k$
            end if
        end if
    end for
end for
$P^m \leftarrow gbest$ ▷ Assign global best to the current path
return P^m

4.3. Lookup-Based Fast Collision Detection

When performing path planning in the Particle Swarm Optimization algorithm, if a path planned by a particle collides with a UAV of higher priority, we will adopt a specific penalty mechanism. Specifically, once such a collision is detected, we set the fitness value of that particle to infinity. By assigning an extremely high fitness value, we can ensure that the particle is eliminated in the next iteration, thereby preventing its path from being selected as the flight path for the UAV. Collision detection calculations between UAVs can be a time-consuming process. To address this issue, we use a lookup list to record the paths of UAVs with priority, thereby accelerating the processing of collision detection.

Dynamic Obstacle List and Path Discretization

Before conducting collision detection, we introduce a dynamic obstacle list (Algorithm 3), which is sorted by priority and stores the timeline of each UAV's optimal path. Specifically, when we need to determine whether a particle's path collides with other UAVs, we compare the particle's path schedule with the dynamic obstacle list. To facilitate this comparison, we discretize the paths at uniform time intervals, enabling efficient collision detection through table lookup in subsequent steps. During the PSO iteration process, we compare each discretized path point of the particle with the paths in the dynamic obstacle list. If the distance between two path points is less than the preset safe distance, we consider it a collision.

Algorithm 5 provides a method for path discretization. It accepts a path P_m and the UAV's flight speed v_{uav} as inputs and outputs a discretized path D. The algorithm iterates

through each pair of adjacent points on the path, calculates the distance between them, and determines the number of intermediate points to be inserted based on the UAV's speed. It then uses linear interpolation to compute the positions of these intermediate points and ultimately adds them to the discretized path D.

Algorithm 5 Discretize Path

Input: Path $P^m = \{P_1^m, \ldots, P_{N_w}^m\}$, Speed v_{uav}
Output: Discretized path D

Initialize: $D \leftarrow \emptyset$, $i \leftarrow 1$
while $i < N_w$ **do**
  $d \leftarrow ||P_{i+1}^m - P_i^m||$ ▷ Distance between points
  $n \leftarrow \lceil d / v_{\text{uav}} \rceil$ ▷ Number of steps
  for $j \leftarrow 1$ to n **do**
    $t \leftarrow j/n$ ▷ Interpolation factor
    $P_{\text{interp}} \leftarrow (1-t)P_i^m + tP_{i+1}^m$ ▷ Interpolate
    $D \leftarrow D \cup \{P_{\text{interp}}\}$ ▷ Add to discretized path
  end for
  $i \leftarrow i + 1$
end while
return D

4.4. Path Smoothing Based on Dubins

As the flight paths generated by the algorithm may not meet the actual flight requirements of multi-UAVs, this paper adopts Dubins curves for smoothing processing to obtain the actual flight paths. Dubins curves were proposed by Dubins in 1957 [37], proving the existence of the shortest path within the set of these curves.

Dubins curves satisfy kinematic constraints through combinations of arcs with maximum curvature (C) and straight line segments (S) [38]. Under the maximum curvature constraint, the shortest feasible path between two directed points in a plane is either a CSC path or a CCC path, or a subset of them. Here, C represents a circular arc segment, and L represents a straight line segment tangent to C. We only consider the CSC-type path as shown in Figure 3. The entire Dubins set comprises six types of paths, namely *LRL*, *LSR*, *LSL*, *RLR*, *RSL*, and *RSR*. Here, *S* represents an arc path segment in the counterclockwise direction, while *R* denotes an arc path segment in the clockwise direction. During trajectory following, it is essential to avoid paths with high curvature to prevent excessive lateral deviations while following the path. Therefore, when calculating the path, it is only necessary to solve for the four types of curves—*LSL*, *LSR*, *RSL*, and *RSR*—and then select the path that satisfies the constraints as the optimal solution, where r_∂ is defined as the turning radius.

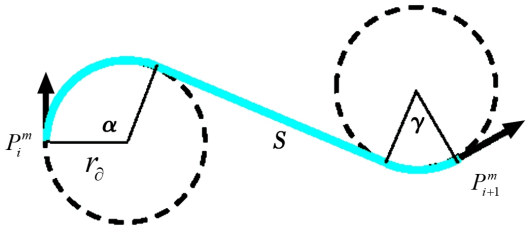


Figure 3. Dubins path of the CSC type.

After Dubins smoothing, the resulting path meets the performance constraints of the UAV and can be used for UAV flight.

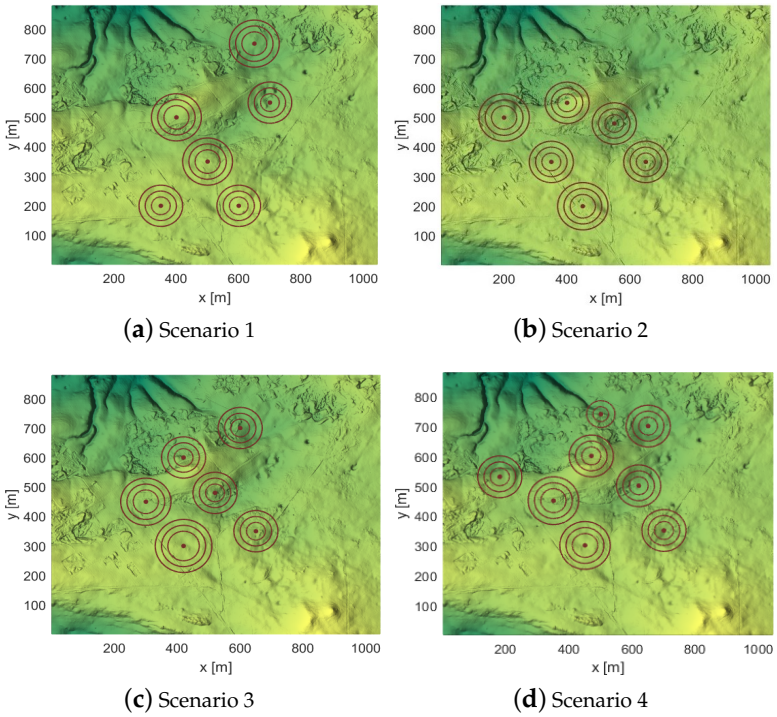
5. Experiment

This section introduces the experimental results of the PPSwarm algorithm. Firstly, the relevant content of the experimental setup is introduced. Then, a comparative exper-

iment of existing algorithms is conducted to evaluate the quality and scalability of the proposed algorithm. Finally, the priority allocation scheme and algorithm parameters in the algorithm are analyzed, and a visual display of the algorithm is presented.

5.1. Experiment Setup

Based on the above algorithm design, simulation experiments are carried out on Lenovo laptop with Intel 2.5 GHz processor and 8 GB of RAM, implemented in the MATLAB R2019a environment. The experiments utilized a map of dimensions 1045 m × 875 m × 200 m, and the experimental scenarios were generated based on a three-dimensional terrain model derived from a digital elevation model (DEM). DEM uses a finite set of ground elevation data to digitally simulate terrain. To facilitate the establishment of the model, obstacles are considered as cylindrical bodies with their center at C_k and radius R_k. It is also assumed that the position information of the obstacles is known when setting up the scenario. When the center coordinates, radius, and height of the obstacles are known, they can be represented on the generated 3D map. A total of four scenarios were set up for this experiment, as shown in Figure 4. The speed of the UAV, v_{uav}, is set at 20 m/s. The turning radius of Dubins is 20 m. The safe distance S_1 between the UAV and the threat edge is 15 m. The safe distance S_2 between UAVs is 10 m.



(**a**) Scenario 1 (**b**) Scenario 2

(**c**) Scenario 3 (**d**) Scenario 4

Figure 4. Top view in four scenarios.

In the absence of a specific designation, the parameters related to the algorithms in this paper are as shown in Table 1. Among these, the settings for the maximum number of iterations N_{iter}, number of restarts $N_{restart}$, population size N_{pop}, and the random ratio of the population ρ are determined based on the experimental analysis in Section 5.3.3. To balance the computational time of the algorithm and the quality of the solution, the number of restarts $N_{restart}$ is set to 5. Other parameters are set according to reference [36].

Table 1. PPSwarm algorithm parameters.

Parameter	Setting
Maximum number of iterations	N_{iter} = 20
Number of restarts	$N_{restart}$ = 5
Population size	N_{pop} = 300
Random ratio of the population	ρ = 20%
Cognitive scaling parameter	$c1$ = 1.5
Social scaling parameter	$c2$ = 1.5
Inertia weight	w = 1

5.2. Comparative Experiment with Existing Algorithms

5.2.1. Runtime and Quality

The comparative experiment of the algorithm pits it against five other algorithms: PSO [10–12], DE [6,7], ABC [15], GWO [39], and SPSO [36], across four scenarios. All five comparison algorithms adopt the LH allocation scheme of prioritized planning to address UAV collision issues. To better test the performance of the algorithms, the number of UAVs was increased to 40, and each algorithm was run 20 times. The relevant parameters of other algorithms are set according to the above references, as shown in Table 2.

Table 2. Comparison algorithm parameter settings.

Algorithm	Parameters
DE	N_{iter} = 100, N_{pop} = 300, F = 0.8, CR = 0.9, $refresh$ = 10
PSO	N_{iter} = 100, N_{pop} = 300, w = 0.8, $c1$ = 1.45, $c2$ = 1.5
ABC	N_{iter} = 100, N_{pop} = 300, $FoodNumber$ = 150
GWO	N_{iter} = 100, N_{pop} = 300
SPSO	N_{iter} = 100, N_{pop} = 300, w = 1, $\eta 1$ = 1.5, $\eta 2$ = 1.5

Figure 5 shows the convergence curve of PPSwarm and the comparison algorithms when the restart strategy is not adopted. It can be observed that, without employing the restart strategy, the proposed algorithm outperforms the other five algorithms in terms of convergence accuracy across all four scenarios. In Figure 5a, the PPSwarm algorithm exhibits the phenomenon of converging too quickly when the restart strategy is not used. Figure 6 presents the box plots of the best cost for 20 independent iterations of the six algorithms, with the statistical results shown in Table 3. Compared with Figures 5a and 6a, the best value of PPSwarm becomes smaller after adopting the restart strategy, indicating that this strategy makes the algorithm jump out of the local optimal and solves the local optimal caused by converging too fast. From the figures, it is apparent that, in large-scale multi-UAV and complex environments, the SPSO, PSO, and GWO algorithms are prone to falling into local optima due to premature convergence. Although the DE algorithm has a strong global search capability, the setting of its parameters significantly impacts its performance, resulting in a lower convergence accuracy compared to PPSwarm. The ABC algorithm is overly dependent on the initial solution, leading to slow convergence. The PSO and SPSO algorithms exhibit the poorest convergence accuracy compared to other algorithms. Among the six algorithms, the PPSwarm algorithm outperforms the others in terms of optimal value, minimum standard deviation, and average value, as well as runtime. The lower standard deviation confirms the stability of the PPSwarm algorithm, while the reduced runtime and lower path cost demonstrate its efficiency.

The results demonstrate that the proposed algorithm is able to meet the requirements of the path planning cost function, quickly generating a reasonable flight route while satisfying the obstacle avoidance prerequisites. The algorithm surpasses other algorithms in terms of convergence speed, convergence accuracy, and running time. It is evident that this algorithm outperforms the other five algorithms.

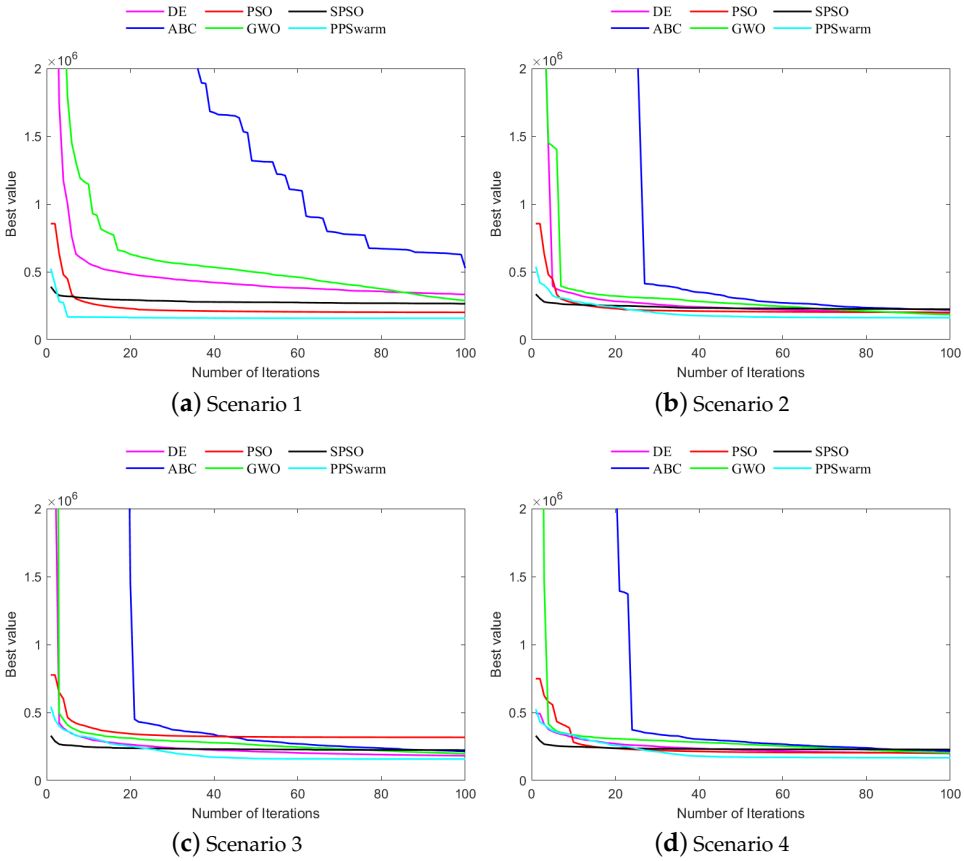


Figure 5. The convergence curves of the six algorithms.

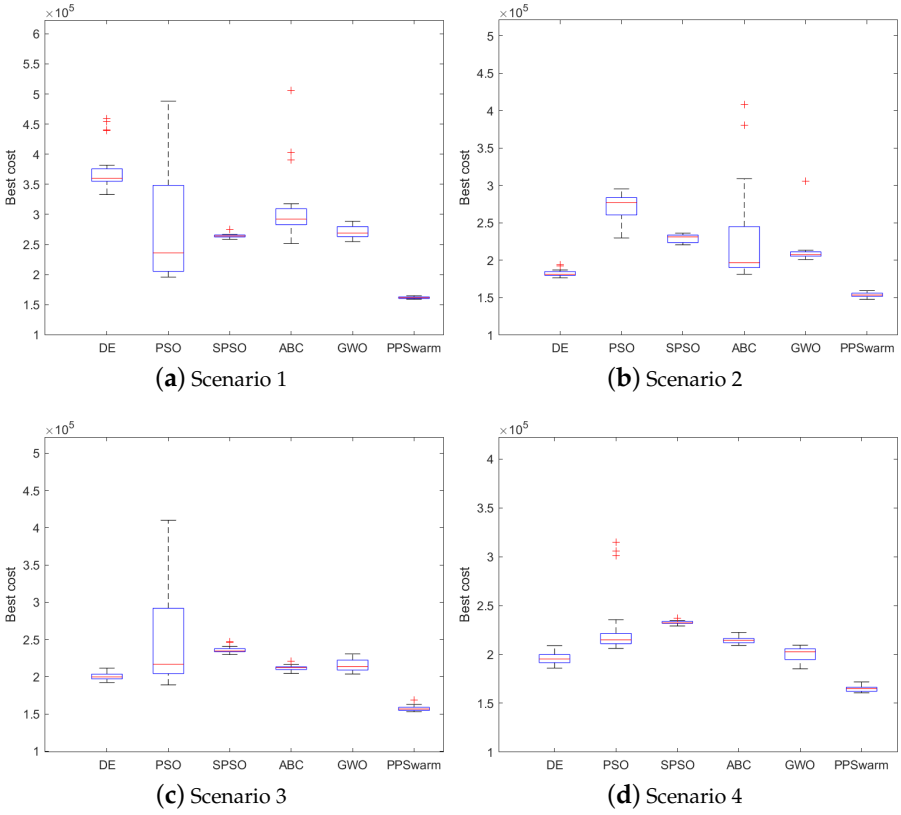


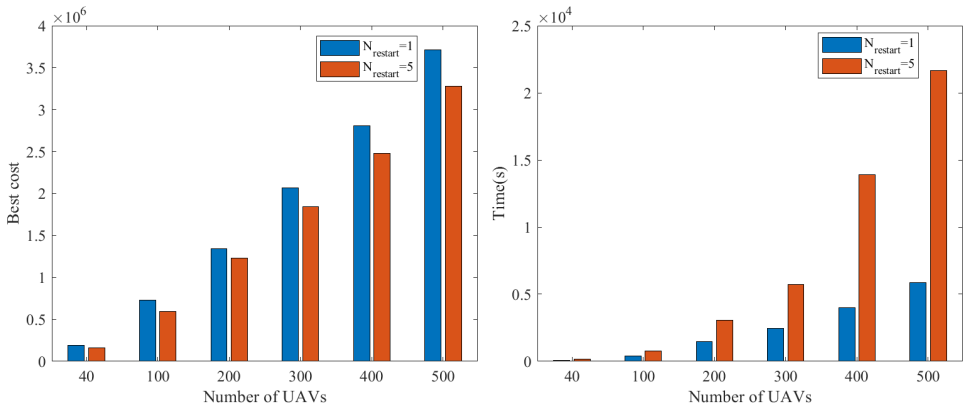
Figure 6. Box plots of the six algorithms after 20 iterations.

Table 3. Comparative results of the six algorithms in terms of cost and runtime. The best results achieved among all algorithms are shown in bold.

Scenario		PSO	DE	ABC	GWO	SPSO	PPSwarm
Scenario 1	Best Cost	195,927	332,830	251,589	254,721	258,545	**159,023**
	Worst Cost	488,132	458,830	506,118	288,642	274,688	**164,938**
	Mean Cost	275,781	373,006	311,405	270,288	264,363	**161,649**
	Std Cost	889,871	35,699	58,226	9868	3212	**1681**
	Runtime (s)	337.6	330.3	261.8	325.6	1405.4	**149.9**
Scenario 2	Best Cost	229,738	176,380	181,202	200,904	220,546	**147,572**
	Worst Cost	295,353	194,223	408,479	306,069	2,361,814	**159,212**
	Mean Cost	272,361	182,666	228,962	212,742	229,682	**153,744**
	Std Cost	16,250	4576	67,392	22,216	5329	**3171**
	Runtime (s)	348.2	412.4	280.8	325.6	2425.2	**111.7**
Scenario 3	Best Cost	189,039	192,234	204,343	203,650	229,676	**152,923**
	Worst Cost	409,891	211,615	220,851	230,760	246,710	**168,341**
	Mean Cost	244,827	200,284	211,773	215,424	229,676	**157,357**
	Std Cost	59,342	5104	**3719**	7815	235,959	4023
	Runtime (s)	208.5	327.7	218.4	385.9	1811.6	**111.8**
Scenario 4	Best Cost	206,189	185,995	209,277	185,229	229,093	**160,720**
	Worst Cost	314,960	209,045	222,588	209,501	236,863	**171,798**
	Mean Cost	228,587	195,390	214,462	200,212	232,768	**165,053**
	Std Cost	34,610	5742	3392	6577	**1736**	3123
	Runtime (s)	217.5	194.9	238.7	189.7	1875.4	**110.5**

5.2.2. Scalability

Scalability analysis is performed for Scenario 1. When the number of unmanned aerial vehicles (UAVs) reaches 100, other comparative algorithms are no longer able to obtain UAV flight paths that satisfy the constraints. To test the performance of the proposed algorithm, we compared the cases with $N_{restart} = 1$ and $N_{restart} = 5$. The comparison results are shown in Figure 7a, where the fitness value of $N_{restart} = 5$ is significantly lower than that of $N_{restart} = 1$. The fitness values of both exhibit approximately a linear increase, and the gap between the fitness values becomes larger as the number of UAVs increases. Figure 7b represents the runtime of the algorithm. The experimental results indicate that the algorithm not only meets the flight requirements of a large number of UAVs but also demonstrates excellent scalability. The results of the scalability analysis are shown in Table 4.



(**a**) Fitness of $N_{start} = 1$ and $N_{start} = 5$ (**b**) Runtimes of $N_{start} = 1$ and $N_{start} = 5$

Figure 7. Scalability analysis.

Table 4. Results of scalability analysis.

N_{start} \ Number		40	100	200	300	400	500
N_{start} = 1	Best Cost	189,604	727,884	1,341,113	2,069,532	2,807,932	3,715,660
	Time (s)	81.2	419.6	1468.6	2458.2	3981.7	5847.5
N_{start} = 5	Best Cost	161,649	597,725	1,228,034	1,840,845	2,479,620	3,275,223
	Time (s)	149.9	778.3	3070.4	5744.7	13,898.4	21,636

5.3. Algorithm Analysis

5.3.1. Analysis of Priority Allocation Schemes

To verify that the allocation scheme of the Priority Planning method can effectively adapt to large-scale complex terrain and obstacle environments, we conducted an analysis of the priority allocation scheme. We analyzed the allocation schemes of three baseline PP algorithms with 40 UAVs in four different scenarios: (1) LH (Longer Heuristic): This is a heuristic algorithm that assigns higher priority to agents with longer distances between their start and target positions on the graph. The intuition behind this algorithm is that agents with longer distances have a greater need for prioritized planning to ensure they reach their targets efficiently (van den Berg and Overmars 2005 [40]); (2) SH (Shorter Heuristic): This is another heuristic algorithm that gives higher priority to agents with shorter distances between their start and target positions. The rationale for this approach is that agents with shorter distances may require less time and fewer resources to complete their tasks, making them more suitable for prioritization (Ma et al., 2019 [41]); (3) RND (Random Heuristic): This heuristic algorithm randomly generates the overall priority ranking. It does not consider any specific criteria or heuristics but rather assigns priorities randomly. This serves as a baseline for comparing the performance of the other two heuristic algorithms (Bennewitz, Burgard, and Thrun 2002 [42]).

The experimental data for four scenarios are shown in Table 5. In terms of algorithm cost, the LH outperforms the other two allocation schemes, generating flight paths with lower costs. Regarding time consumption, LH is generally superior to the other two allocation methods and demonstrates better stability in terms of time compared to SH and RND. The RND allocation scheme exhibits the worst stability. In summary, using the LH allocation scheme for Prioritized Planning methods results in better and faster flight paths.

Table 5. Comparison of results from different priority allocation schemes. The best results achieved among all algorithms are shown in bold.

Scenario	Num	Cost ($\times 10^4$)			Average Time (s)		
		LH	SH	RND	LH	SH	RND
Scenario 1	10	**41.97**	46.01	43.60	7.171	**6.806**	6.888
	20	88.01	**87.64**	88.80	17.90	19.31	**17.79**
	30	**1.286**	1.291	1.305	**34.20**	44.33	45.79
	40	**7.556**	8.317	7.565	45.82	**44.33**	45.79
Scenario 2	10	**42.15**	42.81	45.87	**11.72**	18.25	17.70
	20	**85.33**	85.98	87.91	**20.26**	44.72	47.76
	30	**1.285**	1.295	1.295	36.17	35.53	**35.23**
	40	**1.969**	3.023	3.005	**56.45**	61.59	56.70
Scenario 3	10	48.15	48.43	**47.21**	**7.099**	7.221	7.168
	20	**89.88**	90.70	91.73	**18.96**	23.90	19.92
	30	1.380	1.412	**1.364**	38.60	41.80	**34.74**
	40	**2.919**	3.022	3.654	**56.21**	57.93	66.96
Scenario 4	10	**45.37**	46.01	45.47	**6.992**	7.452	7.858
	20	**87.25**	87.31	87.89	**21.93**	20.31	27.40
	30	**1.374**	1.394	1.377	70.19	**68.232**	79.83
	40	**3.421**	4.133	3.669	135.8	141.5	**75.20**

5.3.2. Low-Level Evolutionary Algorithm Analysis

Low-level updates to UAV paths utilize an evolutionary algorithm based on the standard PSO. On the foundation of the aforementioned algorithm, comparative experiments are conducted using PSO variants (CLPSO [43], SPSO [36], and ExPSO [44]) to explore the impact of these variants on the algorithm's performance. Ten UAVs are used in Scenario 1, with each experiment repeated 20 times to obtain an average value. The algorithm parameters for the PSO variants are set according to the parameters referenced in the aforementioned literature.

As shown in Figure 8, based on the PPSwarm algorithm, the use of a variant of the PSO algorithm in the low-level evolutionary algorithm has minimal impact on the results of the algorithm. Therefore, this paper adopts the standard PSO algorithm.

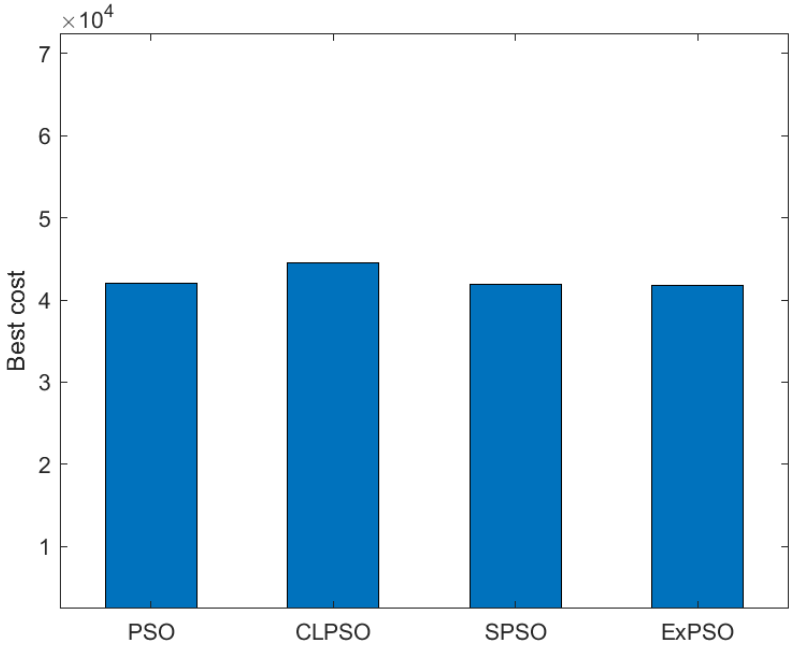
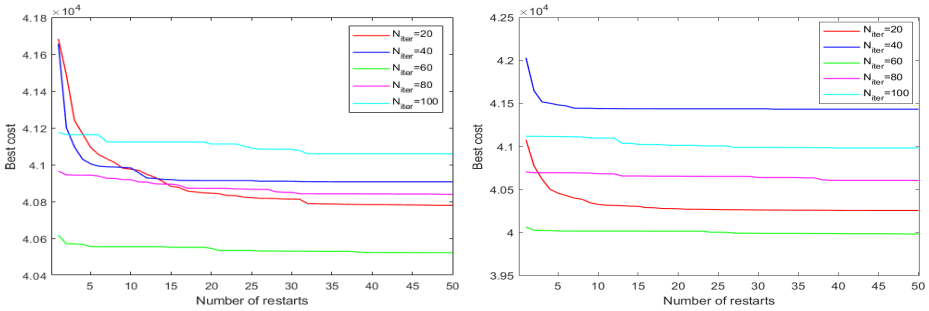


Figure 8. Evolutionary algorithm comparison.

5.3.3. Parameter Analysis

To ensure optimal algorithm performance through the coordination of various parameters, an analysis of the relevant parameters is conducted. In this section, the focus is on performing parameter analysis on the iteration count (N_{iter}), the number of restarts ($N_{restart}$), the size of the particle swarm (N_{pop}), and the number of random populations (ρ). Each set of experiments is conducted 20 times in Scenario 1, and the results are averaged.

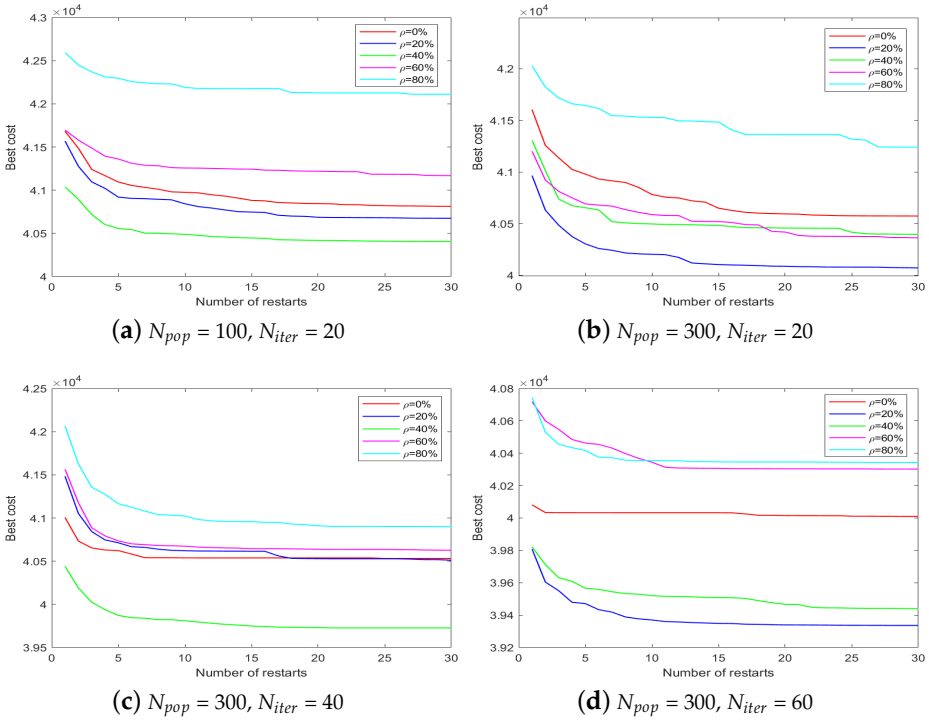
When the random population ρ is set to 0, the iteration behavior for populations with N_{pop} = 100 and N_{pop} = 300 is illustrated in Figure 9. It is evident that, once the number of iterations N_{iter} in the PSO algorithm exceeds 40, the algorithm tends to converge too quickly, which can result in the population becoming stuck in local optima. The iterations appear to fully converge around $N_{restart}$ = 10. However, particles with N_{iter} values between 20 and 40 possess the ability to break free from local optima. Upon comparison of the two images, populations N_{pop} = 300 with N_{iter} = 20 and N_{pop} = 300 with N_{iter} = 60 exhibit superior fitness values. Notably, N_{iter} = 20 demonstrates enhanced exploratory ability, which allows for a more effective escape from local optima.



(**a**) N_{pop} = 100 (**b**) N_{pop} = 300

Figure 9. N_{iter} parameter analysis.

To determine the optimal value of the random population size ρ for maximizing the algorithm's performance, experimental analyses were conducted on populations with different configurations: N_{pop} = 100, $N_{restart}$ = 30, N_{iter} = 20 and N_{pop} = 300, $N_{restart}$ = 30, with N_{iter} values of 20, 40, and 60, respectively. As shown in Figure 10, comparing subfigures (a) and (b), as ρ increases, a population size of 300 achieves a path with a lower fitness value compared to a population size of 100, indicating better performance for N_{pop} = 300. When comparing subfigures (b–d) collectively, it is evident that a ρ between 20% and 40% exhibits superior performance. This range allows for better evasion of local optima and results in the lowest fitness value for the flight path. Therefore, the introduction of ρ increases particle diversity, enabling particles to overcome local optima.



(**a**) N_{pop} = 100, N_{iter} = 20 (**b**) N_{pop} = 300, N_{iter} = 20

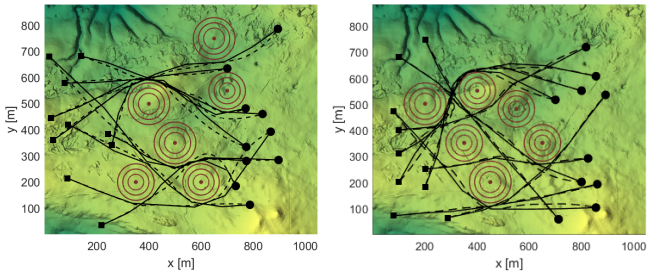
(**c**) N_{pop} = 300, N_{iter} = 40 (**d**) N_{pop} = 300, N_{iter} = 60

Figure 10. Parameter analysis for ρ.

From the aforementioned figures, it can be observed that almost all iterations converge within $N_{restart}$ = 25. Based on this, we can deduce the optimal parameter settings as follows: N_{pop} = 300, ρ ranging between 20% and 40%, N_{iter} values between 20 and 40, and $N_{restart}$ = 25, where the $N_{restart}$ parameter can be changed according to the situation.

5.3.4. Visual Display of Planned Paths

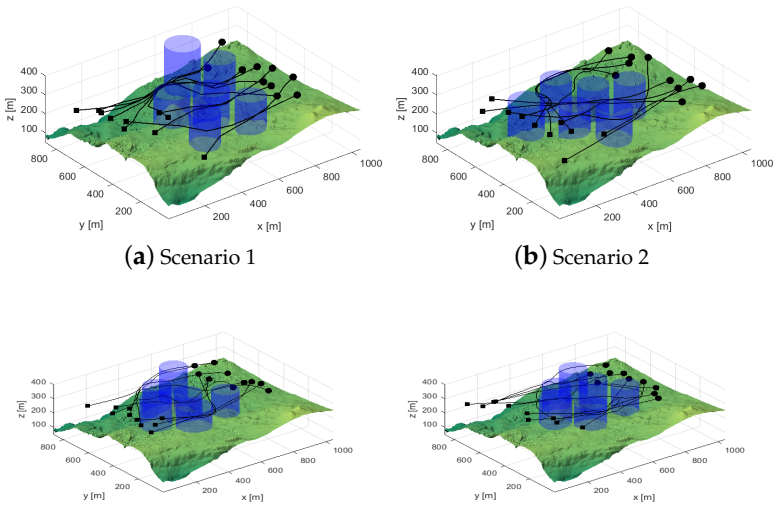
This article sets up six obstacles and selects 10 UAVs to plan the optimal path from the starting point to the destination. The relevant parameters of the algorithm used in this article are listed in Table 3. The path planning results of the proposed algorithm are shown in Figure 11, and Figure 12 presents the flight trajectories of the UAVs from different three-dimensional views. It can be seen that the planned trajectories of the 10 UAVs can meet the height and angle constraints while efficiently completing the flight mission under the collision constraints between obstacles and UAVs. In summary, this algorithm satisfies all the constraints required for all UAVs to complete their missions safely, quickly, and collision-free, making it able to perform flight path planning in complex environments with a large number of UAVs.



(**a**) Scenario 1 (**b**) Scenario 2

(**c**) Scenario 3 (**d**) Scenario 4

Figure 11. Flight path planned by the proposed algorithm. The solid line represents the theoretical path, and the dashed line represents the Dubins smoothed path.



(**a**) Scenario 1 (**b**) Scenario 2

(**c**) Scenario 3 (**d**) Scenario 4

Figure 12. Ten UAVs' 3D view.

6. Discussion

This study delves into the optimization of particle swarm algorithms and provides innovative ideas for further enhancing their performance. However, the methods proposed thus far primarily focus on scenarios involving cylindrical obstacles and fixed formation flights, which present certain limitations in practical applications. To more closely align with real-world demands, future work needs to be deepened and expanded in multiple directions. One direction that warrants immediate consideration is the collision issue with obstacles of different shapes. In the real world, obstacles often take many forms and may vary depending on the environment, time, or other factors. However, using only cylindrical obstacles does not facilitate the representation of real-world phenomena by swarm agents. Specifically, research in [45] indicates that using simple non-concave obstacles leads to poor results during swarm performance validation, and references [46,47] emphasize the importance of complex environments in UAV swarm testing. Another direction worthy of in-depth exploration is the performance of UAVs in dynamic mission scenarios. As mission requirements continuously evolve, UAV formations may need to adjust flexibly to accommodate new task demands. This necessitates an in-depth study of dynamic UAV formation control technologies, enabling UAVs to quickly and accurately adjust their formation and flight strategies based on mission needs. Through research in this area, we can significantly enhance the adaptability and operational capability of UAV formations in complex and variable environments.

In summary, while this study has achieved certain results in the optimization of particle swarm algorithms, further in-depth research is still needed in areas such as obstacle shape and adaptation to dynamic mission scenarios to realize the broader application and higher efficiency of UAV technology. In addition, for complex environments, the number of obstacles is also a potential research direction in the same field. These research directions will help promote technological advancements in the field of unmanned aerial vehicles, laying a solid foundation for future practical applications.

7. Conclusions

This paper proposes a heuristic algorithm based on PPSwarm to address the collaborative path planning problem for multiple unmanned aerial vehicles (UAVs) in complex three-dimensional environments. In the proposed algorithm, a reasonable multi-objective optimization cost function is designed, considering the performance of the UAVs and the constraints of the flight environment. Secondly, to decouple the multi-UAV path planning problem and reduce the difficulty of solving it, a two-level path planning strategy consisting of a high-level and a low-level planning strategy is proposed to hierarchically implement the PPSwarm algorithm. Specifically, by combining the exploration advantages of RRT*, the high-level planning strategy adopts the RRT* algorithm for path initialization. Meanwhile, a priority planning algorithm is utilized in this strategy to assign priorities to the UAVs, and the optimal UAV path obtained is incorporated into the cost function as an obstacle, which can significantly save search space and enhance path safety. In the low-level planning strategy, the Particle Swarm Optimization (PSO) algorithm is employed for path planning of UAVs with higher priorities. Subsequently, Dubins curves are used for smoothing the paths, enabling the UAVs to meet actual flight requirements. In the proposed algorithm, to fully leverage the results obtained from the RRT* algorithm and iterations and to accelerate convergence, several strategies are introduced, including a *restart* strategy, population initialization, and population diversification. Additionally, a time-discretized global obstacle list is introduced to consider collisions between UAVs. Finally, experimental results demonstrate that the PPSwarm algorithm can effectively satisfy collision constraints among multiple UAVs and successfully plan safe and efficient UAV flight paths, especially in large-scale and complex environments. Comparative experiments indicate that the PPSwarm algorithm outperforms five other algorithms in terms of convergence accuracy and stability, exhibiting higher optimization capabilities. In larger-scale experiments in-

volving 500 UAVs, the proposed algorithm also showcases excellent processing power and scalability.

Author Contributions: Conceptualization, Q.M., Q.Q. and K.C.; methodology, Q.M., Q.Q. and K.C.; software, Q.M., Q.Q. and K.C.; formal analysis, Q.M. and K.C.; investigation, Q.M.; resources, Q.M. and K.C.; data curation, Q.M.; writing—original draft preparation, Q.M.; writing—review and editing, Q.M.; visualization, Q.M.; supervision, Q.Q.; project administration, Q.Q.; funding acquisition, Q.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained within the article.

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

References

1. Liu, X.; Gong, D. A comparative study of A-star algorithms for search and rescue in perfect maze. In Proceedings of the 2011 International Conference on Electric Information and Control Engineering, Wuhan, China, 15–17 April 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 24–27.
2. Changxi, M.; Aixia, D.; Zhizhong, C.; Bo, Q. Notice of Retraction: Study on the hazardous blocked synthetic value and the optimization route of hazardous material transportation network based on A-star algorithm. In Proceedings of the 2011 Seventh International Conference on Natural Computation, Shanghai, China, 26–28 July 2011; IEEE: Piscataway, NJ, USA, 2011; Volume 4, pp. 2292–2294.
3. Wang, W.; Deng, H.; Wu, X. Path planning of loaded pin-jointed bar mechanisms using Rapidly-exploring Random Tree method. *Comput. Struct.* **2018**, *209*, 65–73. [CrossRef]
4. Karur, K.; Sharma, N.; Dharmatti, C.; Siegel, J.E. A survey of path planning algorithms for mobile robots. *Vehicles* **2021**, *3*, 448–468. [CrossRef]
5. Orozco-Rosas, U.; Montiel, O.; Sepúlveda, R. Mobile robot path planning using membrane evolutionary artificial potential field. *Appl. Soft Comput.* **2019**, *77*, 236–251. [CrossRef]
6. Zhang, X.; Duan, H. An improved constrained differential evolution algorithm for unmanned aerial vehicle global route planning. *Appl. Soft Comput.* **2015**, *26*, 270–284. [CrossRef]
7. Chakraborty, J.; Konar, A.; Jain, L.C.; Chakraborty, U.K. Cooperative multi-robot path planning using differential evolution. *J. Intell. Fuzzy Syst.* **2009**, *20*, 13–27. [CrossRef]
8. Chen, Y.; Chen, M.; Chen, Z.; Cheng, L.; Yang, Y.; Li, H. Delivery path planning of heterogeneous robot system under road network constraints. *Comput. Electr. Eng.* **2021**, *92*, 107197. [CrossRef]
9. Wu, H.; Li, H.; Xiao, R.; Liu, J. Modeling and simulation of dynamic ant colony's labor division for task allocation of UAV swarm. *Phys. A Stat. Mech. Appl.* **2018**, *491*, 127–141. [CrossRef]
10. Shao, S.; Peng, Y.; He, C.; Du, Y. Efficient path planning for UAV formation via comprehensively improved particle swarm optimization. *ISA Trans.* **2020**, *97*, 415–430. [CrossRef]
11. Zhang, Y.; Gong, D.W.; Zhang, J.H. Robot path planning in uncertain environment using multi-objective particle swarm optimization. *Neurocomputing* **2013**, *103*, 172–185. [CrossRef]
12. Qiaorong, Z.; Guochang, G. Path planning based on improved binary particle swarm optimization algorithm. In Proceedings of the 2008 IEEE Conference on Robotics, Automation and Mechatronics, Chengdu, China, 21–24 September 2008; IEEE: Piscataway, NJ, USA, 2008; pp. 462–466.
13. Ye, F.; Chen, J.; Tian, Y.; Jiang, T. Cooperative multiple task assignment of heterogeneous UAVs using a modified genetic algorithm with multi-type-gene chromosome encoding strategy. *J. Intell. Robot. Syst.* **2020**, *100*, 615–627. [CrossRef]
14. Shorakaei, H.; Vahdani, M.; Imani, B.; Gholami, A. Optimal cooperative path planning of unmanned aerial vehicles by a parallel genetic algorithm. *Robotica* **2016**, *34*, 823–836. [CrossRef]
15. Xue, Y.; Jiang, J.; Zhao, B.; Ma, T. A self-adaptive artificial bee colony algorithm based on global best for global optimization. *Soft Comput.* **2018**, *22*, 2935–2952. [CrossRef]
16. Aljarah, I.; Ludwig, S.A. A new clustering approach based on glowworm swarm optimization. In Proceedings of the 2013 IEEE Congress on Evolutionary Computation, Cancun, Mexico, 20–23 June 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 2642–2649.
17. Majumder, A.; Majumder, A.; Bhaumik, R. Teaching–learning-based optimization algorithm for path planning and task allocation in multi-robot plant inspection system. *Arab. J. Sci. Eng.* **2021**, *46*, 8999–9021. [CrossRef]
18. Li, H.; Chen, Y.; Chen, Z.; Wu, H. Multi-UAV Cooperative 3D Coverage Path Planning Based on Asynchronous Ant Colony Optimization. In Proceedings of the 2021 40th Chinese Control Conference (CCC), Shanghai, China, 26–28 July 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 4255–4260.
19. Ahmed, N.; Pawase, C.J.; Chang, K. Distributed 3-D path planning for multi-UAVs with full area surveillance based on particle swarm optimization. *Appl. Sci.* **2021**, *11*, 3417. [CrossRef]

20. Wang, Y.; Hu, Y.; Zhang, G.; Cai, C. Research on multi-UAVs route planning method based on improved bat optimization algorithm. *Cogent Eng.* **2023**, *10*, 2183803. [CrossRef]
21. Yu, X.; Luo, W. Reinforcement learning-based multi-strategy cuckoo search algorithm for 3D UAV path planning. *Expert Syst. Appl.* **2023**, *223*, 119910. [CrossRef]
22. Li, M.; Richards, A.; Sooriyabandara, M. Reliability-aware multi-UAV coverage path planning using a genetic algorithm. In Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems, Online, 3–7 May 2021; pp. 1584–1586.
23. YongBo, C.; YueSong, M.; JianQiao, Y.; XiaoLong, S.; Nuo, X. Three-dimensional unmanned aerial vehicle path planning using modified wolf pack search algorithm. *Neurocomputing* **2017**, *266*, 445–457. [CrossRef]
24. Qu, C.; Gai, W.; Zhang, J.; Zhong, M. A novel hybrid grey wolf optimizer algorithm for unmanned aerial vehicle (UAV) path planning. *Knowl.-Based Syst.* **2020**, *194*, 105530. [CrossRef]
25. Yang, Q.; Liu, J.; Li, L. Path planning of UAVs under dynamic environment based on a hierarchical recursive multiagent genetic algorithm. In Proceedings of the 2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, UK, 19–24 July 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–8.
26. Phung, M.D.; Quach, C.H.; Dinh, T.H.; Ha, Q. Enhanced discrete particle swarm optimization path planning for UAV vision-based surface inspection. *Autom. Constr.* **2017**, *81*, 25–33. [CrossRef]
27. Sánchez-García, J.; Reina, D.; Toral, S. A distributed PSO-based exploration algorithm for a UAV network assisting a disaster scenario. *Future Gener. Comput. Syst.* **2019**, *90*, 129–148. [CrossRef]
28. Wu, X.; Bai, W.; Xie, Y.; Sun, X.; Deng, C.; Cui, H. A hybrid algorithm of particle swarm optimization, metropolis criterion and RTS smoother for path planning of UAVs. *Appl. Soft Comput.* **2018**, *73*, 735–747. [CrossRef]
29. Das, P.; Behera, H.; Panigrahi, B. A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning. *Swarm Evol. Comput.* **2016**, *28*, 14–28. [CrossRef]
30. Das, P.K.; Behera, H.S.; Das, S.; Tripathy, H.K.; Panigrahi, B.K.; Pradhan, S.K. A hybrid improved PSO-DV algorithm for multi-robot path planning in a clutter environment. *Neurocomputing* **2016**, *207*, 735–753. [CrossRef]
31. Yu, Z.; Si, Z.; Li, X.; Wang, D.; Song, H. A Novel Hybrid Particle Swarm Optimization Algorithm for Path Planning of UAVs. *IEEE Int. Things J.* **2022**, *9*, 22547–22558. [CrossRef]
32. Ji, Y.; Zhao, X.; Hao, J. A Novel UAV Path Planning Algorithm Based on Double-Dynamic Biogeography-Based Learning Particle Swarm Optimization. *Mob. Inf. Syst.* **2022**, *2022*, 8519708. [CrossRef]
33. He, W.; Qi, X.; Liu, L. A novel hybrid particle swarm optimization for multi-UAV cooperate path planning. *Appl. Intell.* **2021**, *51*, 7350–7364. [CrossRef]
34. Erdmann, M.; Lozano-Perez, T. On multiple moving objects. *Algorithmica* **1987**, *2*, 477–521. [CrossRef]
35. Yang, P.; Tang, K.; Lozano, J.A. Estimation of distribution algorithms based unmanned aerial vehicle path planner using a new coordinate system. In Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, China, 6–11 July 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 1469–1476.
36. Ha, M.D.P.P. Safety-enhanced UAV path planning with spherical vector-based particle swarm optimization. *Appl. Soft Comput.* **2021**, *107*, 107376.
37. Dubins, L. On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents. *Am. J. Math.* **1957**, *79*, 497–516. [CrossRef]
38. Song, X.; Hu, S. 2D path planning with Dubins-path-based A* algorithm for a fixed-wing UAV. In Proceedings of the 2017 3rd IEEE International Conference on Control Science and Systems Engineering (ICCSSE), Beijing, China, 17–19 August 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 69–73.
39. Goel, U.; Varshney, S.; Jain, A.; Maheshwari, S.; Shukla, A. Three Dimensional Path Planning for UAVs in Dynamic Environment using Glow-worm Swarm Optimization. *Procedia Comput. Sci.* **2018**, *133*, 230–239. [CrossRef]
40. Van Den Berg, J.P.; Overmars, M.H. Prioritized motion planning for multiple robots. In Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, Edmonton, AB, Canada, 2–6 August 2005; IEEE: Piscataway, NJ, USA, 2005; pp. 430–435.
41. Ma, H.; Harabor, D.; Stuckey, P.J.; Li, J.; Koenig, S. Searching with consistent prioritization for multi-agent path finding. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 29–31 January 2019; Volume 33, pp. 7643–7650.
42. Bennewitz, M.; Burgard, W.; Thrun, S. Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots. *Robot. Auton. Syst.* **2002**, *41*, 89–99. [CrossRef]
43. Liang, J.J.; Qin, A.K.; Suganthan, P.N.; Baskar, S. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Trans. Evol. Comput.* **2006**, *10*, 281–295. [CrossRef]
44. Kassoul, K.; Zufferey, N.; Cheikhrouhou, N.; Belhaouari, S.B. Exponential particle swarm optimization for global optimization. *IEEE Access* **2022**, *10*, 78320–78344. [CrossRef]
45. Phadke, A.; Medrano, F.A.; Chu, T.; Sekharan, C.N.; Starek, M.J. Modeling Wind and Obstacle Disturbances for Effective Performance Observations and Analysis of Resilience in UAV Swarms. *Aerospace* **2024**, *11*, 237. [CrossRef]

46. Liang, X.; Meng, G.; Xu, Y.; Luo, H. A geometrical path planning method for unmanned aerial vehicle in 2D/3D complex environment. *Intell. Serv. Robot.* **2018**, *11*, 301–312. [CrossRef]
47. Fu, J.; Sun, G.; Liu, J.; Yao, W.; Wu, L. On hierarchical multi-UAV Dubins traveling salesman problem paths in a complex obstacle environment. *IEEE Trans. Cybern.* **2023**, *54*, 123–135. [CrossRef]