symmetry

MDPI

Article

Three-Dimensional Unmanned Aerial Vehicle Trajectory Planning Based on the Improved Whale Optimization Algorithm

Yong Yang *, Yujie Fu, Dongyang Lu, Honghui Xiang and Kaijun Xu

School of Flight Technology, Civil Aviation Flight University of China, Guanghan 618307, China; fuyujie@cafuc.edu.cn (Y.F.); ludongyang24@cafuc.edu.cn (D.L.); xianghonghui@cafuc.edu.cn (H.X.); xukaijun@cafuc.edu.cn (K.X.)

* Correspondence: yangyong@cafuc.edu.cn

Abstract: The effective planning of UAV trajectories in a 3D environment presents a complex global optimization challenge that must account for numerous constraints, including urban settings, mountainous terrain, obstacles, no-fly zones, flight boundaries, travel distances, and trajectory change rates. This paper addresses the limitations of the whale optimization algorithm in 3D trajectory planning—specifically its slow convergence, low accuracy, and susceptibility to local optimum—by proposing an improved whale optimization algorithm. This enhancement incorporates an inverse learning mechanism to increase the diversity of the initial population and integrates a nonlinear convergence factor with a random number generation mechanism to optimize the balance between global and local search capabilities. Our findings indicate that for both the standard and improved whale optimization algorithms, each individual in the population represents a feasible solution, corresponding one-to-one with distributed trajectories in the search space. Given that route planning typically occurs in three dimensions, there is spatial symmetry among the multiple potential trajectories from the starting point to the endpoint. The optimization algorithm identifies the optimal solution by exploring these symmetric trajectory paths, ultimately selecting the most favorable one based on additional constraints (e.g., no-fly zones and fuel consumption). Moreover, the convergence of the whale optimization algorithm depends on the diversity of individuals in the population and the thorough exploration of the search space. This symmetry facilitates a more uniform exploration of various trajectories by the population. In some instances, the optimization algorithm has achieved a 7.00% improvement in fitness value, a 10.05% reduction in optimal distance, and a 28.73% decrease in standard deviation. The increase in optimal values and the decrease in worst-case values underscore the effectiveness of the optimization algorithm, while the reduction in standard deviation reflects the stability of the algorithm's output data. These results further confirm the advantages of the optimized algorithm.

Keywords: unmanned aerial vehicle; three-dimensional path planning; whale optimization algorithm; inverse learning; nonlinear convergence; random number generation

Citation: Yang, Y.; Fu, Y.; Lu, D.; Xiang, H.; Xu, K. Three-Dimensional Unmanned Aerial Vehicle Trajectory Planning Based on the Improved Whale Optimization Algorithm. *Symmetry* **2024**, *16*, 1561. https://doi.org/10.3390/sym16121561

Academic Editor: Hongkun Xu

Received: 18 September 2024
Revised: 8 November 2024
Accepted: 14 November 2024
Published: 21 November 2024

1. Introduction

With the rapid development of the low-altitude economy, the application of Unmanned Aerial Vehicles [1] (UAVs)—a core technology in this sector—has garnered significant attention in low-altitude airway planning. UAVs possess autonomous control, high flexibility, and safety, demonstrating immense potential across various fields. Their applications in low-altitude economic scenarios include urban mapping, logistics and transportation, urban monitoring, crop monitoring, environmental protection, and emergency rescue. As smart cities and intelligent transportation systems evolve, the role of UAVs has expanded beyond traditional surveying and monitoring tasks to include scenarios that demand high timeliness and accuracy, such as urban logistics, distribution, emergency medical supply transport, and disaster monitoring and response.

The booming low-altitude economy has catalyzed rapid advancements in UAV technology, particularly in innovations related to intelligence, autonomy, and collaboration. Trajectory planning, a critical aspect of Unmanned Aircraft Systems (UASs), significantly influences the efficiency and safety of mission execution. To navigate the complexities of low-altitude environments and diverse mission requirements, trajectory planning must rapidly calculate collision-free and threat-free paths, while considering multiple factors, including flight path smoothness, mission execution costs, fuel consumption, and flight duration. These demands impose rigorous technical requirements on UAV applications, particularly in complex environments like logistics and distribution within densely populated urban areas and emergency medical rescues. AV path planning must not only avoid static terrain obstacles, but also adapt to dynamic conditions such as weather changes and moving obstacles encountered during flight.

In the field of UAV low-altitude path planning, numerous heuristic search algorithms have been widely applied. For example, the Ant Colony Algorithm [2] enhances UAV obstacle avoidance and path optimization in complex environments by introducing a dynamic pheromone-updating mechanism. The Particle Swarm Optimization (PSO) Algorithm [3], when combined with the Simulated Annealing algorithm, forms a hybrid method that significantly improves path planning accuracy and convergence speed in dynamic environments. Similarly, adaptations of the Grey Wolf Optimization (GWO) Algorithm [4] have optimized its predation and tracking strategies to better suit UAV path planning in 3D environments. Classical algorithms such as A* [5,6], the whale optimization algorithm (WOA) [7], and Dijkstra's Algorithm [8] are also widely used in UAV trajectory planning. Recent studies have focused on addressing path planning challenges under specialized conditions. For instance, Zhang et al. [9] proposed an improved Slap Swarm Algorithm to model UAV trajectory planning for rotary-wing UAVs. Their model considers patrol efficiency, trajectory penalties, and power consumption to overcome issues like low search efficiency and unsmooth paths in open-air warehouse environments. Cherif et al. [10] formulated the cargo–UAV mission as a multi-objective problem aimed at minimizing energy consumption, reducing handoff events, and ensuring cellular connectivity and reliability along the flight path. Yuan et al. [11] applied an enhanced PSO algorithm to optimize UAV path planning under terrain constraints, generating new constraint–altitude waypoints based on altitude and terrain resolution data. While these algorithms aim to find optimal or near-optimal paths by simulating natural group behaviors and other heuristic mechanisms, they often encounter practical challenges. Specifically, in complex 3D environments with dynamic obstacles, traditional algorithms are sensitive to initial conditions, parameter settings, and local optimum, which can limit their efficiency and accuracy in real-world applications.

To address this issue, this paper presents an improved whale optimization algorithm that incorporates a reverse learning mechanism and a nonlinear convergence factor to enhance the algorithm's global search capability and prevent it from falling into local optimum. Experimental validation demonstrates that the improved whale optimization algorithm significantly enhances convergence accuracy and computational efficiency. It is better suited for navigating complex three-dimensional environments, effectively improving the overall performance of UAVs in low-altitude route planning and providing reliable and efficient technical support for low-altitude economic applications. As technology continues to innovate, UAVs are poised to play an increasingly critical role in various low-altitude economic scenarios, including smart cities, precision agriculture, and emergency response. Furthermore, efficient and intelligent route planning algorithms will be key drivers of breakthroughs in UAV technology.

2. Modeling

2.1. Environmental Modeling

To visualize the results and the realistic aspect of the simulation environment, this paper compares and verifies the algorithm in a 3D environment. The 3D space is gridded

and numerically coded to simulate a mountainous terrain environment, while the simulation object also includes the no-fly zone. In this paper, the 3D coordinates are planned as $n_x \times n_y \times n_z$ parts, and all the height information is represented by matrix H, with z_{ij} representing the coordinates (i, j) of the three-dimensional height; in this study, $n = 100$. The simulated topography is shown in Figure 1.

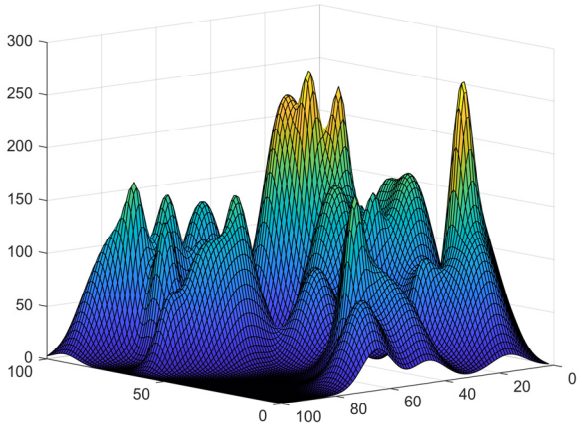

Figure 1. The simulation generates mountainous terrain maps, and each simulation generates new maps to verify that the algorithm works for different environments.

The specific expression for matrix H is as follows:

$$H = \begin{bmatrix} z_{11} & z_{12} & \cdots & z_{1\,n} \\ z_{21} & z_{22} & \cdots & z_{2\,n} \\ \vdots & \vdots & \ddots & \vdots \\ z_{n\,1} & z_{n\,2} & \cdots & z_{n\,n} \end{bmatrix} \tag{1}$$

In real-world environments, UAV flights are subject to many restrictions, such as crowded areas, military control zones, and radar zones. In this paper, these restricted areas are collectively referred to as no-fly zones, which need to be avoided during UAV trajectory planning. They were modeled as red cylinders in contact with the terrain surface model; the simulation of the no-fly zone is shown in Figure 2.

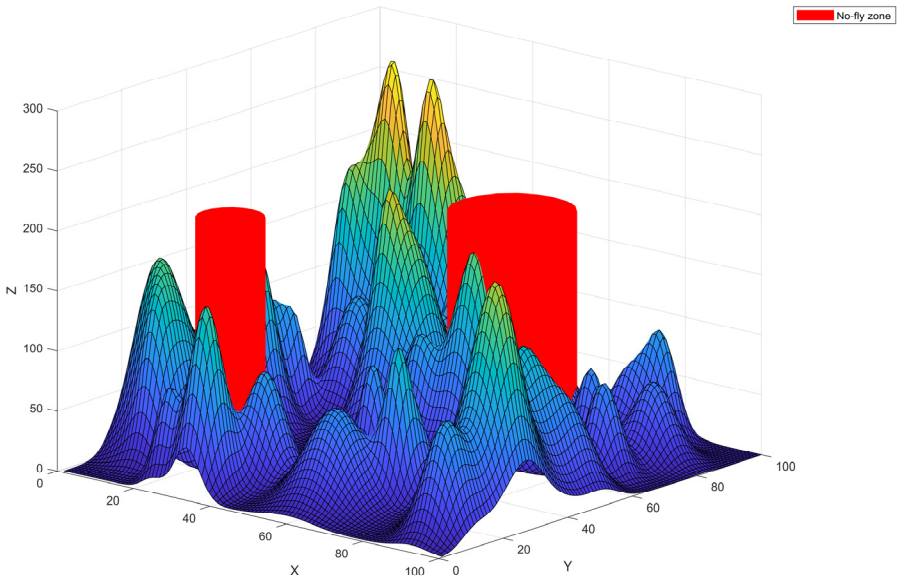

Figure 2. Simulated terrain map containing the no-fly zone.

A simulation of all cylinder model information is represented by the matrix shown in Equation (2).

$$O = \begin{bmatrix} x_1 & y_1 & r_1 \\ x_2 & y_2 & r_2 \\ \vdots & \vdots & \vdots \\ x_k & y_k & r_k \end{bmatrix} \tag{2}$$

In the formula, (x_k, y_k) represents the coordinate of the center and r_1 represents the radius of the threat zone. From a security point of view, the altitude of the threat zone is set to be infinite, i.e., no flight is allowed over the threat zone.

2.2. Penalty Function Consideration

To identify the optimal trajectory, it is crucial to consider the impact of various factors. This paper utilizes a penalty function to evaluate these influences. The total penalty of the proposed research design model encompasses terrain constraints, boundary constraints, no-fly zone constraints, flight path considerations, altitude, altitude changes, and path smoothing penalties. The objective is to minimize the overall solution penalty.

2.2.1. Terrain Constraints

To avoid collisions with the terrain during UAV flight, the height at each point along the trajectory must exceed the terrain height. In the experiment, if a point is lower than the terrain height, a positive value is generated to indicate the degree of violation of the terrain constraint; otherwise, the value is 0.

$$R_1 = \sum_{1}^{N} max(H - N + P, 0) \tag{3}$$

where R_1 represents the degree of violation of terrain constraints, H represents terrain altitude information, N represents trajectory altitude information, and P represents a safety threshold to ensure that the altitude of the path point will not be lower than the altitude of the terrain; the actual value is adjusted according to the implementation of the task and the flight environment, and is usually set as a few meters to several tens of meters. In this paper, it has a value of 1.

2.2.2. Boundary Constraints

In order to avoid the planned UAV trajectory from going beyond the flight region, the path algorithm is investigated with the requirement that each point on the trajectory does not exceed the bounds of the specified spatial coordinates. In the experiments, if the boundaries are exceeded, a positive value is generated, indicating the degree of violation of the boundary constraints; otherwise, it is 0.

$$R_2 = \sum_{i=1}^{M} \sum_{j=1}^{2} max(2 - C(i,j), 0) \tag{4}$$

$$R_3 = \sum_{i=1}^{M} \sum_{j=1}^{2} max(C(i,j) - T(j) + 2, 0) \tag{5}$$

where R_2 denotes the total extent to which the path point exceeds the left and lower boundaries of the map; R_3 denotes the total extent to which the path points exceed the right and upper boundaries of the map; C denotes the matrix of the 3D coordinates of the total path points after interpolation; M denotes the total number of path points after interpolation; i shows the first i path point, where i has two coordinate values of x and y. $j = 1$ when the i x coordinate value is taken; $j = 2$ when i takes the y coordinate value. T is a vector containing the extent of the map.

2.2.3. No-Fly Zone Constraint Penalty

In order to avoid UAVs from colliding with buildings and staying away from restricted areas during flight, in this paper, all no-fly zones are set as an infinite-height impenetrable cylinder model during the research process. The distance from each point on the trajectory to the center of the no-fly zone circle is calculated and then this distance is subtracted from the radius. If this difference is less than zero, it is replaced by zero to ensure that there are no negative values.

$$R_4 = \sum_{i=1}^{N_f} \sum_{j=1}^{N_p} \max\left(R_i - L_{ij}, 0\right) \tag{6}$$

where R_4 denotes the degree to which the indicated path point has entered the no-fly zone, N_f denotes the number of no-fly zones, N_p denotes the track point, R_i denotes the radius of the no-fly zone, and L_{ij} denotes the distance from the point on the track to the center of the no-fly zone circle.

2.2.4. Flight Trajectory Penalty

In order to reduce the distance traveled, the shortest paths are sought during trajectory planning in order to save energy. This is carried out by calculating the square root of the sum of the squares of the differences in the coordinates of two neighboring trajectory points.

$$R_5 = \sum_{i=1}^{N-1} D_i \tag{7}$$

$$D_i = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2} \tag{8}$$

where R_5 denotes the degree of flight distance, and D_i is the first i and $i+1$ distance between the path points.

2.2.5. Flight Altitude Change Penalty

In order to ensure the safety, stability, and energy efficiency of the UAV, the trajectory should be planned with as little altitude fluctuation as possible. This is achieved by calculating the variance of the altitude of the trajectory points during design.

$$R_6 = \frac{1}{N} \sum_{i=1}^{N} (z_i - \bar{z})^2 \tag{9}$$

where R_6 denotes the degree of UAV altitude fluctuation.

2.2.6. Flight Altitude Penalty

In order to maintain the advantages of UAVs flying at low altitudes, such as good maneuverability, stealth, clarity of shots, and low energy consumption, trajectory planning should aim to ensure a low altitude. This is achieved by calculating the average altitude of the trajectory points during the design process.

$$R_7 = \frac{1}{n} \sum_{i=1}^{n} z_i \tag{10}$$

where R_7 denotes the degree of UAV flight altitude, and i is the number of path points.

2.2.7. Flight Path Smoothing Penalty

In order to ensure the safety of the UAV and its surroundings during flight, as well as the feasibility of the operation and its own structure, it is required that the path should be kept smooth during trajectory planning.

$$v1_i = P_{i+1} - P_i \tag{11}$$

$$v2_i = P_{i+2} - P_{i+1} \tag{12}$$

$$\delta_i = \cos^{-1}\left(\frac{v1_i \cdot v2_i}{\|v1_i\|\|v2_i\|}\right) \tag{13}$$

$$R_8 = \sum_{i=2}^{n} \delta_i \tag{14}$$

where R_7 denotes the degree of the angle of change in all directions, which is used to measure the smoothness of the path. $v1_i$ is the same as $v2_i$, and represents the vector difference between two neighboring points in the same direction as the UAV flight direction. δ_i represents the difference between the inverse cosine of the cosine of the angle between $v1_i$ and the inverse cosine of the cosine of the angle between $v2_i$. When smoothing the path against the best searching agent, B-spline is utilized to curve and interpolate the best candidate points to form the final trajectory planning result. Currently, as can be seen from the results, it looks very curved since only three interior points (constituting nine parameters) were selected as the target search dimension. We can adjust this by increasing the dimensions to 15 or 18, which will reduce the curvature of the trajectory to some extent; however, the rest of the logic will not be affected.

2.2.8. Fitness Calculation

The fitness calculation based on the swarm algorithm is a critical part of determining the strengths and weaknesses of a search agent. We designed the fitness calculation with a constraint penalty weighting method to ensure that the fitness reflects the impact of constraints. This is shown in Equations (15) and (16), as follows:

$$f = \left[f_{length}, f_{height}, f_{smooth}, height\right] \tag{15}$$

$$fit = f \cdot [0.5, 0.1, 0.3, 0.1]^T + restion \cdot 100 \tag{16}$$

where f_{length} stands for flight trajectory penalty; f_{height} stands for flight altitude change penalty; f_{smooth} stands for flight path smoothing penalty; *height* stands for flight altitude penalty; *restion* stands for the sum of terrain constraints, boundary constraints, and no-fly zone constraints; and fit stands for fitness. These penalty values are separately implying the importance of length cost, vertical smoothness, terrain-related height, and horizontal smoothness characteristics in the corresponding planning result, in which the specified values can be setup accordingly and adjusted dynamically in more optimized methods in further investigations.

Meanwhile, the distance of each search agent is obtained by calculating the Euclidean distance between consecutive track points on the trajectory, and then the lengths of all the line segments are summed to obtain the flight distance of the complete planned trajectory, as shown in Equations (17) and (18):

$$L_i = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \tag{17}$$

$$distance = \sum_{i=1}^{n-1} L_i \tag{18}$$

3. Whale Optimization Algorithm

The whale optimization algorithm (WOA) is a meta-heuristic intelligent optimization algorithm inspired by the foraging behavior of whales, proposed by Seyedali Mirjalili and Andrew Lewis in 2016 [12]. The mathematical model is grounded in three primary foraging behaviors of humpback whales—encircling prey, bubble net feeding, and searching for food. The flowchart of the algorithm computation is shown in Figure 3.

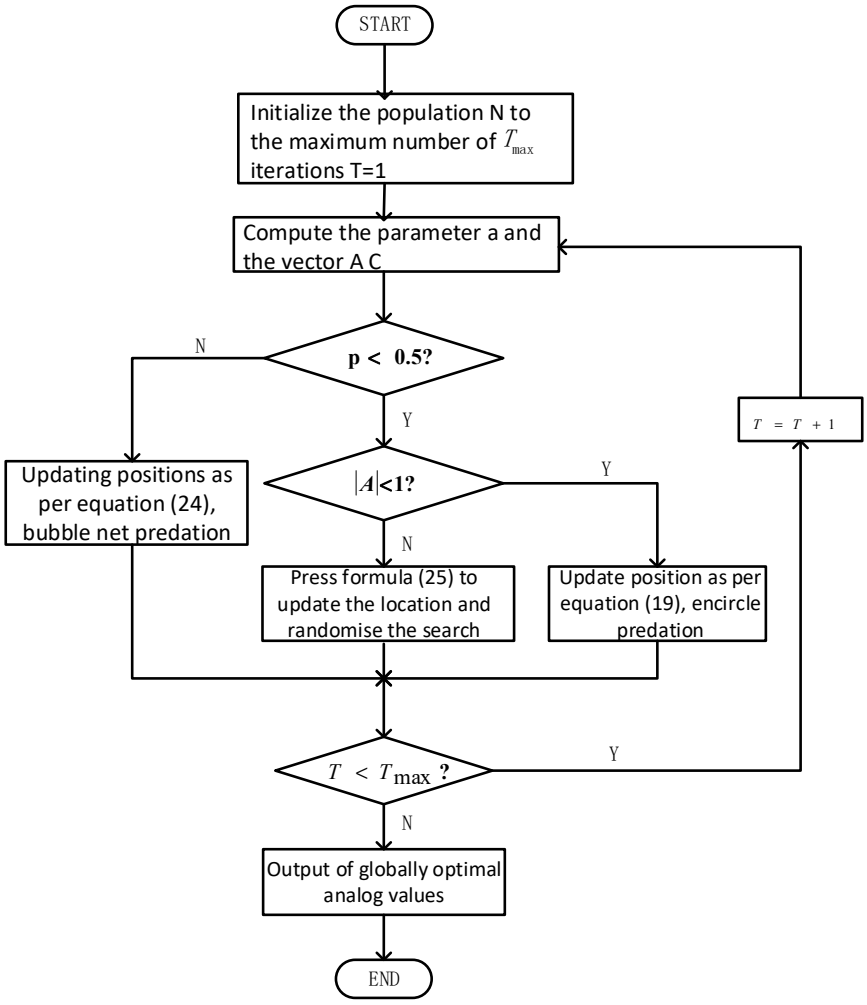

Figure 3. Whale optimization algorithm flowchart. The process reflects the stages of initialization, encircling prey, bubble net predation, and searching for prey.

3.1. Surrounding the Prey

The locally optimal solution for encircling the prey is the one closest to the food, and the other individuals gradually approach the locally optimal individual to encircle the food. The mathematical model for this stage is as follows:

$$D = \left| C \cdot x_* - x_i^{t-1} \right| \tag{19}$$

$$x_i^t = x_* - A \cdot D \tag{20}$$

x_* is the current optimal solution and x_i^t denotes the position of the ith whale in generation t. A and C are coefficient vectors, which have the following expressions:

$$A = 2a \cdot r - a \tag{21}$$

$$C = 2 \cdot r \tag{22}$$

where r is a random number uniformly distributed on [0, 1] and α is the convergence factor, whose expression is as follows:

$$a = 2 - \frac{2t}{T_{max}} \tag{23}$$

where t is the current number of iterations, and T_{max} is the maximum number of iterations.

3.2. Subsection Bubble Attacks

There are two attack mechanisms that occur during bubble net predation—the constrictive encirclement mechanism and the spiral updating position. The constriction enclosure mechanism is encircling predation, which is achieved by lowering the α value. Now, we can assume that the probability of both is 0.5. The mathematical model for this stage is as follows:

$$x_i^t = x_* - A \cdot \left| C \cdot x_* - x_i^{t-1} \right| (p < 0.5) \tag{24}$$

$$x_i^t = x_* - A \cdot \left| C \cdot x_* - x_i^{t-1} \right| \cdot e^{bl} \cdot \cos(2\pi l) (p \geq 0.5) \tag{25}$$

p is a random number on [0, 1], obeying a uniform distribution; b is a constant coefficient; and l is a random number between [0, 1].

3.3. Random Search

Referring to the whale at this point, A has a value greater than 1 or less than -1, and the search agent's position in the exploration phase will be updated here based on the randomly selected search agent, not on the optimal search agent so far [13].

$$X_{t+1} = x_{rand} - A \cdot D_{rand} \tag{26}$$

$$D_{rand} = |C \cdot x_{rand} - x_t| \tag{27}$$

where x_{rand} denotes the coordinates of any whale.

3.4. WOA Pseudo-Code

In summary, the pseudo-code of the IWOA part of the algorithm flow is shown in Algorithm 1.

Algorithm 1 WOA pseudo-code

Input: initialized parameters
Output: optimal solution

01: Set the population size N and the maximum number of iterations T
02: Initialize the population and calculate the fitness value for each individual to determine the best individual x_*
03: While (t < T),
04: For each search agent,
05: Update, a, A, C, l, p
06: If $p < 0.5$,
07: If $|A < 1|$,
08: Use Equation (19) to update the position of the current search agent
09: Else,
10: Use Equation (25) to update the position of the current search agent
11: End If
12: Else,
13: Use Equation (24) to update the position of the current search agent
14: End If
15: End For
16: Check if any search agent goes beyond the search and amend it
17: Calculate the fitness of each search agent
18: update x_*
19: t = t + 1
20: End While

4. Improving the Whale Optimization Algorithm

This section is organized with subheadings and provides a concise description of the experimental results, their interpretation, and the conclusions drawn from the experiments.

The WOA [14] features a straightforward structure with minimal parameters to adjust, making it suitable for both continuous and discrete optimization problems. However, in some complex high-dimensional scenarios—particularly those involving spatially intricate and multi-peaked landscapes—the WOA's convergence can be slow. Even after numerous iterations, it may still succumb to a local optimum. In this paper, we propose an IWOA. The IWOA enhances the original algorithm by incorporating reverse learning to increase the diversity of the initial population, transitioning the convergence factor from a linear to a nonlinear decrease, and integrating a random number generation mechanism into the iterative process to balance the algorithm's global search and local development capabilities. The coordination of global search and local development is a key feature of the IWOA [15]. The IWOA flowchart is shown in Figure 4.

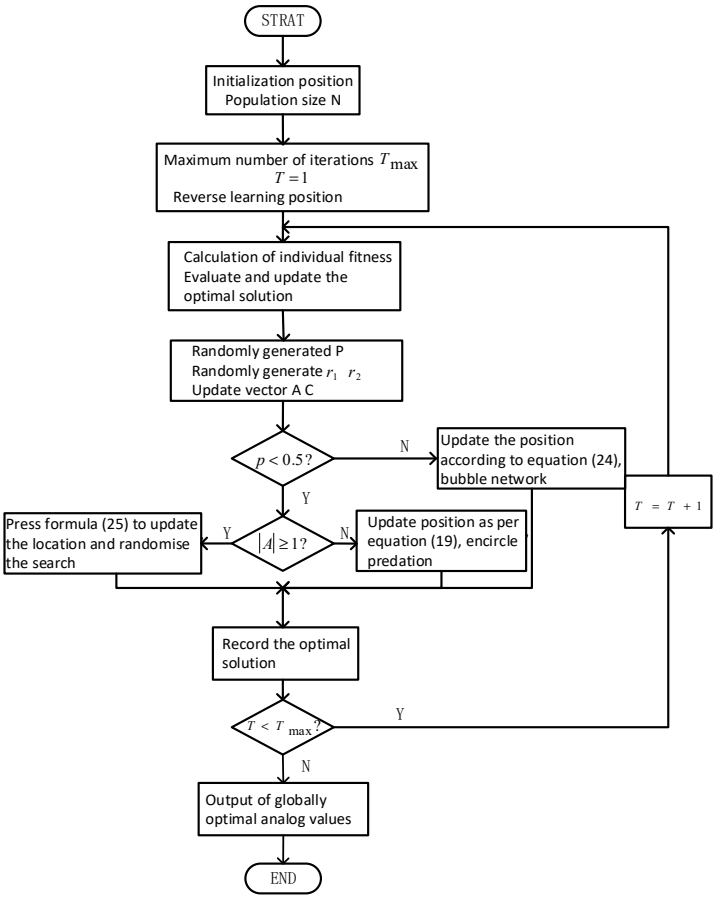

Figure 4. The improved whale optimization algorithm flowchart. The process compared with the original algorithm in the initial stage with the reverse learning mechanism to expand the search range; the late addition of nonlinear convergence factors improves the ability to solve the local search.

4.1. Reverse Learning Initialization

Inverse learning is a commonly used global optimization strategy that is currently proven and applied in many swarm intelligence algorithms [16]. The idea of reverse learning is to use the current solution and its inverse solution to improve the search efficiency. In the study, it is assumed that the dimension of the search space is D; the population size is N; and the upper and lower bounds of the search space are, respectively, a and b. The initial population is X, where each individual is x_{ij}, and the inverse solution

of the population is X', where each individual of the initial population is x'_{ij}. The formula is as follows:

$$x'_{ij} = l(a_j + b_j - x_{ij}) \tag{28}$$

$$l = (a_2 - 1)\cdot r_{rand} + 1(-2 < r_{rand} < -1) \tag{29}$$

where l is the reverse learning factor, where $[0, 2]$ is the random number.

$$X = SelectBest(X, X') \tag{30}$$

At this point, X is the population after calculating the fitness update.

4.2. Nonlinear Convergence Factors

In the WOA, a is the convergence factor [17], which decreases linearly from 2 to 0, as formulated in Equation (19). At a approaching 2, due to the complexity of the space, the step size of its searching agent decreases rapidly as a result, which makes the exploration range of the global searching phase shrink rapidly, resulting in missing some potential globally optimal regions. The convergence of a in the algorithm from global search to local search can also lead to a premature shift from global search to local search, resulting in local optimums. The convergence of the algorithm from global search to local search can also involve a premature shift, resulting in local optimum. The algorithm converges to a local optimum as a approaches 0. The linearly decreasing convergence property slows down the change in its value, making the step size of the local search less significant and weakening the ability to search around the local optimal solution. To address the above issues, this paper proposes a nonlinear convergence factor with the following specific formula:

$$a_{nonlinear} = \alpha + \frac{1 - \beta\cdot(t/T_{max})}{1 - \gamma\cdot(t/T_{max})} \tag{31}$$

In order to ensure that the $t = 0$ time $a_{nonlinear}$ is close to 2, at $t = T_{max}$ time, $a_{nonlinear}$ is close to 0, and $\alpha = 1$, The $\beta = 0.5$, *and* $\gamma = 50$. As a result, there is a smaller value of t in the early stages when the need to decline is slow, with little change in the numerator denominator, which results in $a_{nonlinear}$ having less variation in the early stage and the global search capability is improved. In the middle stage, the algorithm's ability to balance between global and local search is improved because the convergence factor varies nonlinearly. In the late stage, when fast descent is required, a larger value of t makes the $a_{nonlinear}$ rapid descent in the later stages and enhances the local exploitation capability.

4.3. Random Number Generation Mechanism

In population intelligence algorithms, it is common to generate random numbers to improve the algorithm's global and local search capabilities [18]. In this paper, the random number generation mechanism is used to optimize the algorithm during individual initialization and position updates. During individual initialization, the random number p is used to determine the prey action to be selected when the individual is updated. For position updates, the random number r_1 is used to determine the predator action to be selected during the individual update. The random number r_2 and parameters A and C are shown in Equations (21) and (22); the index of the individual is selected by the random number. For the calculation of the nonlinear convergence factor, the random number r_{rand} ensures that l has randomness, as shown in Equation (28), so that different spiral trajectories can be produced and the algorithm can avoid falling into a local optimum.

4.4. Pseudo-Code

In summary, the pseudo-code of the IWOA part of the algorithm flow is shown in Algorithm 2.

Algorithm 2 WOA pseudo-code

Input: initialized parameters
Output: optimal solution

```
01: Set the population size N and the maximum number of iterations T
02: Initialize the population by using Equations (28)–(30) and calculate fitness for each individual
to determine the best individual x*
03: While t (t < T),
04:   For each search agent,
05:   Update a, l, A, C, P
06:   Modify the position of the current search agent to ensure it is within bounds
07:     Update a using Equation (31)
08:     Generate p for current agent
09:       For each dimension j = 1 to dim
10:           Generate random values r1 = rand(), r2 = rand()
11:         If p < 0.5,
12:               If |A < 1|,
13:                  Use Equation (19) to update the position of the current search agent
14:               Else,
15:                  Use Equation (25) to update the position of the current search agent
16:               End If
17:           Else,
18:              Use Equation (24) to update the position of the current search agent
19:           End If
20:     End For
21:     Check if any search agent goes beyond the bounds and amend it using constrains
22:     Calculate the fitness of each search agent
23:     Update x*
24:     t = t + 1
25: End While
```

As shown in pseudo-code table line 02, the reverse learning mechanism is incorporated during the initialization phase. By generating new candidate solutions across the solution space via reverse learning, this mechanism effectively broadens the initial population's coverage, enhancing diversity and reducing the risk of early convergence to local optimum. In code table line 07, the second enhancement focuses on adjusting the convergence behavior of vector a. By optimizing the nonlinear convergence factor, the search strategy transitions from exploration to exploitation, enabling the algorithm to converge more rapidly towards known optimal solutions. Code table lines 08–10 illustrate the third improvement, namely the inclusion of a random number generation mechanism. This mechanism independently generates a random value p in each iteration and applies independent random variables to each dimension within the inner loop, increasing the probability of discovering globally optimal solutions and reducing early convergence risks. Finally, as shown in code table line 21, rather than confining search agents to the boundary when they exceed it, this study introduces a constraint-handling strategy. For individuals that surpass the boundary, a random position near the boundary is assigned, allowing search agents to explore the boundary region, which increases the likelihood of finding improved solutions.

5. Simulation Verification and Result Analysis

In this study, the 3D space is planned as a 100 × 100 × 100 grid, and 30 sittings of peaks are randomly generated, as shown in Figure 5.

No-fly zones are also planned in space, which are non-flyable cylinders with base coordinates and radii, as shown in Table 1.

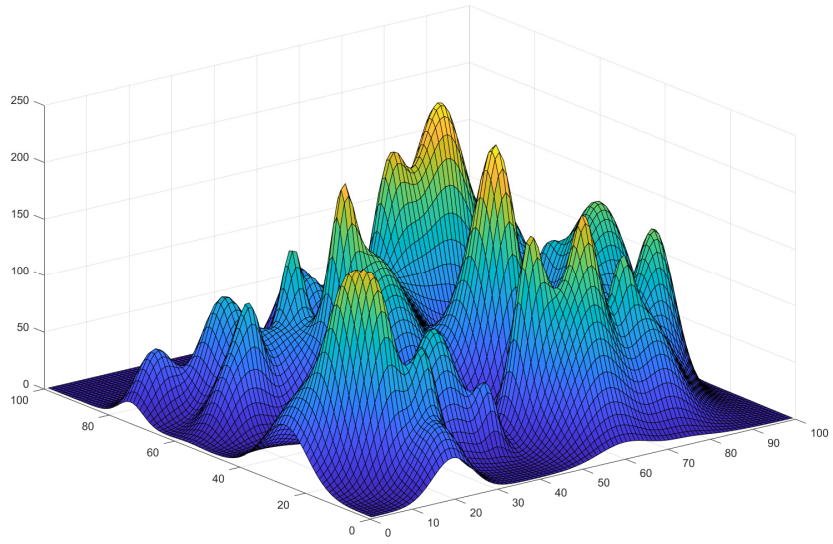

Figure 5. Terrain model.

Table 1. No-fly zone information.

Threat Zone	Center of the Circle	Radius
Cylinder 1	(25,25)	16
Cylinder 2	(75,60)	5

5.1. Simulation Results

As mentioned earlier, the whale optimization algorithm was modified to study UAV trajectories. Both algorithms used the same start and end points in a 3D Cartesian coordinate system; the value of z is set as the height of the surface on which the given starting point or ending point lies plus ten. The computation was set to three times, the population size was set to 80, and each computation was iterated 500 times for a total of six experiments. The performance information of the two algorithms for six test cases is shown in Table 2; the simulation results are shown in Figure 6; and the convergence results are shown in Figure 7.

The IWOA employs a reverse learning strategy during the initialization phase to generate a set of solutions relative to the initial solution. This approach effectively broadens the distribution of solutions and enhances the algorithm's exploration capabilities, allowing it to achieve a better balance between global and local search throughout the optimization process. The accompanying figure illustrates that the IWOA curve declines more rapidly and smoothly, indicating that the algorithm quickly approaches the optimal solution in the early stages, while maintaining stable convergence in the later stages. In contrast, the traditional WOA tends to focus more on exploration during the initial phase and shifts to exploitation later. However, it sometimes loses the ability to adequately explore the local optimum. This is evident from the figure, which shows that the WOA curve declines more slowly and exhibits significant fluctuations around 150 iterations, suggesting that the algorithm may become trapped in local optimum.

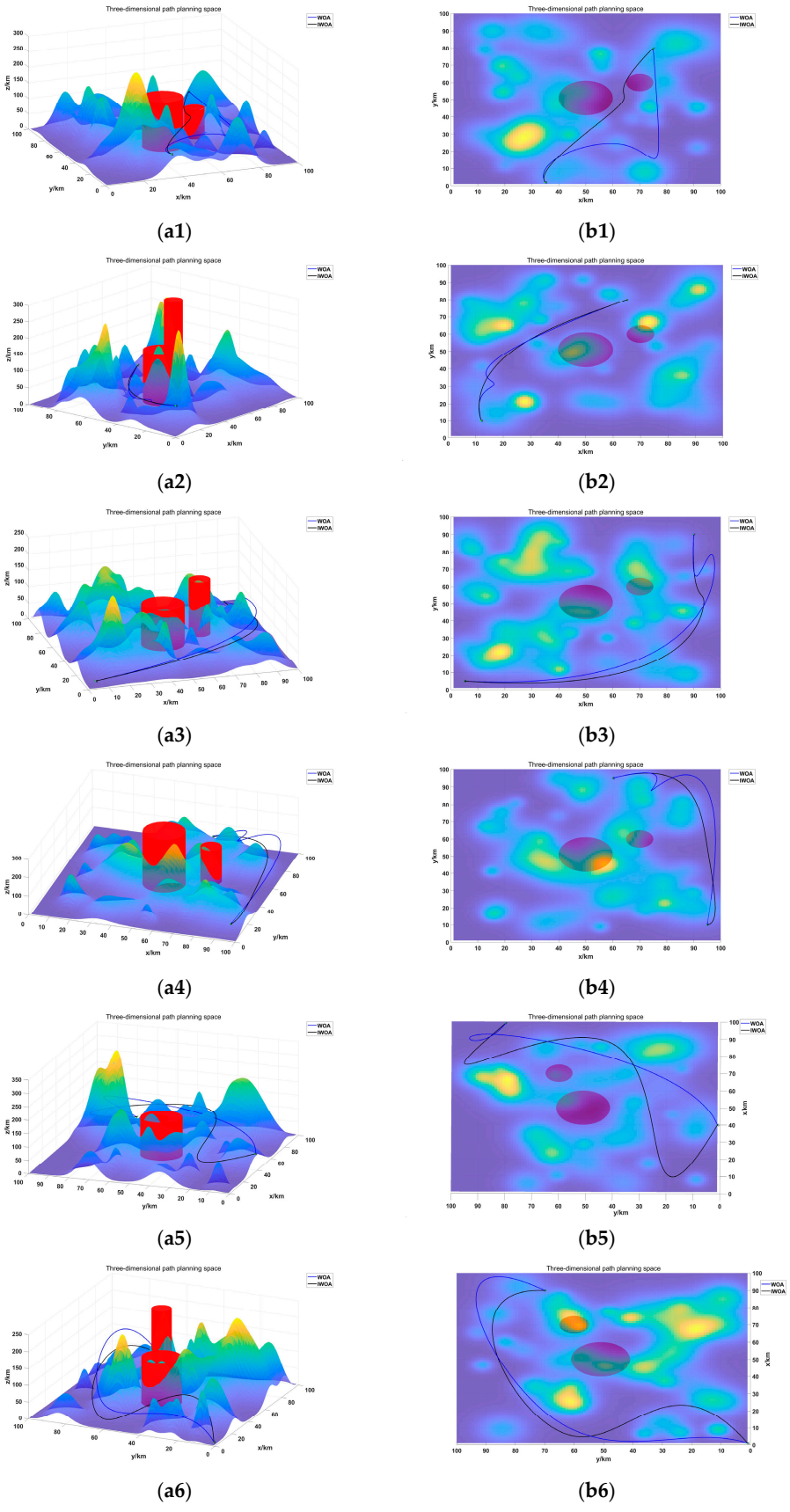

Figure 6. (**a1**–**a6**) show a side view of the two algorithms planning a trajectory in the same terrain. (**b1**–**b6**) show a top view of the two algorithms planning a trajectory in the same terrain. The performance of the two algorithms is visualized in the simulation environment.

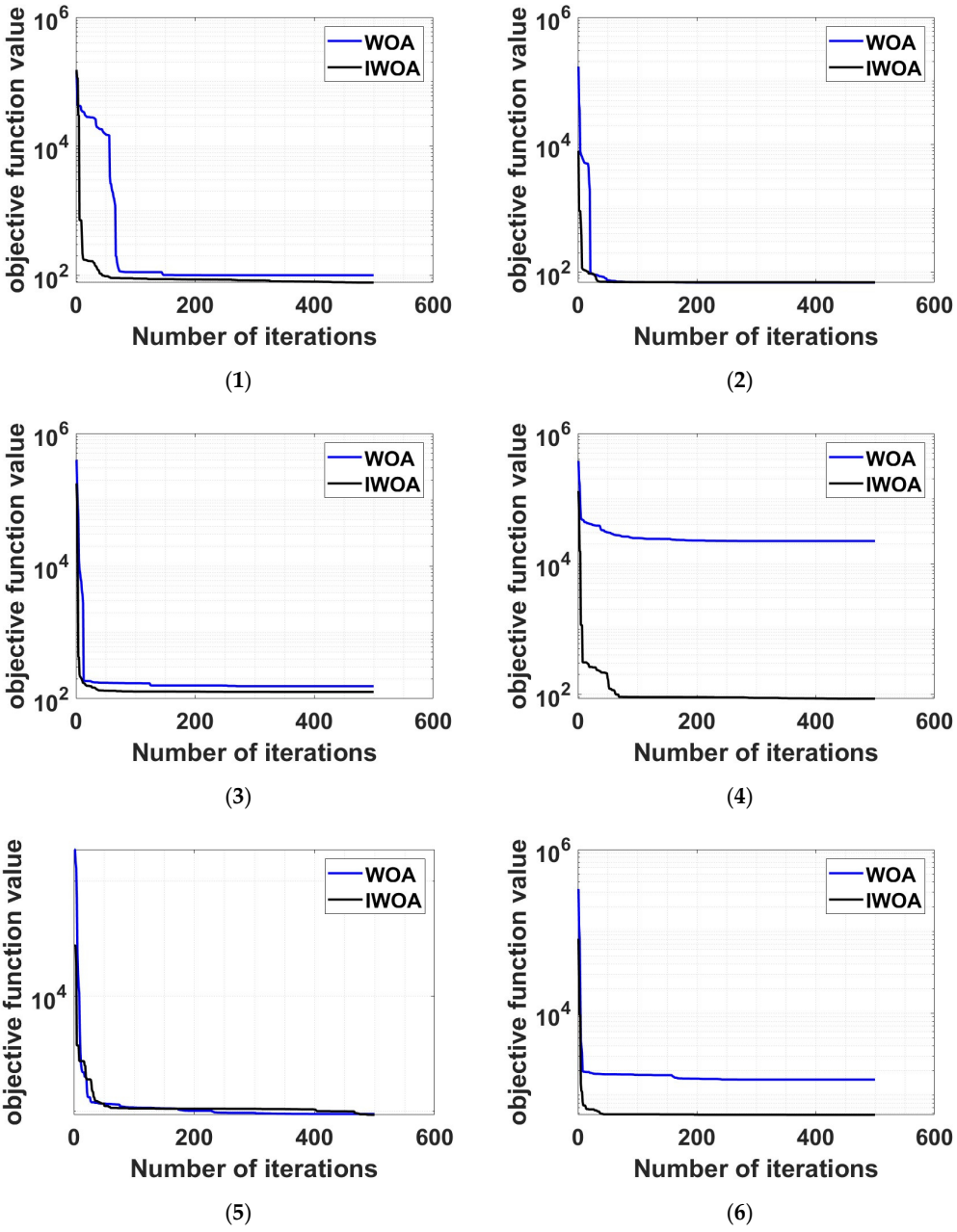

Figure 7. (**1**) Fitness curves for group 1 experiments (corresponding to Figure 6(a1,b1)); (**2**) Fitness curves for group 2 experiments (corresponding to Figure 6(a2,b2)); (**3**) Fitness curves for group 3 experiments (corresponding to Figure 6(a3,b3)); (**4**) Fitness curves for group 4 experiments (corresponding to Figure 6(a4,b4)); (**5**) Fitness curves for group 5 experiments (corresponding to Figure 6(a5,b5)); (**6**) Fitness curves for group 6 experiments (corresponding to Figure 6(a6,b6)).

5.2. Results Analysis

The simulation results of both algorithms in a unified environment are presented in Figure 6. When addressing the 3D trajectory planning problem, both algorithms successfully plan smooth trajectories between the starting and ending points, effectively avoiding collisions with mountains and no-fly zones. From the front view in Figure 6(a1–a6), it is evident that the IWOA selects a shorter route to circumvent the mountain, resulting in a more efficient trajectory. Additionally, the top view in Figure 6(b1–b6) shows that the IWOA generates a smoother path while avoiding the mountain and no-fly zones, which significantly enhances the UAV's flight safety. This improvement can be attributed to the WOA's tendency to enter the local search too early, leading to local optimum. Moreover,

the data in Table 2 indicate that the IWOA improves the fitness value by approximately 7.00%, reduces the optimal distance by about 10.05%, and decreases the standard deviation by roughly 28.73% compared to the WOA. The relevant data are shown in Table 2.

Table 2. Algorithm performance information.

Number	Location		Algorithm					
			WOA			IWOA		
	Start	End	Optimal Fitness	Optimal Distance	Variance	Optimal Fitness	Optimal Distance	Variance
1	(35, 2, 10.69)	(75, 80, 38.48)	87.37	129.02	99.89	44.84	99.46	42.05
2	(12,10,75.54)	(65, 80, 57.47)	70.28	99.74	81.24	70.02	97.63	66.67
3	(5, 5, 19.71)	(90, 90, 144.64)	168.09	194.49	123.47	162.01	190.27	100.08
4	(60, 95, 11.20)	(95, 10, 42.75)	92.02	149.80	64.49	88.48	135.07	41.39
5	(40, 1, 10.07)	(99, 80, 33.07)	387.3	176.39	125.78	369.6	163.68	95.53
6	(1, 1, 11.20)	(90, 70, 35.14)	532.29	295.43	197.11	508.75	253.71	147.39

A comprehensive analysis reveals that the incorporation of improved methods—such as reverse learning, a nonlinear convergence factor, and a random number generation mechanism—effectively optimizes the algorithm. The IWOA demonstrates enhanced global and local search capabilities, allowing it to achieve lower total penalties, shorter paths, and smoother trajectories in three-dimensional trajectory planning problems compared to the WOA.

Four common optimization algorithms were selected—PSO: Particle Swarm Optimization [19]; SWO: spider wasp optimization algorithm; GRO: Gold rush optimizer [20]; and KOA: Keplerian Optimization Algorithm [21]—to verify the performance of the IWOA. In the comparison test, the populations of the IWO algorithm and WOA proposed in this paper, along with the remaining six algorithms, are set to 50, and the maximum number of iterations is set to 100; the test results are shown in Figures 8 and 9.

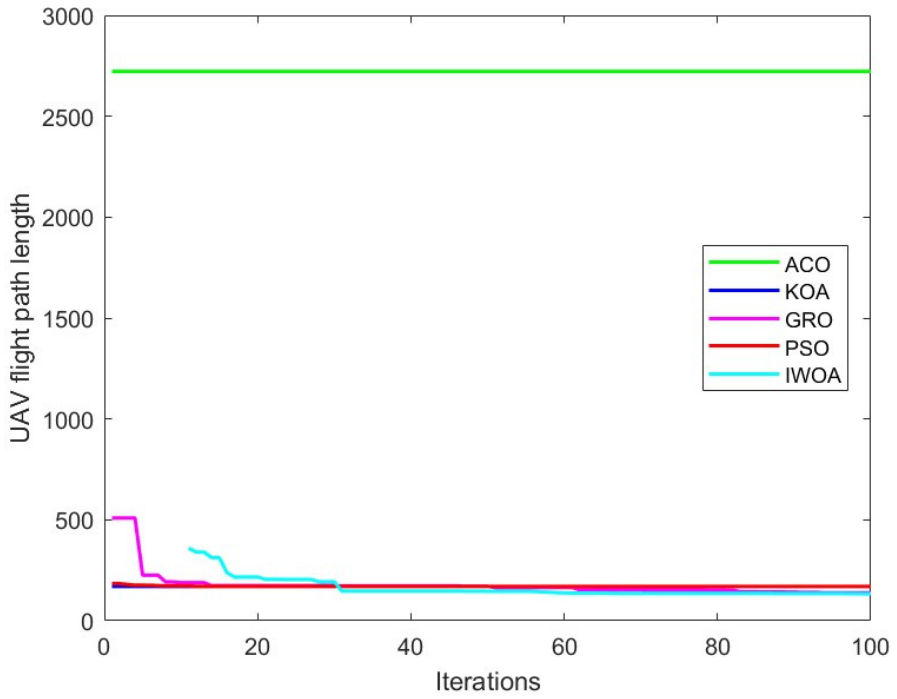

Figure 8. Comparison of convergence curves of multiple algorithms.

In this paper, the time complexity and space complexity of the algorithms are computed using the tic, toc, and memory functions. The population is set to 30 and the maximum number of iterations is 100. The data obtained under the same conditions are shown in the table below. From Table 3, it is clear that the IWOA takes the least amount of

time to compute, followed by the KOA, PSO, GRO, and ACO. The KOA takes up the most storage space, particularly the ACO, GRO, PSO, and IWOA.

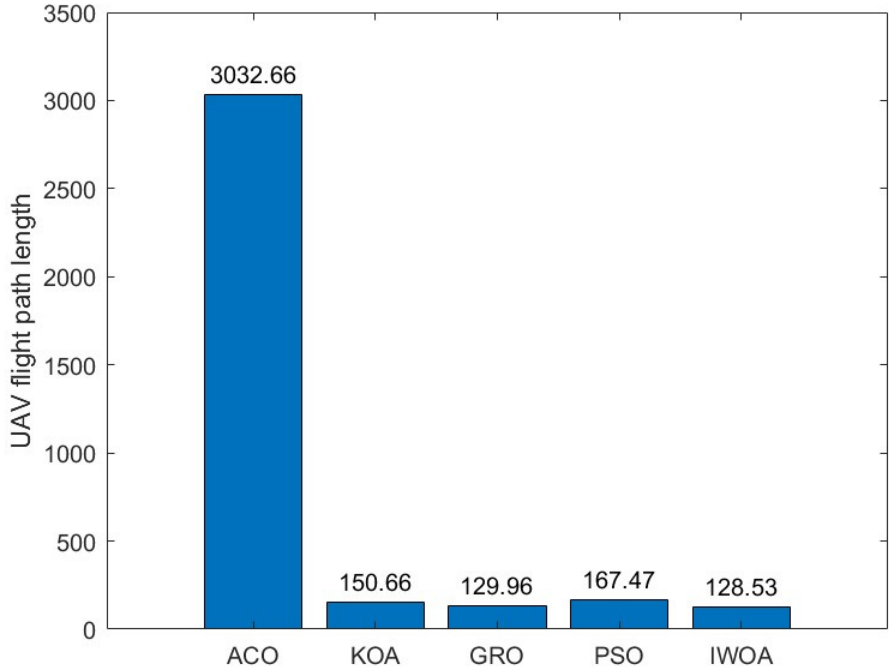

Figure 9. Comparison of path lengths of various algorithms.

Table 3. Algorithm complexity data information.

Algorithm	Time Use (S)	Before Size	After Size	Before Execution (Byte)	After Execution (Byte)
ACO	0.38066	1×16	1×15	38	36
PSO	0.074246	1×16	1×15	40	38
KOA	0.060862	1×16	1×15	38	36
GRO	0.098793	1×16	1×15	38	36
IWOA	0.029181	1×16	1×15	38	36

6. Conclusions

In this paper, we introduce the IWOA, an enhancement of the conventional WOA that incorporates inverse learning, nonlinear convergence factors, and a random number generation mechanism. Inverse learning functions as an effective initialization and search strategy; it generates both regular solutions and their corresponding inverse solutions. This dual-solution generation method enables the initial and inverse solutions to cover distinct regions of the search space, as the inverse solutions are created according to specific rules that differentiate them from regular solutions in terms of value and position. Consequently, this approach mitigates the problem of over-concentration in solution distribution and assists the algorithm in avoiding local optimum.

Traditional optimization algorithms typically use linear convergence factors to manage the transition between the exploration and exploitation phases. In contrast, nonlinear convergence strategies offer a smoother and more flexible exploration and exploitation process through more sophisticated control mechanisms. For instance, a nonlinear convergence factor may adjust its value based on a complex function related to the iteration number or the current solution quality, allowing the algorithm to adaptively modify the search step size compared to a simple linear decrease. Additionally, random number generation plays a crucial role in optimization algorithms by enhancing diversity and preventing convergence to local optima.

With these enhancements, the inverse whale optimization algorithm (IWOA) significantly outperforms traditional methods such as the conventional whale optimization algorithm (WOA), Particle Swarm Optimization (PSO), and other similar algorithms. As a result, the IWOA is particularly well suited for UAV trajectory planning under various constraints in simulation experiments. The improvements effectively address the issue of local optimum, enhance the algorithm's capability to manage individual boundary constraints, and ensure its applicability across a range of optimization problems. Future research will focus on exploring additional influencing factors to enrich the model and will also focus on UAV trajectory planning in dynamic environments.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/sym16121561/s1.

Author Contributions: Writing—review and editing, Y.Y.; writing—original draft preparation, Y.F.; visualization, D.L.; investigation, H.X.; data curation, K.X. All authors have read and agreed to the published version of the manuscript.

Funding: This study has been supported by the Open Fund Project of the Key Laboratory of Civil Aviation Flight Technology and Fight Safety (Grant No. FZ2021ZZ06). This work has also been supported by Central University Basic Research Projects (Grant No. 24CAFUC04002) and the Sichuan Engineering Research Center for Smart Operation and Maintenance of Civil Aviation Airports (Grant No. JCZX2023ZZ07). This work has also been supported by Sichuan Flight Engineering Technology Research Center Project (Grant No. GY2024-30D).

Data Availability Statement: The original contributions presented in the study are included in the article/Supplementary Material, further inquiries can be directed to the corresponding authors.

Acknowledgments: We are sincerely grateful to the reviewers who have been highly professional, meticulous, and dedicated. Their insightful comments have been of great value to us. We also extend our gratitude to the Editors for their prompt feedback. Additionally, we would like to thank Fan Li for his painstaking efforts in proofreading the language of the paper.

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

References

1. Tao, Y.J.; Li, P.F. An Overview of UAS Development and Key Technologies. *Aviat. Manuf. Technol.* **2014**, *20*, 34–39.
2. Ma, Z.; Zhu, Q.; Zhang, J.; Qu, Q. Ant Colony Optimization Algorithm for UAV Indoor Trajectory Planning. *J. Xi'an Univ. Sci. Technol.* **2022**, *42*, 307–316.
3. Deng, L.; Chen, H.; Zhang, X.; Liu, H. Three-Dimensional Path Planning of UAV Based on Improved Particle Swarm Optimization. *Mathematics* **2023**, *11*, 1987. [CrossRef]
4. Xu, L.; Zhao, W. UAV Trajectory Planning Based on a Novel Grey Wolf Optimization Algorithm. *Electron. Meas. Technol.* **2022**, *45*, 55–61.
5. Fu, D.; Fan, P. Improvement of A* Algorithm for 3D UAV Path Planning. *Intell. Compute. Appl.* **2020**, *10*, 155–159.
6. Funk, N.; Tarrio, J.; Papatheodorou, S.; Alcantarilla, P.F.; Leutenegger, S. Orientation-Aware Hierarchical, Adaptive-Resolution A* Algorithm for UAV Trajectory Planning. *IEEE Robot. Autom. Lett.* **2023**, *8*, 6723–6730. [CrossRef]
7. Zhang, F.; Liu, Z.; Lin, D. Optimal Design and Implementation of UAV Trajectory Control Module. *Compute. Technol. Autom.* **2024**, *43*, 16–24.
8. Yang, X.; Qu, D. A Trajectory Optimization Method Based on Genetic Simulated Annealing Algorithm. *Sichuan J. Mil. Eng.* **2013**, *34*, 66–70.
9. Zhang, J.; An, Y.; Cao, J.; Ouyang, S.; Wang, L. UAV Trajectory Planning for Complex Open Storage Environments Based on an Improved RRT Algorithm. *IEEE Access* **2023**, *11*, 23189–23204. [CrossRef]
10. Cherif, N.; Jaafar, W.; Yanikomeroglu, H.; Yongacoglu, A. RL-Based Cargo-UAV Trajectory Planning and Cell Association for Minimum Handoffs, Dysconnectivity, and Energy Consumption. *IEEE Trans. Veh. Technol.* **2024**, *73*, 7304–7309. [CrossRef]
11. Yuan, J.; Liu, Z.; Xiong, X.; Ai, Y.; Chen, L.; Tian, B. UAV Path Planning With Terrain Constraints for Aerial Scanning. *IEEE Trans. Intell. Veh.* **2024**, *9*, 1189–1203. [CrossRef]
12. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
13. Hua, W.; Huang, W.; Yu, F. Improved Model-Predictive-Flux-Control Strategy for Three-Phase Four-Switch Inverter-Fed Flux-Reversal Permanent Magnet Machine Drives. *IET Electr. Power Appl.* **2017**, *11*, 717–728. [CrossRef]
14. Yu, J.; Gao, N.; Li, H. A Whale Algorithm Based on Nonlinear Convergence Factor and Local Perturbation. *Compute. Eng. Des.* **2019**, *40*, 2861–2866.

15. Zhang, Q.; Long, W.; Li, Y.; Li, Y. Wear Monitoring of Milling Cutter Based on Whale Algorithm Optimized LSSVM. *J. Sichuan Univ. Nat. Sci.* **2022**, *59*, 68–74.
16. Wang, Y.; Huang, L.; Zhong, J.; Hu, G. LARO: Opposition-Based Learning Boosted Artificial Rabbits-Inspired Optimization Algorithm with Lévy Flight. *Symmetry* **2022**, *14*, 2282. [CrossRef]
17. Zhang, Y.; Du, S.; Zhang, Q. Improved Slime Mold Algorithm with Dynamic Quantum Rotation Gate and Opposition-Based Learning for Global Optimization and Engineering Design Problems. *Algorithms* **2022**, *15*, 317. [CrossRef]
18. Shuo, L.B.; Song, K.Z.; Wang, Y.F. FPGA Implementation and Research of Pseudo-Random Number Generator. *J. Circuits Syst.* **2003**, *83*, 121–124.
19. Zhan, Z.; Zhang, J.; Li, Y.; Chung, H. Adaptive Particle Swarm Optimization. *IEEE Trans. Syst. Man Cybern.* **2009**, *39*, 1362–1381. [CrossRef]
20. Zolfi, K. Gold rush optimizer: A new population-based metaheuristic algorithm. *Oper. Res. Decis.* **2023**, *33*, 113–150. [CrossRef]
21. Hakmi, S.H.; Shaheen, A.M.; Alnami, H.; Moustafa, G.; Ginidi, A. Kepler Algorithm for Large-Scale Systems of Economic Dispatch with Heat Optimization. *Biomimetics* **2023**, *8*, 608. [CrossRef] [PubMed]