aerospace

MDPI

Article

3D Path Planning for UAVs Based on an Improved DOA

Weiqi Feng [1,2], Hongyu Chen [1,2], Yujie Fu [1,2], Yong Yang [1,2] and Kaijun Xu [1,2,*]

[1] School of Flight Technology, Civil Aviation Flight University of China, Guanghan 618307, China; fengweiqi@cafuc.edu.cn (W.F.); chenhongyu@cafuc.edu.cn (H.C.); fuyujie@cafuc.edu.cn (Y.F.); yangyong@cafuc.edu.cn (Y.Y.)

[2] Sichuan Provincial Engineering Research Center of Domestic Civil Aircraft Flight and Operation Support, Chengdu 610021, China

* Correspondence: kjxu@cafuc.edu.cn

Abstract

Three-dimensional (3D) path planning for Unmanned Aerial Vehicles (UAVs) presents a challenging multi-objective optimization problem that necessitates a balanced trade-off among path length, flight safety, and trajectory smoothness, especially in complex environments such as mountainous or hilly terrains. Traditional and even many meta-heuristic planning algorithms often suffer from premature convergence and suboptimal solution quality when navigating such intricate 3D spaces. To address these limitations, this paper proposes an Improved Dhole Optimization Algorithm (IDOA) that exhibits fast convergence and strong global optimization capabilities. The IDOA enhances the original DOA framework by integrating a logistic-map-based chaotic mapping, a dynamic chaotic perturbation mechanism, and an adaptive stage-division strategy. The algorithm is designed to address the 3D path planning problem for quadrotor UAVs, supporting typical flight maneuvers including climb/descent, obstacle avoidance, and smooth turning in simulated complex hilly terrain. A multi-objective fitness function incorporating path length, safety, and smoothness is designed, which constrains the optimization to generate collision-free, smooth paths that satisfy the quadrotor UAV's dynamic maneuver constraints. Convergence curves confirm that IDOA significantly outperforms the original DOA in terms of convergence speed and final path optimality. Detailed experimental results show that compared to the original DOA, IDOA achieves a 6.31% improvement in minimum fitness values, a 9.7% reduction in average path length, and a 45.28% reduction in average path curvature when compared to the baseline DOA. These consistent performance improvements demonstrate that IDOA provides an effective and robust solution for offline pre-flight 3D path planning in complex terrain, offering valuable technical support for autonomous UAV navigation in practical application scenarios such as terrain surveying and disaster search and rescue.

Keywords: UAV; 3D path planning; improved dhole optimization algorithm; chaotic mapping; dynamic chaotic perturbation mechanism; adaptive phase segmentation strategy

Check for updates

Academic Editors: Mostafa Hassanalian and James Ferris Whidborne

Received: 9 April 2026
Revised: 7 June 2026
Accepted: 5 August 2026
Published: 7 August 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

The objective of three-dimensional path planning [1] is to generate an optimal or feasible path from a start point to a destination, whilst considering terrain, obstacles and aircraft performance constraints. Such paths are typically required to be short in length, highly safe and smooth in flight.

Traditional path planning methods, such as A* [2] and the artificial potential field algorithm [3], are widely used; however, classic graph-search methods are commonly built on diverse network topologies including square grids and triangulation networks; their basic implementations are sensitive to network density and suffer from low efficiency in dynamic and high-dimensional environments. While advanced graph-search variants and non-grid displacement graphs have been developed to improve efficiency, they still often yield paths that are insufficiently smooth and typically require additional post-processing. Rapidly Expanding Random Trees (RRT) [4] are representative sampling-based methods. Essentially, graph-search methods generate macroscopic flight plans rather than executable flight trajectories, and neither graph-search methods nor basic RRT take UAV aerodynamic characteristics and maneuverability into account. The basic RRT generates non-smooth trajectories and lacks inherent consideration of UAV maneuverability, requiring further refinement for practical applications. Algorithms such as the Particle Swarm Optimization (PSO) [5], Genetic Algorithm (GA) [6], Ant Colony Optimization (ACO) [7], Grey Wolf Optimization (GWO) [8], and Whale Optimization Algorithm (WOA) [9] have been successfully applied to UAV path planning. However, with the growing demand for real-time decision-making, obstacle avoidance and energy efficiency, traditional intelligent optimization methods have gradually revealed their limitations when dealing with high-dimensional, non-linear and dynamic optimization problems.

Overall, existing UAV path planning methods can be categorised into graph-search-based methods, sampling-based methods, artificial potential field-based methods, swarm intelligence-based methods, and learning-based methods. Graph-search-based methods feature a well-defined search mechanism and are capable of generating feasible paths in discrete environments. However, their search efficiency and memory consumption are affected by grid resolution, particularly in three-dimensional environments characterized by complex mountainous terrain and dense obstacles. Nevertheless, the generated paths are often insufficiently smooth and typically require additional post-processing. Recent research has attempted to combine RRT* with artificial potential fields to improve sampling directions and path quality in three-dimensional UAV environments. For example, Huang et al. [10] proposed the Bi-APF-RRT* algorithm by combining an improved artificial potential field with bidirectional RRT*, thereby enhancing search efficiency and path smoothness in complex three-dimensional environments.

In response to the aforementioned issues, a significant body of research has focused on improvement mechanisms for swarm intelligence algorithms, resulting in the development of various effective strategies. Common improvement mechanisms include back-propagation, cluster-based initialization, chaotic mappings, adaptive parameter tuning, and multi-strategy fusion. These improvement methods have been widely applied to the problem of UAV trajectory planning, yielding significant progress. Mu et al. [11], addressing the tendency of traditional meta-heuristic algorithms to become trapped in local optima, introduced chaotic mapping and back-propagation strategies to propose an improved swarm intelligence optimization algorithm. This effectively enhanced population diversity and strengthened global search capabilities, significantly improving the accuracy and stability of solutions in three-dimensional path planning; Feng et al. [12] integrated the Kent chaotic map with the particle swarm optimization mechanism into the grey wolf optimization algorithm, proposing the CPS-GWO algorithm. By optimizing population initialization and position update strategies, they achieved a dynamic balance between global exploration and local exploitation capabilities, thereby obtaining shorter and smoother UAV trajectories in complex urban environments; Li et al. [13] introduced chaotic initialization, competitive behavior mechanisms and differential evolution strategies into the Sparrow Search Algorithm, proposing an improved algorithm applied to three-dimensional path

planning for multiple UAVs, which effectively improved convergence speed and enhanced obstacle avoidance capabilities; Deng et al. [14] improved the Grey Wolf Optimization Algorithm by integrating differential evolution, chaotic perturbation and adaptive weighting strategies, proposing a hybrid optimization model that achieved superior search accuracy and robustness in multi-UAV path planning problems. By introducing chaotic mechanisms, adaptive strategies and multi-algorithm fusion, these studies have effectively addressed the issues of local optima and convergence performance in traditional swarm intelligence algorithms for UAV trajectory planning. However, problems such as high parameter sensitivity, an inadequate balance between exploration and exploitation, and limited stability in high-dimensional complex environments remain prevalent.

The Dhole Optimization Algorithm (DOA) [15] is a novel meta-heuristic algorithm proposed in 2025, inspired by the cooperative hunting behavior of dhole packs. The algorithm offers advantages such as few parameter settings, a simple structure, and relatively fast convergence. However, when faced with complex multi-constraint optimization problems, particularly for UAV navigation in challenging 3D environments like mountainous or hilly terrains, DOA is prone to getting stuck in local optima, and the quality of the generated paths is unstable. The improved Dhole Optimization Algorithm proposed in this paper builds upon the DOA by first introducing a logistic chaotic map for population initialization, thereby enhancing the exploration and diversity of solutions. Subsequently, a dynamic chaotic perturbation mechanism is incorporated into the iterative process to strengthen global search capabilities and prevent the algorithm from becoming trapped in local optima. Furthermore, an adaptive stage-division strategy is designed to dynamically adjust the exploration-to-exploitation ratio based on the iteration progress, thereby achieving a balanced optimization of the search process. This paper specifically applies and validates the algorithm to address the three-dimensional path planning problem for quadrotor UAVs operating in complex hilly or mountainous environments. These environments demand that planned paths accommodate typical UAV flight maneuvers such as climbing, descending, and smooth turning to navigate uneven terrain.

Although the above methods have achieved considerable progress, several limitations remain in UAV three-dimensional path planning for terrain-rich environments. Traditional graph-search and sampling-based methods may suffer from high computational costs or insufficient path smoothness in complex three-dimensional mountainous environments. Classical swarm intelligence algorithms are prone to premature convergence and may fail to maintain a stable balance between global exploration and local exploitation, learning-based methods, such as RL and DRL, show strong adaptability in dynamic environments, but they usually require large-scale training data and high computational resources. Therefore, there is still a need to develop an optimization algorithm that can maintain population diversity, enhance global search capability, and generate safe and smooth paths under the specific constraints of 3D terrain navigation, such as in hilly or mountainous regions.

To address these issues, this paper proposes an Improved Dhole Optimization Algorithm (IDOA) for UAV three-dimensional path planning in complex terrain. The main contributions of this study are as follows. First, logistic chaotic mapping is introduced into the population initialization stage to improve the diversity and distribution quality of candidate solutions. Second, a dynamic chaotic perturbation mechanism is embedded into the iterative update process to enhance global exploration and reduce the probability of falling into local optima. Third, an adaptive stage-division strategy is designed to dynamically adjust the relationship between exploration and exploitation during different optimization stages. Finally, the algorithm's performance is rigorously tested and validated through simulations in a synthetically generated complex hilly terrain environment. A multi-objective fitness function considering path length, safety, and smoothness is con-

structed, and comparative experiments with DOA, GA, ACO, and other representative methods are conducted to verify the effectiveness of the proposed algorithm in this focused application context. To illustrate the overall logic clearly, the high-level framework of the proposed IDOA is shown in Figure 1.

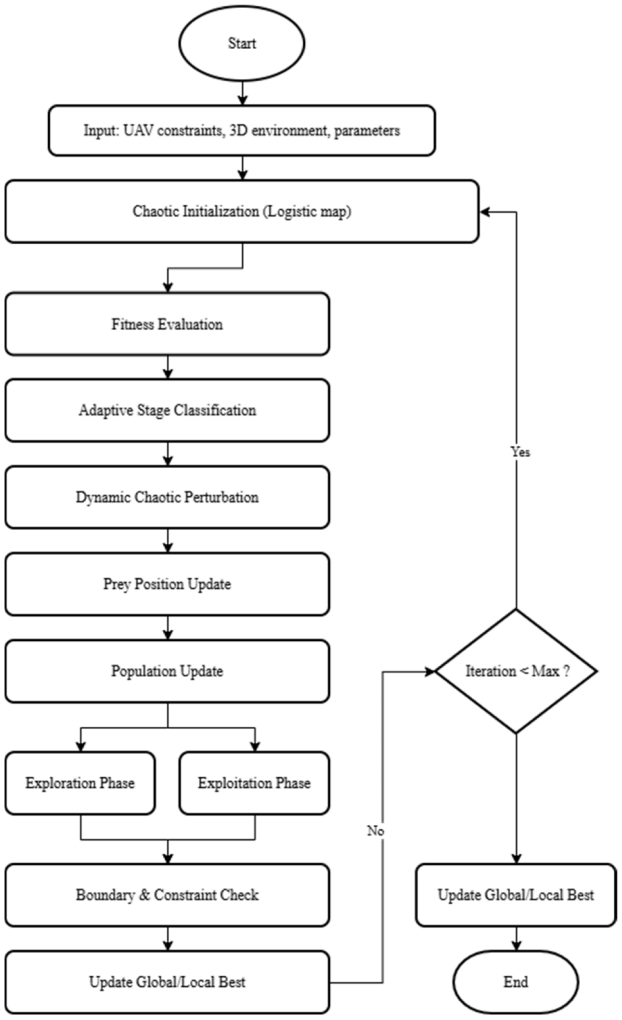


Figure 1. High-level framework of the proposed IDOA.

2. Modeling

2.1. Layer Environmental Model

The simulation environment in this paper consists of terrain formed by the superposition of multiple Gaussian peaks, with all peaks treated as obstacles. The peaks are divided into two categories: major peaks and minor hills. Major peaks range in height from 30 to 60 m, whilst minor hills range from 10 to 25 m; an automatic retry mechanism ensures that no peak exceeds 60 m. Each peak is defined by its center coordinates $\left(\mathrm{x}_j, \mathrm{y}_j\right)$, height h_j and radius r_j. There are $N_o \in [5, 8]$ main peaks, which are randomly distributed with a height of $h_j \in [30, 60]$ m and a radius of $r_j \in [8, 15]$ m. There are $\mathrm{N}_h = 3$ small hills, each with a height of $A_k \in$ [10,25] m and a radius of $\sigma_{\mathrm{x,k}}, \sigma_{\mathrm{y,k}} \in$ [6,11] m. To ensure that obstacles are distributed evenly, the center-to-center distance between adjacent major peaks is no less than 25 m, and all peaks avoid the 20 m area surrounding the start and finish lines. The peaks are defined as shown in Formula (1):

$$Z(x,y) = \sum_{k=1}^{N_0} A_k \exp\left[-\left(\frac{x-x_k}{\sigma_{x,k}}\right)^2 - \left(\frac{y-y_k}{\sigma_{y,k}}\right)^2\right] + \sum_{k=1}^{N_0} h_j \exp\left[-\left(\frac{x-x_j}{r_j}\right)^2 - \left(\frac{y-y_j}{r_j}\right)^2\right] \quad (1)$$

Here, $\sum_{k=1}^{N_0} A_k \exp\left[-\left(\frac{x-x_k}{\sigma_{x,k}}\right)^2 - \left(\frac{y-y_k}{\sigma_{y,k}}\right)^2\right]$ represents randomly generated small hills; there are $N_h = 3$ of them, with A_k heights ranging from 10 to 25 m, and their centres are randomly distributed whilst avoiding the main peak areas. $\sum_{k=1}^{N_0} h_j \exp\left[-\left(\frac{x-x_j}{r_j}\right)^2 - \left(\frac{y-y_j}{r_j}\right)^2\right]$ represents the main peaks, with a quantity of N_0 ranging from 5 to 8, a height of h_j between 30 and 60 m, and a radius of r_j between 8 and 15 m. During the generation process, if the height at any point exceeds 60 m after superimposition, all peaks are regenerated to ensure that the summit of each peak meets the height requirement. In the 3D environment visualization, all peaks are treated as obstacles. To intuitively display the position and extent of these obstacles, each peak is represented in the graphic as a grey cylinder, with its height and radius determined by the parameters described above. Figure 2 shows the generated 3D environment.

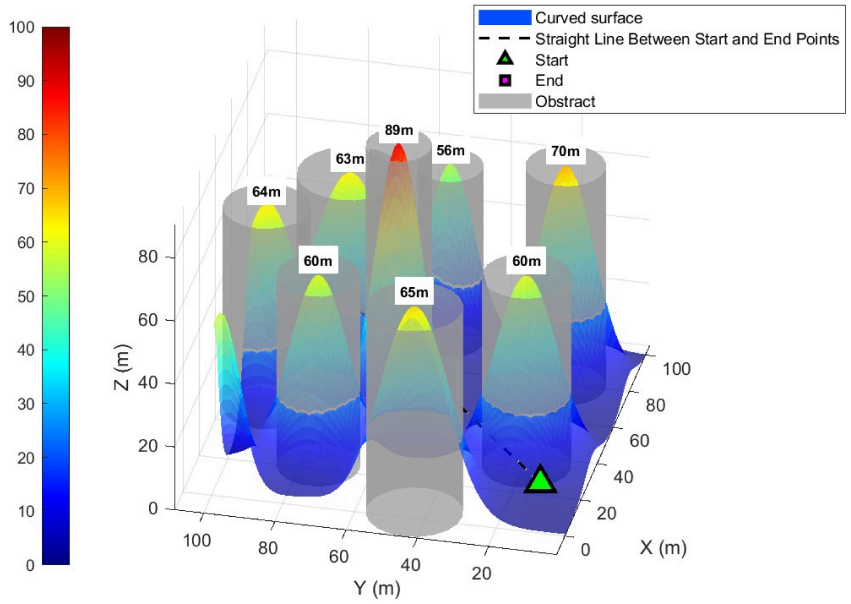


Figure 2. The generated 3D environment.

Remark. This work focuses on 3D path planning algorithm evaluation in simulated mountainous/hilly terrains. Obstacles are modeled as Gaussian-based cylinders to construct a standardized, reproducible testbed. This simplified geometric representation provides a sufficiently complex three-dimensional search space for objective performance comparison of the algorithm. The 10 m minimum safety altitude provides a buffer against terrain undulations. It is important to note that the proposed path planning framework is general and can be adapted to incorporate other obstacle models (e.g., rectangular prisms for urban environments) without loss of functionality.

The grey cylinders in the image represent randomly generated main peaks and smaller hills; the green triangles serve as starting points, and the magenta squares as endpoints; the terrain background is composed of all the peaks together.

2.2. Mathematical Model of the Task Allocation Layer

The fitness function F takes into account path length, safety and smoothness, as shown in Equation (2):

$$F = w_1 f_{len} + w_2 f_{safe} + w_3 f_{smooth} + p_{loop} \tag{2}$$

Take $w_1 = 0.3, w_2 = 0.4, w_3 = 0.3$. The weights w_1, w_2, and w_3 are non-negative real numbers with a sum constrained to unity, i.e., $w_1 + w_2 + w_3 = 1$.

This weighting ensures that the fitness function represents a balanced overall cost, where the contribution of each term is properly scaled.

In the context of offline 3D path planning in mountainous or hilly terrain, the specific weights ($w_1 = 0.3, w_2 = 0.4, w_3 = 0.3$) were determined through systematic parameter tuning. The primary goal was to identify a weighting set that yields safe, reasonably short, and smooth paths—essential attributes for UAV navigation in uneven environments. A higher emphasis is placed on safety ($w_2 = 0.4$) to prioritize obstacle clearance in complex terrain. While these weights are fixed in the present study to ensure a consistent comparison of algorithm performance, the formulation itself is general: operators can adjust the weights according to different mission priorities—for example, increasing w_1 for energy-aware operations or raising w_2 for safety-critical missions.

2.2.1. Path Length Cost

Path length cost f_{len} is used to evaluate the economic viability and efficiency of a path, thereby preventing unreasonable situations such as excessively long routes or detours. A path consists of m interpolation points $P_1, \ldots, P_m$; the path length is defined as the sum of the Euclidean distances between adjacent points, as expressed by Equation (3):

$$f_{len} = \sum_{i=1}^{m-1} \|p_{i+1} - p_i\| \tag{3}$$

2.2.2. The Cost of Security

The safety Cost f_{safe} is designed to quantify the collision risk between the planned path and the terrain or obstacles in the environment. It is computed as the average of penalty values assessed at each discretized waypoint along the path. For the ith path point $p_i = (x_i, y_i, z_i)$, the terrain elevation $Z_{terrain}(x_i, y_i)$ at that point is obtained via interpolation, and the safety margin $\delta_i = z_i - z_{terrain}(x_i, y_i)$ relative to the terrain is defined. The safety margin threshold is $h_{safe} = 10$ m, which is the minimum safe altitude to avoid terrain fluctuations and low obstacles in mountainous environments. Flying below this threshold is considered risky and incurs a penalty. The safety cost comprises three components: terrain safety penalty, elevation penalty and obstacle penalty.

(1) Terrain safety penalty

If $\delta_i < h_{safe}$, the path point is below safe height, a severe penalty is incurred, as shown in Equation (4):

$$p_{terrain,i} = 500(h_{safe} - \delta_i)^2 \tag{4}$$

If $h_{safe} \leq \delta_i < 1.2h_{safe}$ slightly below the expected height, a mild penalty is imposed, as shown in Equation (5)

$$p_{terrain,i} = 10(1.2h_{safe} - \delta_i)^2 \tag{5}$$

Otherwise, there is no terrain penalty.

(2) Severe punishment

To prevent unnecessary climbing, the target altitude is set to $h_{desired} = z_{terrain} + 1.5h_{safe}$. If the altitude at a waypoint exceeds the target altitude, a penalty proportional to the square of the excess is applied, as shown in Equation (6):

$$p_{height,i} = 20(z_i - h_{desired})^2 \tag{6}$$

(3) Obstacle Penalty

Consider an obstacle j centered at point (x_j, y_j), with radius r_j and height h_j. Calculate the horizontal distance $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ between the path point and the center of the obstacle. If $d_{ij} < r_j$ and $z_i < h_j + h_{safe}$ hold, i.e., the path point is directly above the obstacle and the height is insufficient, then a collision penalty is applied, as shown in Equation (7):

$$p_{obs,i,j} = 1500(h_j + h_{safe} - z_i)^2 \tag{7}$$

If $d_{ij} < r_j$ but $z_i \geq h_j + h_{safe}$, i.e., if the aircraft flies over the obstacle, a penalty related to the degree of proximity to the horizontal plane is imposed, as shown in Equation (8):

$$p_{obs,i,j} = 100(1 - \frac{d_{ij}}{r_j})^2 \tag{8}$$

If $r_j \leq d_{ij} \leq r_j + 3$, if this waypoint is located within 3 m of the obstacle boundary, a proximity penalty is applied based on the distance from the boundary, as shown in Equation (9):

$$p_{obs,i,j} = 50(1 - \frac{d_{ij} - r_j}{3})^2 \tag{9}$$

If a path point lies outside the map boundaries, a weight penalty $p_{boundary,i} = 10,000$ is applied directly. In summary, the safety cost is the average of the sum of the penalty terms for all path points, as shown in Equation (10):

$$f_{safe} = \frac{1}{m}\sum_{i=1}^{m}\left(p_{terrain,i} + p_{height,i} + \sum_j p_{obs,i,j} + p_{boundary,i}\right) \tag{10}$$

In the formula, A represents the total number of waypoints (set to M). The lower the cost, the safer the path as a whole. The constants in Equations (4)–(10) adopt a risk-graded penalty strategy [16], which are calibrated via repeated simulations to ensure numerical stability and balanced penalty intensity.

Parameter Note: The safety altitude threshold h_{safe} and the associated penalty coefficients are user-defined parameters. Their values should be configured according to the specific mission requirements and operational environment. For instance, in missions that mandate low-altitude flight (e.g., urban package delivery or terrain-following inspection), h_{safe} can be set to a minimal value (e.g., zero), thereby effectively removing the altitude clearance penalty and focusing the safety cost purely on obstacle collision and proximity avoidance. In this study, focused on mountainous terrain navigation, h_{safe} is set to 10 to ensure a safe distance from uneven ground.

2.2.3. Smoothing Cost

The smoothness cost f_{smooth} reflects the smoothness of the path and directly affects the quality of the UAV's path and the difficulty of control. f_{smooth} is calculated using the angle between adjacent line segments. For the i-th path point p_i, the backward vector $\vec{v}_{i-1}$ connects the previous point p_{i-1} to p_i, and the forward vector $\vec{v}_i$ connects p_i to the next point p_{i+1}, defined as Equations (11) and (12):

$$\vec{v}_{i-1} = p_i - p_{i-1} \tag{11}$$

$$\vec{v}_i = p_{i+1} - p_i \tag{12}$$

Let $\vec{v}_{i-1}$ and $\vec{v}_i$ be normalized unit vectors, and $\theta_i = arc\cos(\vec{v}_{i-1}, \vec{v}_i)$ be the angle between them. $f_{smooth,angle} = \sum_{i=2}^{m-1} \theta_i^2$ is the core term in the smoothness cost calculation. A smaller value of $\theta_i = arc\cos(\vec{v}_{i-1}, \vec{v}_i)$ indicates a smaller turning radius. In practice, to emphasize the impact of sharp turns, each θ_i^2 is multiplied by an scaling factor of 50 which is widely used in UAV smoothness optimization [17,18] and validated as optimal for balancing smoothness and path length [19]. That is, the cumulative term of the smoothness cost in the code is denoted as $50 \cdot \theta_i^2$, and after summation, it is normalized by dividing by the number of interior path points $m-2$ (excluding start and end points), The smoothness cost is given by Equation (13):

$$f_{smooth} = \frac{1}{m-2} \sum_{i=2}^{m-1} (50 \cdot \theta_i^2) \tag{13}$$

2.2.4. Restrictions and Additional Penalties

To improve the feasibility and quality of the path, a constraint on the spacing between control points is introduced, and a looping penalty is incorporated into the fitness function as an additional penalty term. Together with the aforementioned length, safety and smoothness costs, these form the total fitness, ensuring that the generated path satisfies the UAV's dynamic characteristics and environmental constraints. The Euclidean distance between adjacent control points must be no less than $d_{\min} = 5$ m. The 5 m threshold is a standard setting in UAV B-spline path planning, widely adopted in complex mountainous environments. It balances path smoothness, maneuverability, and computational efficiency. If the distance is less than $d_{\min}$, the control points are stretched along the connecting line to $d_{\min}$. If $f_{len}/D_{straight} > R_{\max}$ (where $R_{\max} = 2$ is taken), the path is deemed to be a looping path. Alternatively, the number of times the angle $\theta_i > 120^\circ$ between adjacent line segments occurs is counted; if this count exceeds 10% of the total number of path points, the path is deemed to be a looping path. Once a path is determined to be a looping path, a penalty term must be added to the fitness function, as shown in Equation (14):

$$P_{loop} = 1000 \times \left(\frac{L}{D_{straight}} - 1\right) \tag{14}$$

where A is the path length and B is the straight-line distance between the start and end points.

Circular behaviour (local looping or backtracking) is penalized because it increases flight distance and energy consumption, reduces path directivity, and raises the risk of repeated obstacle proximity.

2.3. Path Representation via Cubic B-Spline

To simplify the optimization process and ensure a smooth path, a path representation based on control points is employed. The path is guided by n_c control points $C_1, C_2, C_3 \cdots C_{n_c}$, with the start point P_s and end point P_g fixed. Performing cubic B-spline interpolation on the control points yields a sequence of path points, as shown in Equation (15):

$$P(u) = \sum_{i=0}^{n_c} N_{i,3}(u) C_i \tag{15}$$

Here, $N_{i,3}(u)$ denotes a cubic B-spline basis function, defined using deBoor's [20] recursive formula; the node vectors are uniformly distributed; $u \in [0,1]$ denotes the number of interpolation points; and $m = 100$ denotes the number of interpolation points.

By performing equidistant sampling on u, a sequence of path points $P_1, \ldots, P_m$ is obtained. As shown in Figure 3.

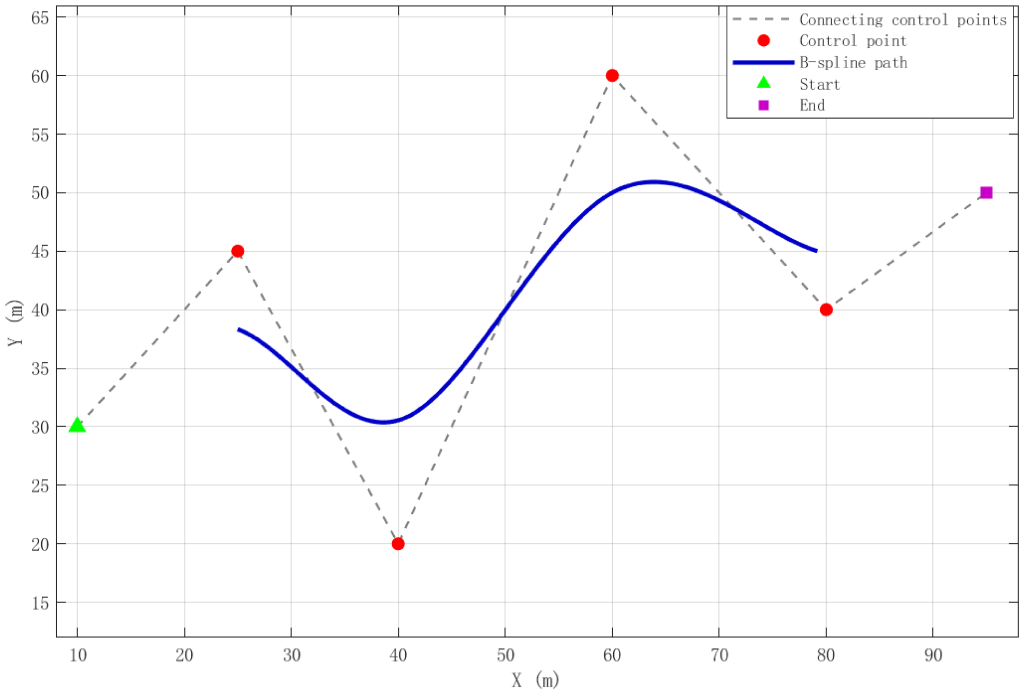


Figure 3. Chematic diagram of control points and cubic B-spline paths.

Note: Control points are auxiliary nodes for cubic B-spline interpolation and are not actual flight points; waypoints refer to discrete points on the final smooth trajectory that the UAV follows during flight. The cubic B-spline path is pre-computed offline before flight according to the 3D environment and control points, and the UAV executes the trajectory during flight.

3. The Improved Dhole Optimization Algorithm (IDOA)

This section presents the proposed Improved Dhole Optimization Algorithm (IDOA). The overall workflow of the algorithm is illustrated in Figure 4, which includes initialization, fitness evaluation, iterative optimization, and solution output modules. The following subsections will elaborate on each part in detail. In the figure, the large dashed box encloses the algorithm's core iterative optimization loop. The dashed arrows represent the iterative feedback process. The solid arrows, meanwhile, represent the algorithm's main forward execution path, illustrating the primary sequence of operations from initialization to output.

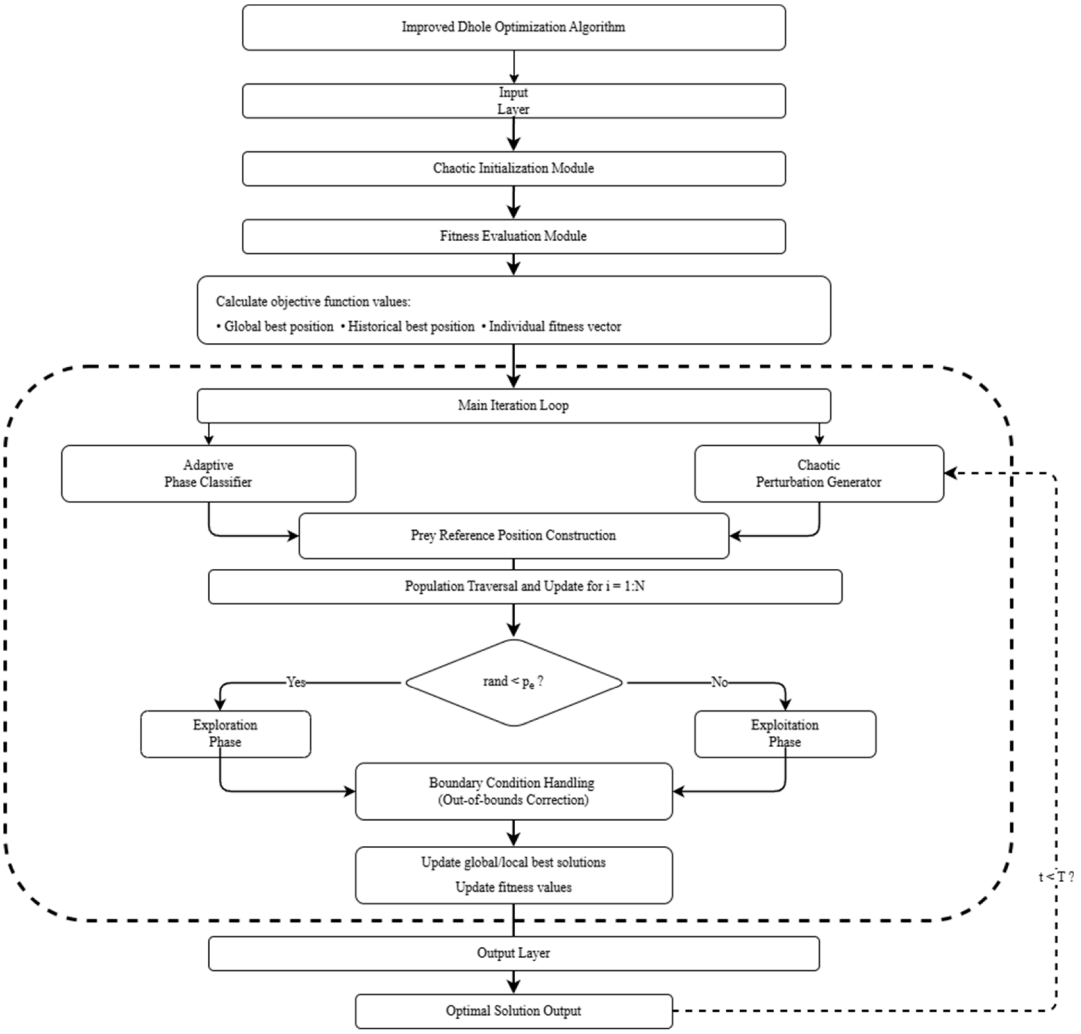


Figure 4. Conceptual framework of the proposed UAV 3D path planning method.

3.1. DOA

The core concept of the Dhole Optimization Algorithm (DOA) is derived from the cooperative hunting behavior of Dhole packs. The entire search and optimization process can be viewed as a series of key steps—such as encircling and attacking the prey—carried out by the pack, with the aim of capturing the target achieved through the continuous adjustment of each individual Dhole's position. As shown in the flowchart, during the optimization process, each Dhole in the population represents a potential solution to the problem, with the corresponding decision variable values quantified by the position coordinates of everyone within the search space. When dealing with a problem involving m decision variables and setting the population size to N, the initial distribution of the Dholes' positions can be represented by an N-row by m-column matrix, as shown in Equation (16).

$$D = \begin{bmatrix} D_1 \\ \vdots \\ D_t \\ \vdots \\ D_N \end{bmatrix}_{N \cdot m} = \begin{bmatrix} d_{1,1} & \cdots & d_{1,j} & \cdots & d_{1,m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ d_{t,1} & \cdots & d_{t,j} & \cdots & d_{t,m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ d_{N,1} & \cdots & d_{N,J} & \cdots & d_{N,m} \end{bmatrix} \quad (16)$$

Each row of this matrix represents the position vector of a single Dhole; specifically, the ith row vector $D_i = [d_{i,1}, d_{i,2} \ldots d_{i,j} \ldots d_{i,m}]$ denotes the ith candidate solution; each

element $d_{i,j}$ represents the value of that solution in the ith dimension. The objective function $F(D_i)$ is introduced to evaluate the quality of each Dhole's position, serving as a metric for assessing the fitness of that candidate solution D_i. The fitness values of all individuals form an N-dimensional column vector, as shown in Equation (17):

$$\begin{bmatrix} F_1 \\ \vdots \\ F_t \\ \vdots \\ F_N \end{bmatrix}_{N \cdot 1} = \begin{bmatrix} F(D_1) \\ \vdots \\ F(D_t) \\ \vdots \\ F(D_N) \end{bmatrix}_{N \cdot 1} \tag{17}$$

The DOA does not use a fixed population size. Instead, it adopts a dynamic population mechanism, in which the number of individuals participating in the search is denoted as N_{pop}. This dynamic population size is randomly determined during initialization or iteration, as shown in Equation (18).

$$N_{pop} = round(rand \times 15 + 5) \tag{18}$$

This formula generates random values within the range of 5 to 20, thereby maintaining the fundamental scale characteristics of collaborative hunting by Dhole packs, whilst also introducing random fluctuations in pack size into the algorithm, which helps to enhance population diversity. The random range of 5 to 20 follows the original DOA design, simulating natural group size to balance population diversity and convergence speed. In this work, the optimization target corresponds to the fixed start and end points of the UAV path, which are static. In the algorithmic model, x is a random number uniformly distributed between 0 and 1, used to generate the random variables required for subsequent steps. When simulating the hunting behaviors of Dholes, the DOA focuses primarily on their ability to capture prey of comparable or even larger size, such as deer and wild boar. Such successful hunts, which cannot be achieved through individual effort alone, are primarily due to the Dhole pack's highly coordinated collective hunting strategy. In terms of timing, dhole hunting activities do not occur around the clock but are primarily concentrated during dawn and dusk, when prey activity is most frequent. This is mainly because predators and prey share similar circadian rhythms. Based on this biological observation, the algorithm incorporates an 'optimal hunting time' parameter ps to quantify the suitability of hunting opportunities under different conditions; its expression is randomly determined as shown in Equation (19).

$$ps = \left(\frac{C1}{(1 + exp(-k(PMN - \mu))}\right)^2 \times EF \tag{19}$$

In this formula: μ represents the optimal group size for cooperative hunting; EF is an environmental factor, taking values between 0 and 1, used to reflect the influence of the external environment on hunting success; k is the hunting efficiency coefficient, representing the level of proficiency in group coordination; $C1$ is a regulatory factor used to adaptively adjust the hunting time according to the size of the prey. In the DOA, the iterative process of determining the Dholes' positions is at the core of the algorithm; this process simulates the complete behavioral logic of a Dhole pack in the wild, from searching for prey to capturing it. The mathematical model is established for the search phase, the encirclement phase and the attack phase.

3.1.1. Search Phase

This stage aims to locate the target prey and simulates the reconnaissance behaviors of a pack of Dholes prior to a hunt. The position of the initial target is determined as shown in Equation (20)

$$prey = \frac{prv_{N\max} + prv_{Syl\max}}{2} \tag{20}$$

Here, $prv_{N\max}$ represents the global optimal solution in the current population, whilst $prv_{Syl\max}$ represents the optimal solution recorded during the historical iteration cycle. The global optimal solution $prv_{N\max}$ refers to the individual with the minimum fitness value among the current population, evaluated by the fitness function in Section 2.2. The search process iteratively refines the population of candidate paths. The iterative process continues until a predefined maximum number of iterations T is reached. The value of T is set empirically to balance solution quality and computational cost. While metaheuristic algorithms like the proposed IDOA do not provide mathematical guarantees of convergence to a global optimum in complex, non-convex search spaces, they are designed to achieve practical convergence—finding high-quality, feasible solutions within a reasonable time. In our experiments, the algorithm typically exhibits rapid fitness improvement in early iterations, followed by stabilization. Subsequent experimental results illustrates the convergence profiles of the best-found fitness value for representative test cases, demonstrating this trend.

During the iteration process, if the population size decreases to less than half of the initial number for any reason, the algorithm will automatically adjust the search strategy by expanding the search range to increase population diversity and avoid prematurely becoming trapped in a local optimum. To achieve this, the algorithm updates the positions of individuals using a perturbation term with a ratio of 0.5, in accordance with Equation (21).

$$D_{ij}^{s-1} = d_{ij} + C_2 \times rand \times (prv_{ij} - d_{ij}) \tag{21}$$

In Equation (21), *rand* is a random number uniformly distributed over the interval [0, 1], used to provide random perturbations for the position update; C_2 is a parameter that decreases with the number of iterations, as shown in Equation (22).

$$C_2 = 1 - \frac{1}{T} \tag{22}$$

In the formula, T represents the maximum number of iterations set by the algorithm. The coefficient C_2 linearly decreases from 1.0 to 0.2 over iterations, balancing global exploration and local exploitation. As the iterations progress, C_2 gradually decreases, thereby enabling the algorithm to maintain a strong global search capability in the early stages of the search, whilst placing greater emphasis on local exploration in the later stages. This dynamic adjustment method enhances the global search capability of the search process whilst, to a certain extent, accelerating the convergence rate towards the optimal solution.

For practical deployment considerations, if a satisfactory solution is not obtained within the allotted iteration budget T, standard contingency strategies can be employed. These include re-initializing the algorithm with different random seeds, temporarily relaxing certain kinematic constraints (e.g., minimum turning radius), deterministic fallback planner to ensure a safe, feasible path is always available.

3.1.2. Encirclement Phase

If the pack of Dholes successfully locates its prey, it will proceed to the encirclement phase. When the current pack size exceeds 10 and the communication strength between

individuals is less than 0.5, the positions of the individuals are dynamically adjusted according to Equation (23) to enhance pack coordination

$$D_{ij}^{s-1} = d_{ij} - d_{zj} + prey_j \tag{23}$$

Here, $prey_j$ denotes the position of the current prey in the jth dimension, d_{ij} denotes the current position of the ith individual in the jth dimension, and d_{zj} denotes the position of another individual, selected at random from the population, in the jth dimension. To ensure the randomness of interactions within the population, the algorithm uses Equation (24) to generate a random integer index z, ranging from 1 to N, which is used to randomly select an individual from the population, thereby simulating interactions between individuals.

$$z = round(rand \times (N-1)) + 1 \tag{24}$$

For UAV applications, the encirclement behavior is mapped to collaborative refinement of candidate paths around the current optimal trajectory, improving path smoothness and safety. This work addresses single-UAV path planning; the DOA population represents parallel candidate paths, not physical UAVs, so no inter-UAV communication or coordination is required.

3.1.3. Attack Phase

Following the encircling and search phases, the algorithm enters the attack phase, which represents an intensive local search and refinement process focused on the most promising candidate solutions (metaphorically, the "prey"). The objective is to fine-tune these solutions to achieve higher precision and potentially escape shallow local optima.

Once the pack of Dholes has surrounded its target prey, it enters the attack phase, which simulates the final, coordinated strike. In the algorithmic context, this phase is triggered when the quality of the current best solution (evaluated by its fitness value) is sufficiently dominant, metaphorically corresponding to a "signal strength" exceeding a threshold (e.g., 0.5). The attack strategy is adaptively determined based on the relative difficulty of capturing the "prey," quantified by the prey size parameter S, as defined in Equation (25):

$$S = C_p + rand \times \left(\frac{Hwma}{Hwcm_{pop}}\right) \tag{25}$$

In this formula, C_p represents the prey factor, which is set to the constant 3 following the original DOA design; $Hwma$ and $Hwcm_{pop}$ denote the fitness values of the current individual and the current target, respectively. The threshold $S > 2$ is derived from the original DOA as an empirical value ($C_p/1.5 = 2$), which distinguishes large prey requiring staged weakening from small prey that can be captured directly. If the prey size parameter S satisfies condition $S > 2$, this indicates that the target is large and will require multiple stages of iterative refinement to weaken it, as shown in Equations (26) and (27).

$$W_{prop} = \exp\left(-\frac{1}{S}\right) \times prv_{N\max} \tag{26}$$

$$\begin{aligned} &D_{ij}^{s-1} = d_{ij} + W_{prop} + \\ &ps \times \left[ln(2 + x + rand) - ln(2 + x + rand) \times W_{prop} + ps\right] \end{aligned} \tag{27}$$

Conversely, if condition A ($S \leq 2$) holds, the pack of Dholes will immediately hunt down the prey; the position update is shown in Equation (28).

$$D_{ij}^{s-1} = (d_{ij} - prey_{Syl\max}) \times ps + ps \times rand \times d_{ij} \tag{28}$$

3.2. Improvements to the DOA

The DOA operates randomly during the population initialization phase, resulting in an uneven distribution of initial solutions, which in turn adversely affects subsequent convergence rates and solution accuracy. To address this, this paper introduces chaos theory to improve the DOA, proposing the Improved Dhole Optimization Algorithm (IDOA), which utilizes the logistic chaotic map to generate the initial population, thereby enhancing the quality and diversity of the initial solutions. Dynamic chaotic perturbations are incorporated into the position update process to enhance the algorithm's global search capability. A strategy for an adaptive partitioning phase is proposed, dynamically adjusting the ratio of exploration to exploitation to achieve a better balance during the iterative process. The algorithm flow is shown in Figure 5.

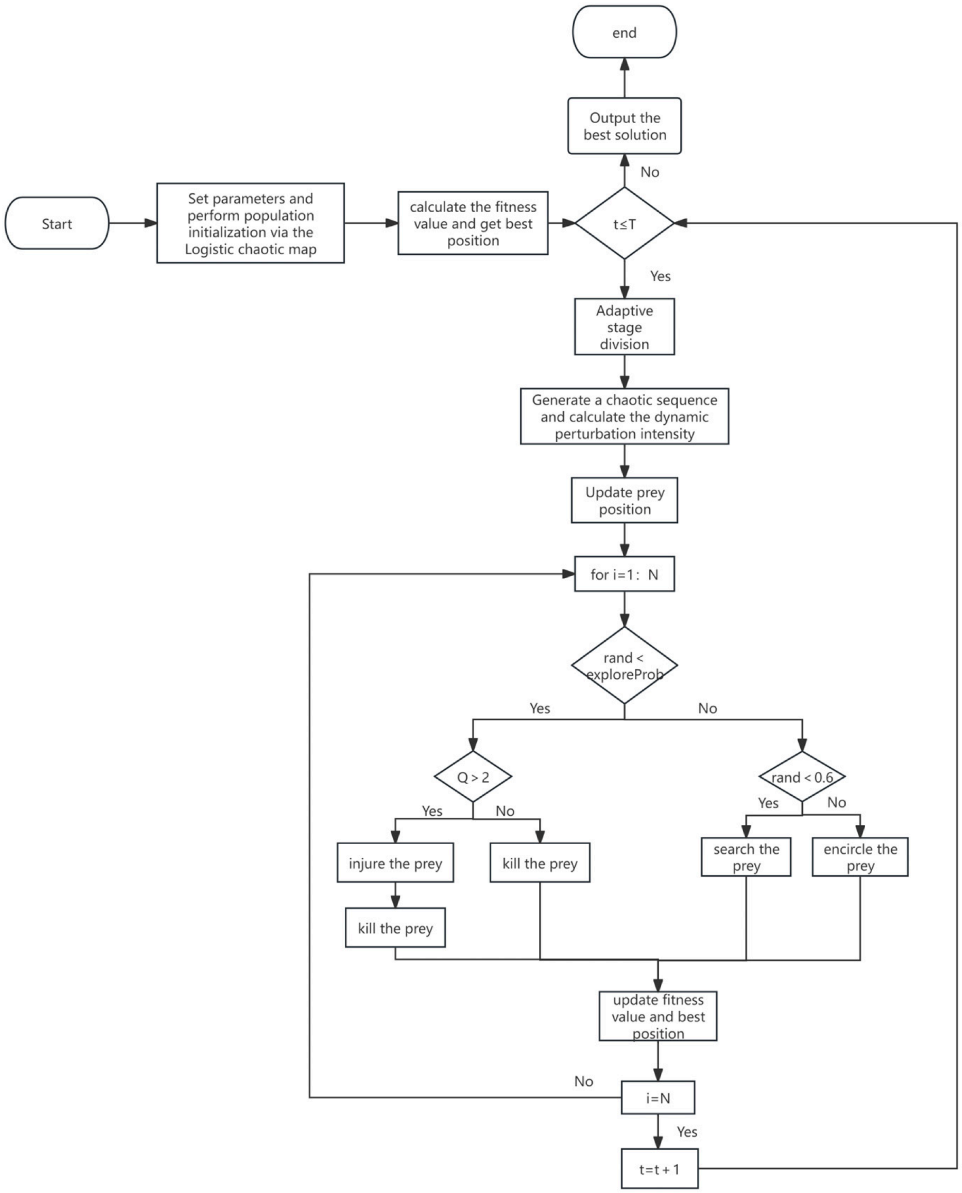


Figure 5. Flow chart of improved dhole optimization algorithm.

3.2.1. Chaos-Based Enhancement Strategy

Chaos is a form of motion in non-linear systems that appears random but is internally deterministic; it is characterized by properties such as ergodicity, sensitivity to initial conditions and pseudo randomness. It has found widespread application in the refinement

of meta-heuristic algorithms. Commonly used chaotic mappings include the logistic map and the tent map [21]. In this paper, the logistic map [22] is adopted in the IDOA, as defined in Equation (29).

$$z_{k+1} = \mu z_k (1 - z_k) \tag{29}$$

These are the dynamical control parameters of System. In this work, the parameter μ is set to 4. At this value, the logistic map operates in a fully chaotic regime, as defined by Devaney's criteria: the generated sequence z_k is topologically transitive, has dense periodic points, and exhibits sensitive dependence on initial conditions. The resulting sequences are non-repeating, ergodic, and uniformly distributed over (0, 1), as confirmed by the invariant density $f(z) = \frac{1}{\pi\sqrt{z(1-z)}}$. This property ensures maximal population diversity in the initial search space, preventing premature convergence and enhancing global exploration.

When $\mu = 4$, the system exhibits fully chaotic behaviors, generating sequences that traverse the interval (0, 1) without repetition. The logistic map operation is computationally lightweight; benchmark implementations on embedded platforms demonstrate execution times under 100 nanoseconds per iteration [23,24], making its one-time use for population initialization negligible within the overall offline planning budget. This allows for the simulation of randomness whilst maintaining a deterministic trajectory, which is used for population initialization and dynamic perturbation. First, generate N d-dimensional chaotic vectors Z_i^N, each generated by iterating the logistic map using differentiated initial values. Discard the first 500 iterations to eliminate transient effects and ensure the sequences exhibit chaotic characteristics, and map the chaotic vectors to the search space, as shown in Equation (30).

$$X_{i,j} = L_j + (U_j - L_j) \cdot Z_{i,j} \tag{30}$$

Here, $j = 1, 2, 3 \cdots d$ denotes the dimension, whilst L_j and U_j represent the lower and upper bounds of the jth dimension, respectively. The chaotic perturbation only adjusts control points within the feasible search space, constrained by the 5 m minimum distance and maximum curvature limits. The final B-spline trajectory always satisfies UAV kinematic constraints and is physically executable.

3.2.2. Chaos Disturbance Update

Building upon the DOA's method for calculating the prey's position, a chaotic perturbation term is introduced to enhance the algorithm's global search capability; the improved calculation of the prey's position is shown in Equation (31).

$$\frac{X_{best} + X_{local,rand}}{2} + \alpha(t) \cdot \xi \cdot (U - L) \tag{31}$$

In this formulation, the term $\frac{X_{best}+X_{local,rand}}{2}$ guides the search using two complementary sources of information:

X_{best}: the global optimal position found so far by the entire population;

$X_{local,rand}$: a randomly selected historical best solution from previous iterations.

This combination balances guidance from the current best solution with diversity from historical local optima. The main innovation of the proposed update lies in the additional chaotic perturbation term, which is designed to dynamically adjust the algorithm's behavior over the course of the optimization process. In this term:

$\alpha(t)$ is a dynamically decaying chaotic intensity coefficient that decreases with iteration number t;

ξ is a d-dimensional random vector generated by the logistic chaotic map, which provides ergodic, non-repeating perturbations;

U and L are the upper and lower bounds of the search space, respectively, ensuring the perturbed solution remains within feasible limits.

By combining these elements, the algorithm gains controllable chaotic exploration capabilities. Specifically:

In early iterations, $\alpha(t)$ is large, leading to strong perturbations that encourage extensive exploration of the entire search space. This increases population diversity and reduces the risk of premature convergence.

As iterations progress, $\alpha(t)$ gradually decreases, weakening the chaotic perturbation and shifting the algorithm's focus to local refinement around promising regions. This allows the algorithm to fine-tune solutions and converge more precisely to high-quality optima.

This dynamic adjustment strategy effectively balances global exploration (to discover promising regions) and local exploitation (to refine solutions), thereby reducing the likelihood of stagnation in local optima and improving both the reliability and accuracy of the algorithm. Importantly, as iterations progress, the decaying coefficient $\alpha(t)$ gradually reduces the magnitude of the chaotic perturbation, ensuring that the influence of randomness diminishes over time. This design guarantees that the algorithm will not continue to diverge indefinitely, allowing the population to converge stably to high-quality solutions. Subsequent experiments empirically verified the convergence behavior, where the fitness values exhibit a stable and monotonic decrease throughout the optimization process.

3.2.3. Adaptive Stage Classification

To achieve a better balance between exploration and exploitation, the entire iterative process is divided into four stages, and the exploration probability p_{explore} and the intensity of chaotic perturbations are dynamically adjusted according to the key characteristics of each stage. As shown in Equation (32).

$$p_{\mathrm{explore}} = \begin{cases} 0.8 & t \le \frac{T}{4} \\ 0.6 & \frac{T}{4} < t \le \frac{T}{2} \\ 0.4 & \frac{T}{2} < t \le \frac{3T}{4} \\ 0.2 & t > \frac{3T}{4} \end{cases} \tag{32}$$

Here, p_{explore} represents the probability of moving to a new region in search of a better solution; a higher value indicates a greater tendency to leave the current region and explore uncharted territory, whilst a lower value indicates a greater tendency to refine the current solution. The intensity of chaotic perturbation is defined as the magnitude of the random disturbance applied to the current solution; a higher intensity results in more pronounced changes to the solution, thereby facilitating exploration, whilst a lower intensity results in more subtle changes, thereby facilitating refinement. Here, t is the current iteration number, T is the maximum number of iterations. When $t \le \frac{T}{4}$ is selected, exploration takes precedence and the intensity of chaotic perturbation is high; when $\frac{T}{4} < t \le \frac{T}{2}$ is selected, exploration and exploitation are given equal importance, and the chaotic perturbation is appropriately attenuated; when $\frac{T}{2} < t \le \frac{3T}{4}$ is selected, exploitation is the primary task and the degree of chaotic perturbation is further reduced; when $t > \frac{3T}{4}$ is selected, exploitation is performed with minimal chaotic perturbation, focusing on fine-grained local searches to improve convergence accuracy. The function in Equation (32) is continuous at the boundaries $t = \frac{T}{4}$, $t = \frac{T}{2}$, and $t = \frac{3T}{4}$, ensuring smooth stage transitions without discontinuities. The IDOA pseudocode diagram is shown in Algorithm 1.

Algorithm 1: IDOA

Input Initial parameters
Output: Optimal solution
Chaos initialization;
for $t = 2$: T;
  Calculate the current chaos intensity
  Generate Logistic chaotic vector ξ;
  Construct prey reference position:
    $prey = \frac{X_{best}+X_{L,best}}{2} + \alpha_t(2\xi - 1) \odot (ub - lb)\ x^t = r(d(x))$;
  Set exploration probability p_e;
    for $i = 1 : N$;
      if $rand < p_e$ **then** Enter the exploration stage;
        Update in the direction of prey with probability 0.6, otherwise update in the direction of the randomly selected locally optimal individual;
      else Enter the development stage;
        Calculate the attack factor Q;
      else Conduct local development and updates in the direction of prey;
      end if;
    end for;
end for;
Update solution.

3.2.4. Benchmark Function Testing and Analysis

To validate the effectiveness of IDOA, the study selected six benchmark functions—including low-dimensional, high-dimensional, unimodal and multimodal functions—for testing, and compared them with DOA, PSO and GA. The experimental parameters were set as follows: population size N = 80 and maximum number of iterations T = 1000; each algorithm was run 20 times to minimize the impact of randomness. This ensures every algorithm is allocated the same total number of objective function evaluations, forming a consistent baseline for comparison. The algorithm-specific hyperparameters listed in Table 1 are adopted from the original canonical publications or authoritative standard references for each method. This configuration allows each algorithm to operate under its intended native settings, enabling a fair comparison of the inherent search performance of different algorithms under the same computational budget.

Table 1. Algorithm parameter settings.

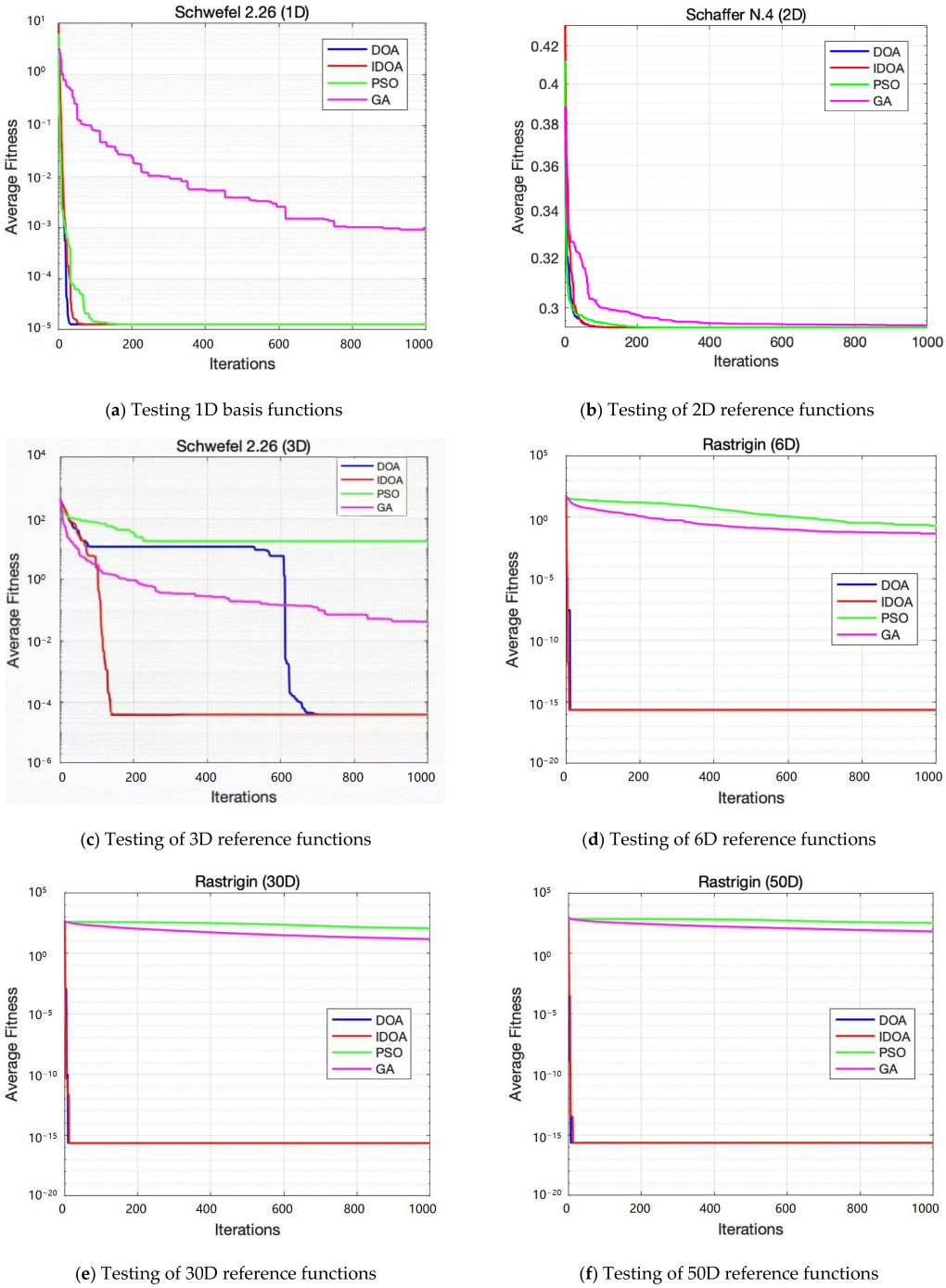
Algorithm	Parameters
DOA	No parameters specified
IDOA	Chaos intensity $\alpha_0 = 0.05$; attenuation factor $\lambda = 0.96$
PSO	Inertial weight ω decreases linearly from 0.9 to 0.4; learning rate $c_1 = 2$; learning rate $c_2 = 2$
GA	Probability of a single-point crossover: $p_c = 0.8$; probability of uniform variation: $p_m = 0.1$

The test functions include unimodal (Schwefel 2.26), multimodal (Rastrigin), and composite types with dimensions from 1D to 50D, covering typical optimization scenarios. The theoretical optimal fitness value for all functions is 0; lower values indicate higher optimization accuracy. The experimental results are shown in Table 2.

Table 2. Comparison of benchmark function test results.

Function (Dimension)	DOA	IDOA	PSO	GA
Schwefel2.26 (1D)	1.27×10^{-5}	1.27×10^{-5}	1.27×10^{-5}	9.12×10^{-4}
SchafferN.4 (2D)	3.8183×10^{-5}	3.8183×10^{-5}	3.8183×10^{-5}	3.8183×10^{-5}
Schwefel2.26 (3D)	3.82×10^{-5}	3.82×10^{-5}	1.7766×10^{1}	4.1430×10^{-2}
Rastrigin (6D)	0.00	0.00	1.9906×10^{-1}	4.6399×10^{-2}
Rastrigin (30D)	0.00	0.00	1.1779×10^{2}	1.4880×10^{1}
Rastrigin (50D)	0.00	0.00	3.3858×10^{2}	6.5766×10^{1}

The convergence curve of the test function is shown in Figure 6.



(**a**) Testing 1D basis functions

(**b**) Testing of 2D reference functions

(**c**) Testing of 3D reference functions

(**d**) Testing of 6D reference functions

(**e**) Testing of 30D reference functions

(**f**) Testing of 50D reference functions

Figure 6. Convergence curve of the test function.

As can be seen from the figure, on the Schwefel2.26 and SchafferN.4 datasets, DOA, IDOA and PSO all performed well, achieving accuracy in the 10^{-5} range, whilst GA performed slightly less well but still achieved accuracy in the 10^{-4} range; On the Schwefel2.26 dataset, the accuracy of DOA and IDOA is higher than that of PSO and GA; they successfully avoid local optima and demonstrate excellent global search capabilities. On the Rastrigin dataset, both DOA and IDOA can achieve the theoretical optimal value of 0, whereas the accuracy of PSO and GA declines sharply as the dimension increases, indicating that DOA-type algorithms possess a distinct advantage in high-dimensional complex problems. IDOA outperforms PSO and GA in terms of convergence accuracy, convergence speed and robustness, and performs particularly well in high-dimensional multi-peaked optimization problems, offering promising prospects for practical application.

4. Simulation Validation and Results Analysis

4.1. Simulation Environment

Fitness functions in real-world engineering problems are typically complex and subject to numerous constraints; the chaotic initialization and dynamic perturbation methods employed by IDOA play a significant role. The experimental environment was set up in MATLAB R2025a, running on a 64-bit Windows 10 operating system with an Intel Core i7-8700 CPU @ 3.20 GHz. The main experimental parameter settings are shown in Table 3. To ensure the fairness of the comparative experiments, IDOA, DOA, ACO, and GA were run independently an equal number of times using the same random seed.

Table 3. Parameter Settings.

Parameters	Value
Number of obstacles	8–10
Obstacle diameters	Ranging from 16 m to 30 m
Population size N	50
Maximum number of iterations T	250
Number of control points n_c	10
Path length weighting w_1	0.3
Safety weighting w_2	0.4
Smoothing weight w_3	0.3
safety margin h_{safe}	10 m
Minimum distance between control points $d_{\min}$	5 m
Maximum detour ratio $R_{\max}$	2.0
Initial intensity of the primordial chaos α_0	0.05
Chaos attenuation factor λ	0.98
Starting point	(10, 10, 10)
Finish line	(90, 90, 30)

4.2. Discussion

All comparative experiments were conducted under the identical 3D environment, parameter settings, and hardware conditions to ensure a fair comparison. Each algorithm was run 20 independent times with the same random seed to eliminate random interference, and the optimal results are reported in Table 4.

Table 4. Comparison of DOA and IDOA path planning performance.

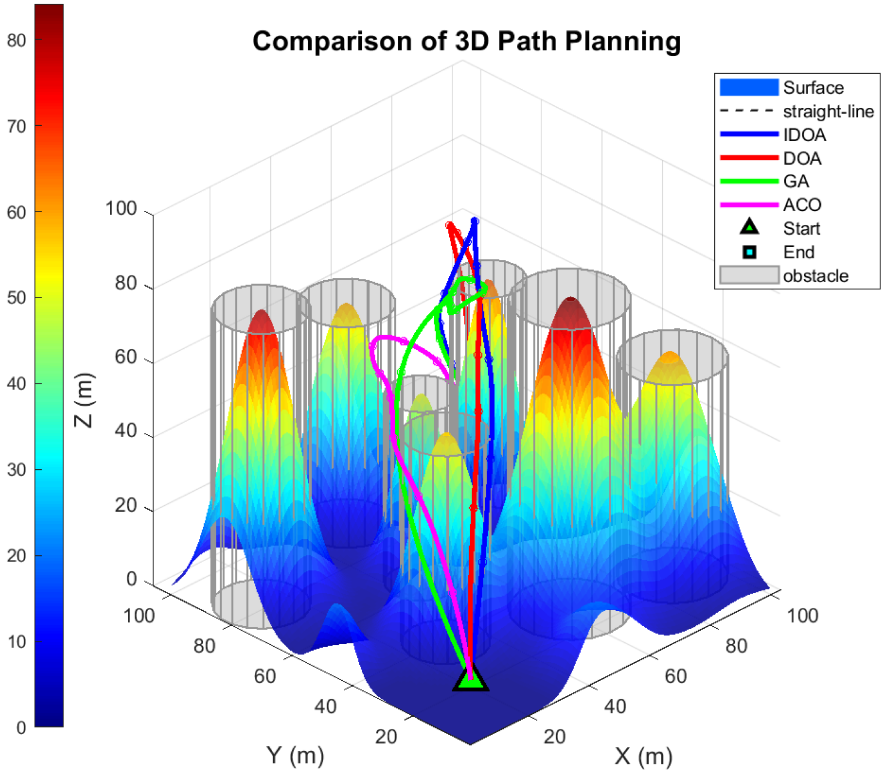
Algorithm	Optimal Fitness	Path Length (m)	Mean Curvature	Runtime (s)	Per-Iteration Runtime (s)	The Cost of Security
DOA	66.528	156.9	0.2723	201.05	0.80	60.4
IDOA	58.319	140.3	0.1490	268.08	1.07	61.1

Table 4. *Cont.*

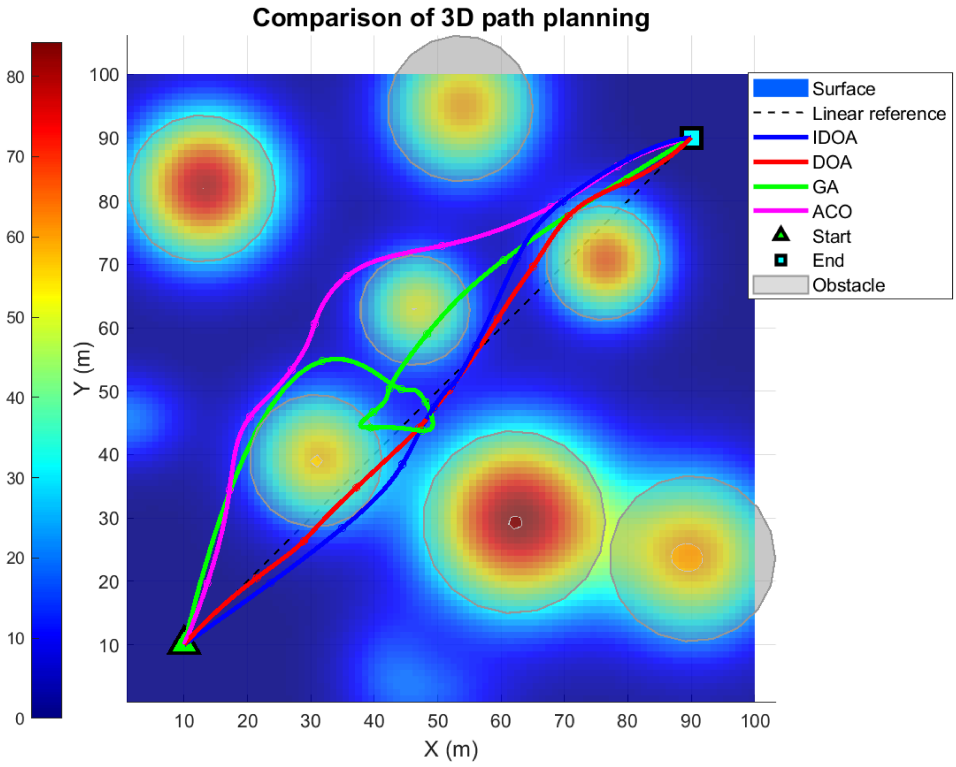
Algorithm	Optimal Fitness	Path Length (m)	Mean Curvature	Runtime (s)	Per-Iteration Runtime (s)	The Cost of Security
ACO	82.646	212.8	0.3042	222.26	0.89	63.5
GA	94.4805	219.5	0.3218	254.77	1.02	64.2

As shown in Table 4, the proposed IDOA exhibits superior optimization performance across all metrics. The optimal fitness value of IDOA is 58.319, representing a 12.34% reduction compared to the baseline DOA (66.528), and significantly outperforming both ACO (82.646) and GA (94.4805). In terms of path length, IDOA yields a trajectory of 140.3 m, which is 10.58% shorter than DOA's 156.9 m, and substantially shorter than those obtained with ACO (212.8 m) and GA (219.5 m). The IDOA-generated path is notably closer to the straight-line distance between the start and end points, confirming its enhanced optimization efficiency. Furthermore, the mean curvature of the IDOA path is reduced to 0.1490, marking a 45.28% improvement over DOA (0.2723) and substantially lower than ACO (0.3042) and GA (0.3218). This indicates that the IDOA trajectory features gentler turns and smaller curvature variations, resulting in a smoother flight path that better meets the kinematic constraints of UAVs. Regarding computational cost, the runtime of IDOA (268.08 s) is slightly longer than that of DOA (201.05 s) and ACO (222.26 s), but remains competitive with GA (254.77 s). The per-iteration overhead across all algorithms is low and comparable, with IDOA's per-iteration runtime only slightly higher than the other methods. This marginal increase is attributable to its chaotic initialization, dynamic perturbation, and adaptive stage division mechanisms, which, while adding a small computational cost per iteration, result in significant improvements in path quality. It should be noted that the proposed IDOA method is designed for offline UAV path pre-planning, where this runtime trade-off is acceptable. In terms of safety, IDOA maintains a safety cost of 61.1, which is marginally higher than DOA (60.4) but significantly lower than ACO (63.5) and GA (64.2). This minor increase reflects a deliberate balance: the algorithm approaches obstacles more closely to generate shorter and smoother paths while still satisfying all safety constraints. This trade-off is acceptable for practical UAV applications. The 3D trajectories obtained by DOA, IDOA, ACO, and GA are visualized in Figure 7.

As can be seen from the figure, the paths generated by IDOA are relatively smooth in both the horizontal and vertical directions, conforming to the UAV's dynamic constraints; all detours are made for the purpose of obstacle avoidance, maximizing path efficiency whilst meeting safety and smoothness requirements; In contrast, the path generated by DOA exhibits local fluctuations, with multiple significant changes in altitude in the vertical direction, and includes unnecessary excessive climbs; the green curve generated by GA contains obvious redundant detours and deviates significantly from the straight reference direction, indicating poor overall directional consistency, which suggests that the genetic algorithm has low search efficiency in complex multi-constraint problems. Comparisons of path curvature, path length and altitude profiles are shown in Figure 8.

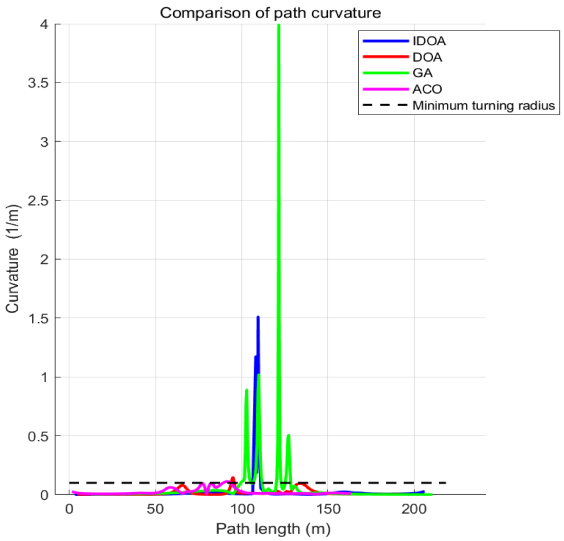


(**a**) side view

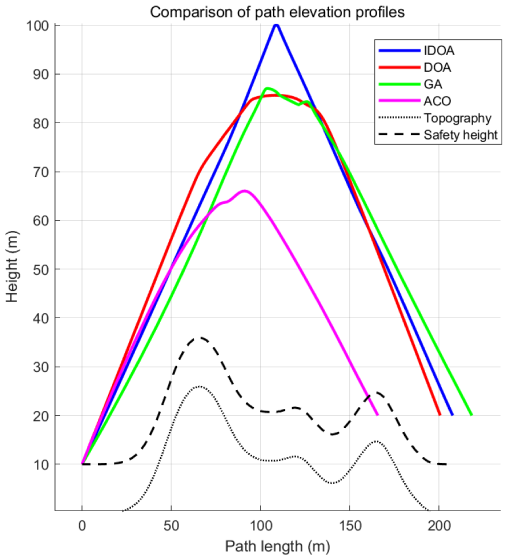


(**b**) top view

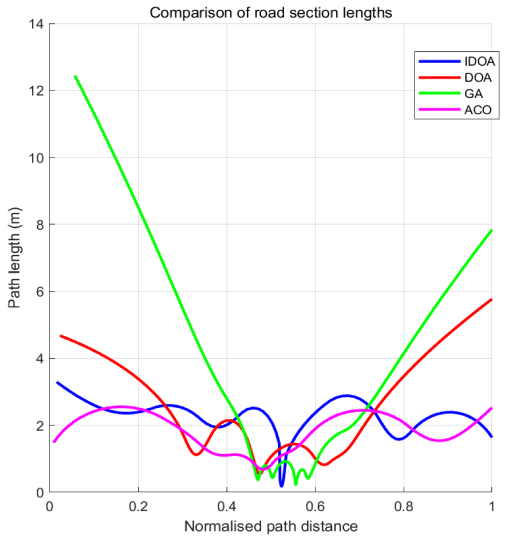
Figure 7. 3D flight path diagram.



(**a**) Comparison of path curvatures



(**b**) Elevation comparison diagram



(**c**) Path Length Distribution Chart

Figure 8. Comparison of the three indicators.

Figure 8 shows that the spikes in curvature occur when the path makes sharp turns to avoid obstacles in complex terrain. The elevation spikes correspond to steep climbs/descents over high peaks. These are normal and necessary adjustments for safe navigation. The average curvature of the IDOA is lower than that of the DOA, with less fluctuation; the proportion of curvature exceeding the limit has decreased from 44.9% to 36.7%, indicating a higher degree of compliance with motion constraints. Figure 7b demonstrates that the elevation changes along the IDOA path are more gradual than those of the DOA, avoiding steep ascents and descents and achieving better conformity with the terrain. The above results fully demonstrate that IDOA outperforms DOA in terms of path length, smoothness and control point distribution, with a significant improvement in path quality. The convergence curve for the trajectory planning is shown in Figure 9:

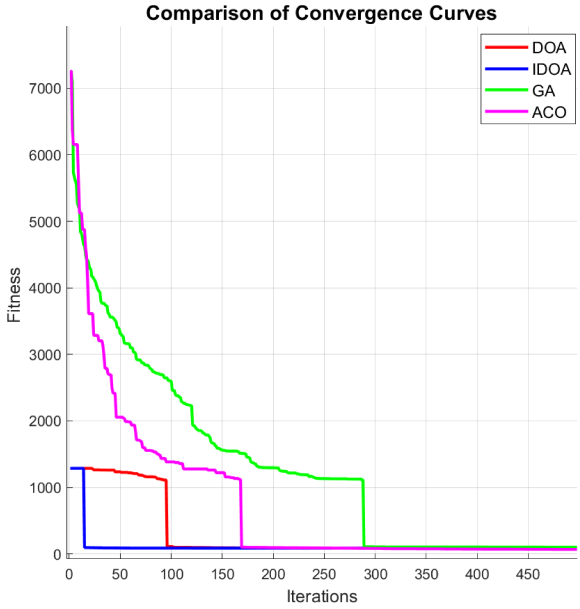


Figure 9. Convergence cure.

As can be seen from the figure, the IDOA demonstrates superior overall convergence performance compared to DOA and GA; its convergence rate is significantly faster than that of DOA, whilst its convergence stability and optimal fitness values are superior to those of GA. In the early stages of iteration, the fitness values of the GA exhibit a sharp downward trend, but the decline is characterized by significant fluctuations and rapidly stabilizes; The fitness values of the IDOA decline steadily and rapidly, levelling off after approximately 60 iterations, with the final fitness value being considerably lower than that of GA; the convergence process of DOA is the smoothest, with the slowest rate of decline in fitness values, and its optimal fitness value is higher than that of IDOA. The per-iteration overhead is comparable across algorithms. The moderate increase in IDOA's total runtime supports offline pre-planning and is justified by improved path quality.

5. Conclusions

This paper proposes an Improved Dhole Optimization Algorithm (IDOA) for 3D path planning of quadrotor UAVs in complex urban and hilly environments. By integrating logistic chaotic mapping, dynamic chaotic perturbation, and an adaptive stage-division strategy, IDOA addresses the limitations of the original DOA, such as premature convergence and unstable path quality.

Experimental results demonstrate that IDOA outperforms DOA, ACO, and GA in fitness value, path length, smoothness, and convergence performance. The generated paths

are shorter, smoother, and safer, fully satisfying UAV kinematic and obstacle avoidance constraints in complex 3D terrain.

Limitations of this work include the focus on static obstacles and offline planning scenarios. Future work will focus on validating IDOA on real UAV platforms, extending it to dynamic obstacle environments, and further optimizing computational efficiency for real-time onboard implementation.

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

Funding: This study has 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 and No. JCZX2024ZZ25). This work has also been supported by Sichuan Flight Engineering Technology Research Center Project (Grant No. GY2024-30D). This work has also been supported by Student Innovation and Entrepreneurship Training Program (Grant No. 202510624005). This work has also been supported by Sichuan Provincial Engineering Research Center of Domestic Civil Aircraft Flight and Operation Support (Grant No. MJCYZY202503). This work has also been supported by the Fundamental Research Funds for the Central Universities (Grant No. 26CAFUC01004). This work has also been supported by the Fundamental Research Funds for the Central Universities (Grant No. 26CAFUC03008).

Data Availability Statement: The original contributions presented in the manuscript are included in the article; any further inquiries can be directed to the corresponding author.

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

References

1. Tan, J.; Ma, X.; Li, X. Research on UAV 3D flight track planning and dynamic obstacle avoidance algorithm. *Chin. J. Sci. Instrum.* **2019**, *40*, 224–233.
2. Zhao, J.; Zhang, Y.; Ning, L.; Xiao, X.; Bai, C.; Zhang, J.; Yang, M. Adaptive Path Planning of UAV Based on A* Algorithm and Artificial Potential Field Method. *Drones* **2026**, *10*, 93. [CrossRef]
3. Orozco-Rosas, U.; Montiel, O.; Sepúlveda, R. Mobile robot path planning using membrane evolutionary artificial potential field. *Appl. Soft Comput. J.* **2019**, *77*, 236–251. [CrossRef]
4. Yin, G.; Wu, B.; Bao, X.; Guo, X.; Xie, Q. Path Planning for Slab Manipulating Robotic Arms Based on Improved RRT Combined with Minimum Snap. *J. Inst. Eng. Ser. C* **2026**, *107*, 795–807. [CrossRef]
5. Xu, F.; Liu, X.; Zhao, X.; Gao, D. Three-Dimensional Path Planning for UAV Swarms Based on an Improved Particle Swarm Optimization Algorithm. In Proceedings of the 2025 5th International Conference on Big Data, Artificial Intelligence and Risk Management, Dongguan, China, 12–14 December 2025; pp. 197–203. [CrossRef]
6. Goldberg, D.E. *Genetic Algorithms in Search, Optimization, and Machine Learning*; Addison-Wesley: Boston, MA, USA, 1989.
7. Dorigo, M.; Maniezzo, V.; Colorni, A. Ant system: Optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man Cybern.* **1996**, *26*, 29–41. [CrossRef] [PubMed]
8. Maleki, A.; Roayaei, M.; Mirjalili, S. Enhancing leadership-based metaheuristics using reinforcement learning: A case study in grey wolf optimizer. *Knowl.-Based Syst.* **2025**, *330*, 114471. [CrossRef]
9. Wang, X.; Zhang, Q.; Jiang, S. Dynamic UAV Path Planning Based on an Improved Whale Optimization Algorithm. *Comput. Appl.* **2025**, *45*, 928–936.
10. Huang, Y.; Li, H.; Dai, Y.; Lu, G.; Duan, M. A 3D Path Planning Algorithm for UAVs Based on an Improved Artificial Potential Field and Bidirectional RRT*. *Drones* **2024**, *8*, 760. [CrossRef]
11. Mu, L.; Liu, W.; Wang, H.; Zhang, Y. Research of UAV 3D path planning based on improved Dwarf mongoose algorithm with multiple strategies. *Sci. Rep.* **2025**, *15*, 26979. [CrossRef] [PubMed]
12. Feng, W.-Q.; Yang, Y.; Yang, L.-F.; Fu, Y.-J.; Xu, K.-J. Unmanned Aerial Vehicle Logistics Distribution Path Planning Based on Improved Grey Wolf Optimization Algorithm. *Symmetry* **2025**, *17*, 2178. [CrossRef]
13. Li, Z.; Zong, X.; Hao, J.; Kochan, O. Multi-UAV 3D path planning based on improved sparrow search algorithm. In Proceedings of the Doors-2025: 5th Edge Computing Workshop, Zhytomyr, Ukraine, 4 April 2025.

14. Deng, J.; Zhang, H.; Zhang, Y.; Zhong, G.; Hua, M. UAV Trajectory optimization method based on improved grey wolf optimizer with hybrid strategy. *J. King Saud Univ. Comput. Inf. Sci.* **2026**, *38*, 69. [CrossRef]
15. Mohammed, O.B.; Aghdasi, S.H.; Salehpour, P. Dhole optimization algorithm: A new metaheuristic algorithm for solving optimization problems. *Clust. Comput.* **2025**, *28*, 430. [CrossRef]
16. Liu, H.; Wang, J. A graded penalty strategy for constrained evolutionary optimization. *Appl. Soft Comput.* **2019**, *82*, 105583. [CrossRef]
17. Liu, Q.; Wang, H. UAV 3D Path Planning Based on Improved Grey Wolf Optimization Algorithm. *Front. Comput. Intell. Syst.* **2023**, *3*, 113–116. [CrossRef]
18. Wang, Y.; Li, Z. Three-Dimensional UAV Path Planning Using Improved PSO-B-Spline Algorithm with Dynamic Constraint Handling. *Drones* **2023**, *7*, 421. [CrossRef]
19. Liu, S.; Zhang, Y. Smooth 3D UAV Path Planning Based on Improved PSO and B-Spline Curve. *Aerospace* **2022**, *9*, 268. [CrossRef]
20. Wu, Q. Clearing-up Doubts about the Derived-Vector Formula of B-Spline Curve by de Boor Algorithm. *J. Xinjiang Norm. Univ. (Nat. Sci. Ed.)* **2000**, *4*, 7–15.
21. Ma, X.; Li, H. A reproducible whale algorithm based on the Tent chaotic map. *Comput. Simul.* **2022**, *39*, 363–368.
22. Luo, C.-Y. Robot Trajectory Planning Using the Whale Algorithm Optimised via the Logistic Chaotic Map. *Mech. Manag. Dev.* **2025**, *40*, 175–176+179.
23. Li, X.; Yu, S. Hardware implementation of chaotic encryption on ARM embedded platform. *Comput. Electr. Eng.* **2017**, *62*, 288–301. [CrossRef]
24. Karagiorgos, N.F.; Stavrinides, S.G.; Benito, C.D.; Nikolaidis, S.; Picos, R. Unconventional Security for IoT: Hardware and Software Implementation of a Digital Chaotic Encrypted Communication Scheme. *IEEE Internet Things J.* **2024**, *11*, 19914–19925. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.