symmetry

MDPI

Article

Unmanned Aerial Vehicle Path Planning Method Based on Improved Dung Beetle Optimization Algorithm

Fengjun Lv [1,*], Yongbo Jian [2], Kai Yuan [3] and Yubin Lu [2]

[1] Zhejiang Institute of Communications, Hangzhou 311112, China
[2] School of Modern Post, Nanjing University of Posts and Telecommunications, Nanjing 210023, China; 1223097101@njupt.edu.cn (Y.L.)
[3] Anhui Yugu Express Intelligent Technology Co., Ltd., Wuhu 241300, China; 15083971961@163.com
* Correspondence: lvfengjun@zjvtit.edu.cn

Abstract: To address the problem of UAV path planning in complex mountainous terrains, this paper comprehensively considers constraints such as natural mountain and obstacle collision threats, the shortest path, and flight altitude. We propose a more practical UAV path planning model that better reflects the actual UAV path planning situation in complex mountainous areas. In order to solve this model, this paper improves the traditional dung beetle optimization (DBO) algorithm and proposes an improved dung beetle optimization (IDBO) algorithm. The IDBO algorithm optimizes the population initialization method based on the concept of symmetry, ensuring that the population is more evenly distributed within the solution space. Additionally, the algorithm introduces a sine–cosine function-based movement strategy, inspired by the symmetry principle, to enhance the search efficiency of individual population members. Furthermore, a population evolution strategy is incorporated to prevent the algorithm from getting stuck in local optima. To demonstrate the algorithm's performance, tests were conducted using 23 commonly used benchmark functions provided by the CEC 2005 competition and six commonly used engineering problem models provided by the CEC 2020 competition. The results indicate that IDBO significantly outperforms DBO in terms of convergence performance, effectively solving various engineering optimization problems. Finally, experimental tests under three different threat scenarios show that the proposed IDBO algorithm has scientific validity when applied to UAV path planning. This solution method effectively reduces UAV flight energy consumption costs and obstacle collision threats while improving the efficiency and accuracy of UAV path planning.

Keywords: UAV; flight path planning; metaheuristic algorithm; dung beetle optimization algorithm

check for updates

Academic Editors: Hsien-Chung Wu and Zhengqiu Zhang

Received: 24 December 2024
Revised: 22 February 2025
Accepted: 25 February 2025
Published: 28 February 2025

Citation: Lv, F.; Jian, Y.; Yuan, K.; Lu, Y. Unmanned Aerial Vehicle Path Planning Method Based on Improved Dung Beetle Optimization Algorithm. *Symmetry* **2025**, *17*, 367. https://doi.org/10.3390/sym17030367

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With the rapid development of information technology, reconnaissance and search technologies, represented by drones, have been widely applied in both military and civilian scenarios [1]. Among these, path planning is a critical aspect of drone applications, aiming to determine the optimal path from a starting point to a target point while meeting various constraints such as obstacle avoidance, energy efficiency, and time limits. Effective path planning not only enhances mission efficiency but also significantly reduces operational costs and risks for drones. In reconnaissance and search missions, drones often need to inspect targets over long distances while navigating environments with potential threats. Additionally, drones are constrained by their mobility, endurance, and payload

capacity, which pose significant challenges to collaborative path planning [2]. Traditional path planning methods often fail to meet the high real-time demands of dynamic and unpredictable environments, resulting in slow responses, poor adaptability, and low computational efficiency. To address these challenges, this study investigates an improved dung beetle optimization (DBO) algorithm for drone path planning, which holds significant engineering value.

To address the problem of drone path planning, scholars worldwide have proposed various solutions. Regarding model construction, Zhang et al. [3] established a drone path planning model by integrating optimization objectives such as energy consumption, trajectory cost, obstacle avoidance, smoothing, and stability, transforming the path planning problem into a multi-constraint objective function optimization task. Liu et al. [4] proposed a path planning model tailored to the limitations of agricultural drones operating in complex hilly terrain, considering topographic features and agricultural scheduling requirements. Bai et al. [5] developed a multi-drone collaborative path planning model based on multi-objective optimization, focusing on trajectory distance, time, threats, and coordination costs. Wang et al. [6] introduced a comprehensive framework incorporating drone capabilities, terrain, complex regions, and dynamic mission requirements, along with a dynamic collaborative path planning algorithm for complete area coverage. Wang et al. [7] formulated a mathematical model integrating multiple constraints such as flight distance, collision threats, and path stability, effectively converting the complex problem of drone formation path planning into an optimization problem. These advancements have enabled subsequent research to construct more practical drone path planning models.

Path planning algorithms can be categorized into two main types: traditional algorithms and intelligent algorithms. Traditional algorithms include the artificial potential field (APF) method [8], Simulated Annealing (SA) [9], A* algorithm [10], and Rapidly-Exploring Random Tree (RRT) [11]. However, these methods are prone to premature convergence, leading to local optima, and exhibit slow convergence speeds in large-scale search spaces. For instance, APF, as a local path planning method, conceptualizes drones as particles moving within an artificial potential field, where attractive and repulsive forces guide the drones' direction [12]. However, APF often falls into local optima and sometimes fails to provide feasible solutions [13]. The A* algorithm discretizes the search space into grids and selects the path covering the fewest grid cells, but its performance heavily depends on the grid segmentation method and incurs excessive computation time in complex spaces [14]. RRT, a sampling-based algorithm, generates redundant nodes during iterations and frequently fails to find optimal paths [15]. These limitations highlight the shortcomings of classical algorithms.

In contrast, intelligent algorithms offer faster convergence, broader applicability, and lower time complexity, such as Ant Colony Optimization (ACO) [16], Particle Swarm Optimization (PSO) [17], and Genetic Algorithm (GA) [18]. Researchers have improved these intelligent optimization algorithms for path planning. For example, Li et al. [19] introduced the Metropolis criterion into ACO node selection to propose an improved MACO algorithm, which includes initial trajectory generation, trajectory correction, and smooth trajectory planning. Zhang et al. [3] addressed the inefficiency and infeasibility of traditional algorithms in complex 3D environments by enhancing Differential Evolution (DE) with mutation crossover strategies, adaptive guidance mechanisms, and elite interference mechanisms, resulting in the multi-strategy improved DE (MSIDE) algorithm. Liu et al. [20] proposed a multi-mechanism enhanced Grey Wolf Optimization (NAS-GWO) algorithm, incorporating evolutionary boundary constraints to improve search accuracy, Gaussian mutation and spiral functions to avoid local optima, and a sigmoid function to balance exploration and exploitation. Hu et al. [21] developed a multi-algorithm hybrid ACO (MA-

HACO) with adaptive foraging strategies, multi-stage stochastic strategies, and aggregation mutation strategies to enhance exploration and diversity. Gu et al. [22] introduced the IRIME algorithm for complex urban environments, integrating frost diffusion mechanisms, high-altitude condensation strategies, and lattice weaving strategies to enhance the RIME algorithm's global exploration and avoid premature convergence. Pang et al. [23] proposed a multi-objective cat swarm optimization with dual-archive mechanisms (MOCSO_TA) to balance convergence and population diversity.

Although these improved algorithms generally exhibit superior convergence performance for low-dimensional unimodal problems, they often fall into local optima when addressing high-dimensional multimodal problems. To overcome these challenges, the main contributions of this study are as follows:

1. An improved population initialization method to comprehensively cover the solution space, enhancing the uniformity of the initial solution distribution.

2. A sinusoidal motion strategy to mitigate local optima in the traditional DBO algorithm, improving global search capability and convergence speed for multi-drone path planning in dynamic environments.

3. Evolutionary strategies to enhance the global search ability of the "rolling dung beetle" and the local exploitation ability of the "small dung beetle", while enabling "thief dung beetles" to evolve positions based on opposition-based learning (OBL), increasing population diversity.

Building on these strategies, an improved dung beetle optimization (IDBO) algorithm is proposed. Validation is performed using 23 benchmark functions from the CEC 2005 competition and six engineering models from the CEC 2020 competition. Results demonstrate that IDBO outperforms DBO in convergence performance and provides better solutions to various problems. Experiments under three threat scenarios further confirm the scientific validity of IDBO for drone path planning, effectively reducing flight costs and threats while improving planning efficiency and accuracy.

Compared with other related papers in the same field [24], this paper has the following distinctive features:

1. Algorithm Improvement: This paper improves the dung beetle optimization (DBO) algorithm using a uniform initialization strategy, a cosine function-based movement strategy, and a population evolution strategy. The strategies are more focused on updating the population positions, and the population evolution strategy is relatively novel.

2. Comprehensive Testing: The paper uses 23 commonly used benchmark functions from the 2005 Congress on Evolutionary Computation (CEC2005) and 6 commonly used engineering problem models from the 2020 Congress on Evolutionary Computation (CEC2020) for testing. This broader testing range is explained in detail in the paper, with the specifics of the test functions provided. Additionally, this paper compares 10 different metaheuristic algorithms, making the testing results more persuasive.

3. Performance Comparison: The improved algorithm proposed in this paper outperforms most other enhanced DBO algorithms. The experimental results show that the proposed algorithm performs better than others on 22 out of the 23 benchmark functions in CEC2005, and it converges the fastest on the engineering problem models in CEC2020.

4. Diverse Testing Scenarios: The paper tests the algorithm in different terrains and environments with varying obstacle densities and compares different cost–weight scenarios, providing a more comprehensive evaluation.

5. Results Visualization: In addition to presenting the total cost and computation time, the paper also provides values for the individual cost components and heading deviation. Furthermore, the optimization paths of each algorithm are visualized in both

3D and overhead views for each testing scenario, allowing readers to better compare the strengths and weaknesses of the algorithms.

This paper is organized as follows: Section 2 introduces the cost components and constraints of the drone path planning model. Section 3 details the traditional DBO algorithm and the proposed IDBO improvement strategies. Section 4 evaluates IDBO's performance by comparing it with nine algorithms on 23 benchmark functions and six engineering models. Section 5 presents simulation experiments using different algorithms for drone path planning. Section 6 concludes the study.

2. Model Construction

2.1. Drone Flight Terrain Environment Modeling

To enhance path search efficiency in UAV 3D path planning, real three-dimensional spatial terrain information must be utilized. This paper simultaneously considers the original terrain and obstacle regions to establish the UAV flight environment. The baseline terrain model is defined as follows:

$$z_1(x,y) = \sin(y+a) + b\sin x + c\cos(d\sqrt{x^2+y^2}) + e\cos y + f\sin(g\sqrt{x^2+y^2}) \quad (1)$$

where x and y represent the horizontal and vertical coordinates projected onto the horizontal plane, respectively. The constants a, b, c, d, e, f and g are control factors that define surface characteristics used to describe terrain undulations. The function $z_1(x,y)$ simulates the baseline terrain features of a digital elevation model.

In complex terrain environments, UAVs must avoid natural mountains. To define mountain peaks, we introduce a composite function:

$$z_2(x,y) = \sum_{i=1}^{n} h_i \times \exp\left[-\left(\frac{x-x_i}{x_{si}}\right)^2 - \left(\frac{y-y_i}{y_{si}}\right)^2\right] \quad (2)$$

where (x_i, y_i) denotes the center coordinates of the i-th mountain; n is the total number of mountain peaks in the environment; h_i is the height parameter controlling the peak; and x_{si} and y_{si} represent the decay rates of peak i along the x and y axes, respectively. The function $z_2(x,y)$ models the hilly and mountainous terrain data of the digital elevation map.

By combining $z_1(x,y)$ and $z_2(x,y)$, the UAV flight environment can be generated. Figure 1 illustrates the three-dimensional terrain map established using the above formulas.

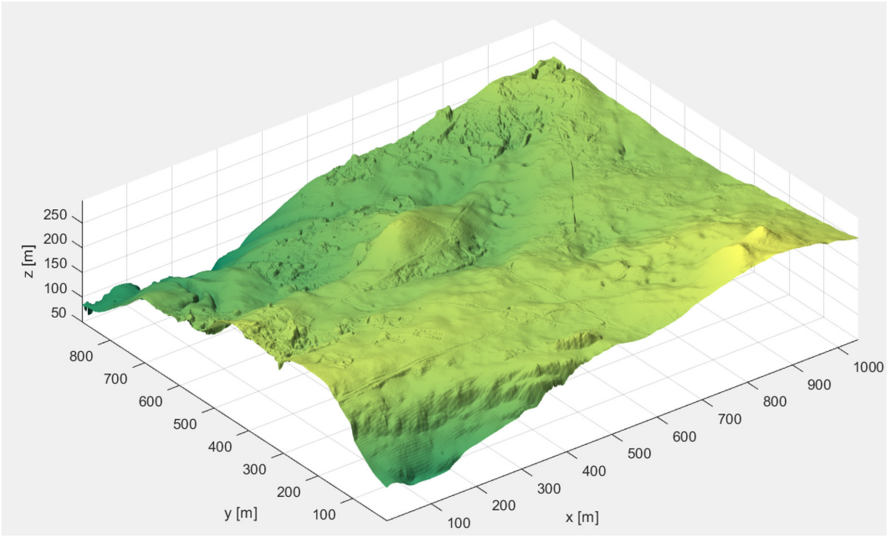

Figure 1. Three-dimensional terrain map.

2.2. Drone Path Planning Objectives

2.2.1. Shortest Path Objective

Taking into account that a shorter flight distance results in lower costs, we set one of the objective functions as the flight distance. This is represented by the sum of Euclidean distances between intermediate path nodes during the flight process. Let $P_i = (x_i, y_i, z_i)$ represent a waypoint on the flight path, and the entire trajectory $X_i = \{P_0, P_1, P_2, \ldots, P_n, P_{n+1}\}$ can be represented as a three-dimensional array consisting of n waypoints, where P_0 is the starting point of the UAV, and P_{n+1} is the target point. Thus, the objective of minimizing the flight distance can be represented as follows:

$$C_1 = \sum_{i=0}^{n} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2} \tag{3}$$

2.2.2. Altitude Variation Objective

During the flight process, frequent changes in altitude can result in unnecessary energy consumption for the UAV. The objective is to minimize altitude changes, represented as follows:

$$C_2 = \sum_{i=0}^{n-1} o_1 \Delta z_{(i-1,i)} \tag{4}$$

where o_1 is the altitude change penalty coefficient, and $\Delta z_{(i-1,i)}$ represents the altitude change between the i-th and $(i-1)$-th path points.

2.2.3. Obstacle Threat Objective

The threat area of obstacles is represented in the form of cylinders, where C_k denotes the center coordinates of cylinder k, R_k represents the radius of the obstacle, and D is the outer collision threat area of the obstacle. The projection of the obstacle threat area on the two-dimensional plane is shown in Figure 2.

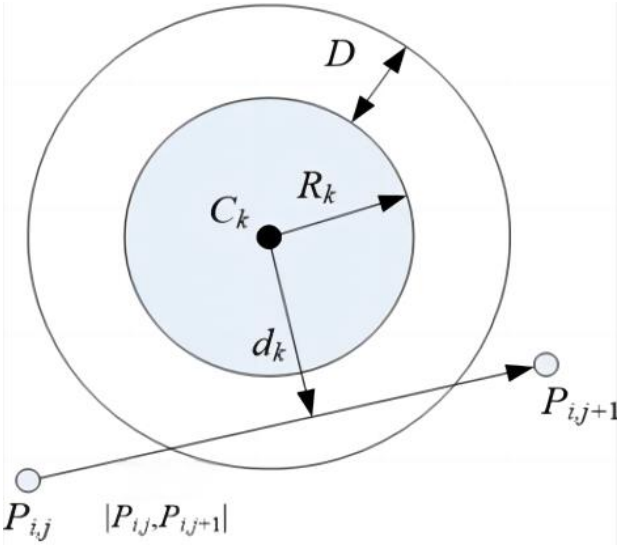

Figure 2. Planar projection of obstacle threat area.

It is observed that the threat cost to the UAV is inversely proportional to the Euclidean distance $|P_i, P_{i+1}|$ between consecutive waypoints P_i and P_{i+1} to the center coordinates d_k

of cylinder k. Let φ_c be the obstacle threat penalty factor and K be the set of all obstacle threat areas. Then, the threat objective function for the trajectory can be defined as follows:

$$C_3 = \sum_{i=0}^{n} \sum_{k=0}^{K} T_k(|P_i, P_{i+1}|) \tag{5}$$

$$T_K(|P_i, P_{i+1}|) = \begin{cases} 0, d_k > D + R_k \\ \varphi_c((D + R_k) - d_k), R_k < d_k < D + R_k \\ \infty, d_k < R_k \end{cases} \tag{6}$$

In summary, the model's objective function is:

$$\text{minC} = \text{C1} + \text{C2} + \text{C3} \tag{7}$$

2.3. Model Constraints

2.3.1. Maximum Range Constraint Indicator

Considering the limitations of the UAV's battery and mission duration, exceeding the maximum range for the planned trajectory will result in mission failure. Therefore, it is crucial to include the maximum range constraint as one of the evaluation metrics. Let the length of a segment i be l_i, the average speed on this segment be $\overline{v}_i$, the time required for this segment be t_i, the UAV's maximum range be $l_{\max}$, and the time limit for the mission be T. The maximum range constraint can be expressed as:

$$l_i = \overline{v} \times t_i (0 < t_i \leq T) \tag{8}$$

$$\sum_{i=1}^{n} l_i \leq l_{\max} (0 < i \leq n) \tag{9}$$

2.3.2. Minimum Step Length Constraint

Due to the impact of UAV attitude adjustment delays, to ensure delivery safety, the straight flight step length f_i reserved for UAV maneuvers in both the horizontal and vertical directions needs to be greater than the minimum step length $f_{\min}$. The formula is:

$$f_i \geq f_{\min} \tag{10}$$

2.3.3. Maximum Yaw Angle Constraint

The maximum yaw angle represents the UAV's maneuvering limit on the horizontal plane. To meet maneuverability constraints, the UAV must perform maneuvers within a certain yaw angle. Assuming the coordinates of two consecutive waypoints are (x_i, y_i, z_i) and $(x_{i+1}, y_{i+1}, z_{i+1})$, and $z_i = z_{i+1}$, the yaw angle α can be calculated as follows:

$$\alpha = \arccos\{\frac{(x_i - x_{i-1})(x_{i+1} - x_i) + (y_i - y_{i-1})(y_{i+1} - y_i)}{\sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \times \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}}\} \tag{11}$$

If $\alpha_{\max}$ is the maximum yaw angle, then the yaw angle α must satisfy:

$$0 \leq \alpha \leq \alpha_{\max} \tag{12}$$

3. Improved Dung Beetle Optimization Algorithm and Algorithm Testing

3.1. Principle of the Dung Beetle Optimization Algorithm

Jianka Xue and Bo Shen [25] proposed a population-based optimization algorithm called the dung beetle optimization (DBO) algorithm in 2022, inspired by the rolling, danc-

ing, foraging, stealing, and breeding behaviors of dung beetles. This algorithm effectively balances global exploration and local exploitation, demonstrating fast convergence and high accuracy. It is suitable for addressing the challenges of complex dynamic environments in multi-UAV path planning and task allocation.

The DBO algorithm first requires population initialization and fitness calculation, which are similar to general algorithms. Then, the dung beetle population is divided into four subpopulations representing rollers, breeders, minors, and thieves, with the proportions set by the authors as 6:6:7:11. After partitioning the subpopulations, individuals update their positions according to different strategies, as follows:

(1) Rollers

Rollers use sunlight to guide them to roll along a straight line, and natural factors such as sunlight intensity and wind can affect their movement trajectory. When there are no obstacles, the position update of rollers is as follows:

$$\Delta x = \left| x_i(t) - X^W \right| \tag{13}$$

$$x_i(t+1) = x_i(t) + \alpha \times k * x_i(t-1) + b * \Delta x \tag{14}$$

In the equation, t represents the iteration number; $x_i(t)$ represents the position information of the i-th dung beetle at the t-th iteration; and k is the perturbation coefficient, where $k \in [0, 2]$. b is a constant value in the range [0, 1], and α represents the deviation in the position of the dung beetle caused by natural factors. It takes a constant value of -1 or 1. When it is 1, there is no deviation, and when it is -1, there is deviation in the direction. X^W is the worst position in the global, and Δx represents the change in light intensity. A larger value indicates weaker light.

When a dung beetle encounters an obstacle and cannot move forward, it changes its direction by dancing, as follows:

$$x_i(t+1) = x_i(t) + \tan\theta |x_i(t) - x_i(t-1)| \tag{15}$$

where θ represents the perturbation angle, with a range of $[0, \pi]$. If its value is 0, $\pi/2$, or π, the position of the dung beetle is not updated.

(2) Breeder Beetles

Breeder beetles use a boundary selection strategy to simulate the egg-laying area of female dung beetles. The egg-laying area is defined as follows:

$$L^* = \max(X^*(1-R), L) \tag{16}$$

$$U^* = \min(X^*(1+R), U) \tag{17}$$

$$R = (1-t)/T \tag{18}$$

where L^* is the upper bound of the egg-laying area, U^* is the lower bound of the egg-laying area, X^* represents the best local position, L is the upper bound of the optimization problem, U is the lower bound of the optimization problem, R is a parameter that varies with the number of iterations, t represents the current iteration number, and T is the maximum number of iterations.

The position of the breeder beetle dynamically changes during the iteration process and can be expressed as:

$$B_i(t+1) = X^* + b_1 \times (B_i(t) - Lb^*) + b_2 \times (B_i(t) - Ub^*) \tag{19}$$

In the equation, $B_i(t)$ represents the position of the i-th breeder ball at the t-th iteration. b_1 and b_2 are two independent vectors of size $1 \times D$, where D represents the dimensionality.

(3) Minor Beetles

An optimal foraging area needs to be established to guide the juvenile dung beetles in their search for food and to simulate their foraging behavior. The boundaries of the optimal foraging area are defined as follows:

$$L^b = \max(X^b(1-R), L) \tag{20}$$

$$U^b = \min(X^b(1+R), U) \tag{21}$$

In the equation, L^b represents the lower boundary of the optimal foraging area, X^b represents the global best foraging position, and U^b represents the upper boundary of the optimal foraging area.

Thus, the updated position for the minor beetles is:

$$x_i(t+1) = x_i(t) + C_1 * (x_i(t) - Lb^b) + C_2 * (x_i(t) - Ub^b) \tag{22}$$

In the equation, $x_i(t)$ represents the position of the i-th minor beetle at the t-th iteration; C_1 is a random number following a normal distribution; and C_2 is a random vector in the range (0, 1).

(4) Thief Beetles

X^b is the optimal position for food competition, so the position update for thief beetles is as follows:

$$x_i(t+1) = X^b + S \times g \times (\left|x_i(t) - X^*\right| + \left|x_i(t) - X^b\right|) \tag{23}$$

In the equation, $x_i(t)$ represents the position of the i-th thief beetle at the t-th iteration; g is a random vector of size $1 \times D$ following a normal distribution; and S is a constant value.

In summary, the DBO algorithm consists of six main steps:

a. Initialize the population of dung beetles and the parameters of the DBO algorithm.

b. Calculate the fitness values of all individuals based on the objective function.

c. Update the positions of all dung beetles.

d. Perform boundary checks.

e. Update the current best solution and its fitness value.

f. Repeat the above steps until the maximum number of iterations is reached, then output the global optimal value and its solution.

In practical applications, to better test the effectiveness of the algorithm, it is necessary to demonstrate the algorithm and combine it with numerical examples for detailed solutions to test its performance, validating its effectiveness in solving real-world problems. Therefore, testing the algorithm's performance through a detailed solution of a numerical example, based on the set initial parameters and the expression of the test function, verifies the applicability of the algorithm to real-world problems. This helps assess the practical value of the algorithm in solving real-world problems such as multi-drone trajectory planning and task allocation in dynamic environments.

3.2. Improvement Strategies for the Dung Beetle Optimization Algorithm

3.2.1. Improvement of Initial Solution Generation Method

Through analysis and research on metaheuristic algorithms, it is understood that the generation of initial solutions for the population can impact the accuracy and speed of the algorithm's convergence. When solving optimization problems, the initial solutions

should be distributed as uniformly as possible within the solution space. The traditional initialization method for the dung beetle optimization algorithm is as follows:

$$x_{ij} = lb + \alpha \cdot (ub - lb) \tag{24}$$

In the equation, x_{ij} represents the value of the j-th dimension of the i-th dung beetle, ub represents the upper bound, lb represents the lower bound, and α is a random number between 0 and 1.

Due to excessive randomness, the initialized population inevitably exhibits local clustering. To avoid this situation, many scholars choose to generate initial solutions through chaos and reverse learning. Initial solutions generated by chaos exhibit some improvement compared to randomly generated initial solutions, but chaos algorithms tend to introduce strong randomness, which may lead to unstable points. On the other hand, reverse learning, lacking randomness, optimizes initial solutions to some extent but does not significantly enhance solution diversity. Therefore, based on the concept of symmetry, this paper proposes an initialization method based on Formula (25) to comprehensively cover the solution space as much as possible.

$$x_{ij} = lb + (i - 1 + \alpha) \cdot \frac{ub - lb}{i} \tag{25}$$

Figure 3 compares the initial solutions generated by the traditional method with those generated based on Equation (25). The left panel illustrates the population initialized by the traditional method, while the right panel illustrates the population initialized by the proposed method in this paper. The improved population is uniformly distributed within the solution space according to central symmetry, making it easier for the algorithm to locate regions near the optimal value during the early stages of iteration. It can be observed from the comparison that the proposed method effectively optimizes the uniformity of the initial solution distribution.

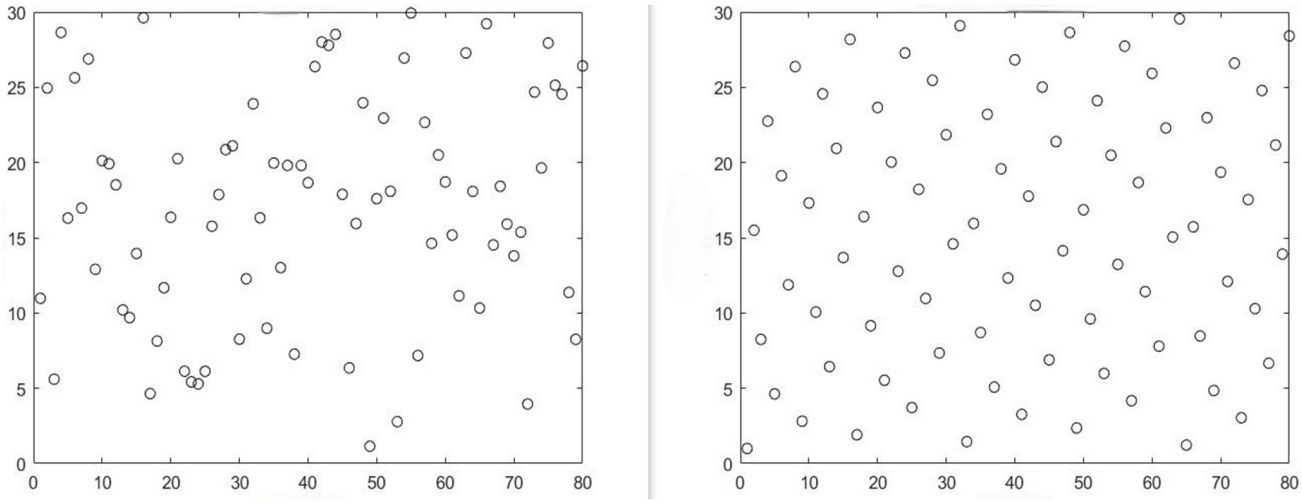

Figure 3. Comparison of population initialization effects.

3.2.2. Sine–Cosine Function-Shifting Strategy

To address the issue of local optima in the dung beetle optimization algorithm, based on the concept of symmetry, this paper introduces a sine–cosine function-based search mechanism to enhance the algorithm's global search capability and convergence speed.

When a dung beetle encounters an obstacle and cannot move forward, it updates its position using Equation (26) as follows:

$$X(\mathrm{t}+1) = \begin{cases} X(t) + r1 \times \sin r2 * |r3 * X^*(t) - X(t)| & p < 0.5 \\ X(t) + r1 \times \cos r2 * |r3 * X^*(t) - X(t)| & p \geq 0.5 \end{cases} \tag{26}$$

where $X^*(t)$ represents the position vector of the current best solution, and r2, r3, and p are uniformly distributed random numbers, with r2 belonging to [0, 2π], r3 belonging to [0, 2], and p belonging to [0, 1]. When the values of $r1\sin(r2)$ and $r1\cos(r2)$ lie within the ranges (1, 2] and [-2, -1), the current individual moves away from the optimal individual, allowing the algorithm to conduct a global search in the solution space. In the range [-1, 1], the current individual stays near the optimal individual, facilitating local exploitation by the algorithm.

The formula above updates the subsequent positions of individuals within or outside the space between their current and previous iteration positions. The random number r2 in Equation (26) determines whether the update occurs inside or outside, ensuring a balance between exploration and exploitation in the solution space. Additionally, r1, acting as a control parameter, primarily governs the amplitude of the sine–cosine function and is adaptively adjusted using Equation (27).

$$r1 = a(1 - t/T) \tag{27}$$

In the equation, a is a constant, typically set to 2, which makes the range of r_1 between [0, 2], thereby ensuring that $r1\sin(r2)$ and $r1\cos(r2)$ lie within the range [-2, 2].

In the early stages of iteration, most dung beetles conduct global searches, thus preventing the algorithm from falling into local optima. As the number of iterations increases, the value of $r1$ gradually decreases, and the fluctuations of $r1\sin(r2)$ and $r1\cos(r2)$ gradually diminish, causing dung beetles to transition from global exploration to local exploitation, thereby enhancing the precision of the search.

By introducing the sine–cosine function-based movement strategy, this paper effectively addresses the issue of local optima in the dung beetle optimization algorithm, enhancing the algorithm's global search capability and convergence speed. This provides an effective solution for dynamic environments in multi-drone trajectory planning and task allocation.

3.2.3. Population Evolution Strategy

The dung beetle optimization (DBO) algorithm divides the population into four subpopulations representing different behaviors: rolling dung beetles, breeding balls, small dung beetles, and thief dung beetles. The ratio given by the authors for these subpopulations is 6:6:7:11. After dividing the subpopulations, individuals update their positions according to different strategies. However, in traditional DBO, the proportions of these subpopulations remain fixed, resulting in a relatively balanced development and exploration capacity throughout the iteration process. Yet, in reality, the algorithm requires more local development capability in the later stages of iteration. To address this issue, this paper introduces a population evolution strategy as follows: after all individuals undergo one position update, the next iteration will involve population evolution to enhance population diversity and convergence speed and eliminate local optima. The evolution strategies for each subpopulation are as follows:

1. Evolution of the rolling dung beetle subpopulation

Individual rolling dung beetles expand the search range by increasing the absolute value of the position vector to enhance global exploration capability. Equation (29) describes the evolutionary process of the exploratory subpopulation.

$$r_4 = rand[0,1] + 1 \tag{28}$$

$$X(t+1) = X(t) * r_4 \tag{29}$$

where r_4 is a random number between [1, 2]. If the current position is a local optimum, Equation (29) can help individuals escape from the local optimum with a certain probability. This is because, under the effect of Equation (29), the absolute value of the current position vector increases, allowing individuals to explore other regions far from the current position, thus enhancing the global search capability. For a solution space centered around the origin, this movement causes the rolling dung beetle subpopulation to diffuse towards the edges of the solution space, allowing the population individuals to explore other areas. This enhances the algorithm's global search capability, as illustrated in Figure 4.

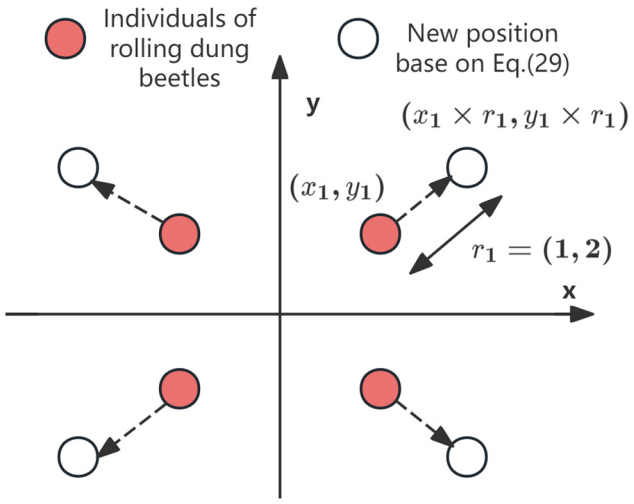

Figure 4. Diagram of the rolling dung beetle population evolution.

2. Evolution of the small ant lion subpopulation

Individuals of the small ant lion subpopulation evolve their positions by conducting deep local searches around the current best solution, as shown in Equation (31).

$$r_5 = 2 \times rand[0,1] \tag{30}$$

$$X(t+1) = X^*(t) * r_5 \tag{31}$$

where r_5 is a random number with a value ranging between [0, 2] and $X^*(t)$ represents the position vector of the current best solution. The position vector of the small ant lion will randomly increase or decrease, thus exploring the vicinity of the current best solution. This evolutionary strategy fully utilizes the positional information of the current best solution to accelerate convergence and improve accuracy.

3. Group evolution of thief cockroaches

OBL is often used to enhance the performance of metaheuristic algorithms, as it can help increase the diversity of the population. Therefore, in this paper, individual thief cockroaches evolve their positions based on OBL, as shown in Equation (33).

$$\hat{X}(t) = lb + ub - X(t) \tag{32}$$

$$X(t+1) = \begin{cases} \hat{X}(t) & if\,fit\,(\hat{X}(t))better\,than\,fit(X(t)) \\ X(t) & \text{else} \end{cases} \tag{33}$$

where lb is the lower bound of the problem space, ub is the upper bound, fit represents the objective function, and $X(t)$ is the new position based on OBL. OBL increases the diversity of the population, allowing it to escape local optima and explore new regions.

By combining the movement strategies of each subpopulation with their respective evolution strategies, the process of improving the cockroach optimization algorithm can be completed. The main steps of the improved dung beetle optimization (IDBO) algorithm are as follows:

(1) Initialize the dung beetle population and the parameters of the IDBO algorithm based on the improved method.
(2) Calculate the fitness value of all individuals according to the objective function.
(3) If the iteration count is odd, update the positions of the dung beetle subpopulations using Equations (14), (26), (19), (22), and (23); otherwise, perform population evolution for the subpopulations using Equations (29), (31), and (32).
(4) Perform boundary checks.
(5) Update the current optimal solution and its fitness value.
(6) Repeat the above steps until the maximum number of iterations is reached, then output the global optimum and its corresponding solution.

The algorithm flow of the improved cockroach optimization algorithm is shown in Figure 5.

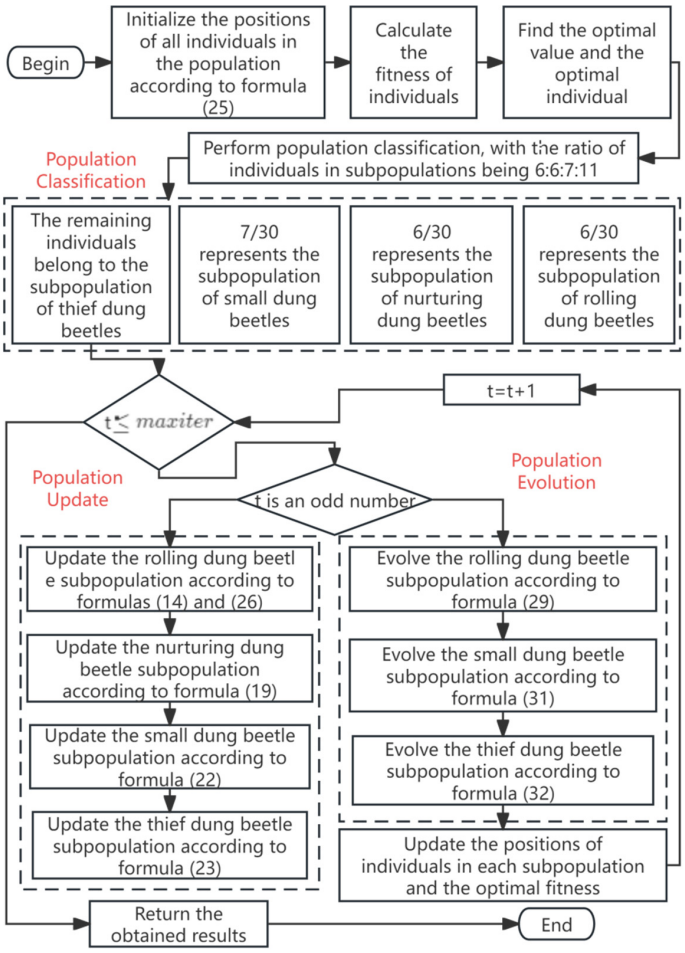

Figure 5. Flowchart of the improved dung beetle optimization algorithm.

4. Algorithm Testing

4.1. Introduction of the Tested Algorithms

In the performance testing process of intelligent optimization algorithms, test functions are often utilized to evaluate the global and local search capabilities of the algorithms. The International Evolutionary Computation Conference 2005 (CEC2005) provides a set of 23 commonly used benchmark test functions for assessing the performance of optimization algorithms. These functions encompass various problem types, including unimodal, multimodal, and fixed-dimensional multimodal functions. The proposed algorithm's search performance is evaluated using the aforementioned test functions. Among them, the first seven benchmark functions are unimodal, the eighth to thirteenth are multimodal, and the rest are composite benchmark functions. In addition, the CEC 2020 competition provides practical engineering problem models commonly used in real-world scenarios, some of which were selected for function testing in this study. To verify the effectiveness of the improved dung beetle optimization algorithm, comparisons were conducted with nine commonly used metaheuristic algorithms across multiple dimensions. These nine algorithms include the Multi-Verse Optimizer (MVO) [26,27], Ant Lion Optimizer (ALO) [28,29], Sine–Cosine Algorithm (SCA) [30–32], Grey Wolf Optimizer (GWO) [33], Whale Optimization Algorithm (WOA) [34–36], Bayesian Optimization (BO) [37], Moth–Flame Optimization (MFO) [38], Water Hyacinth and Water Lily-based Optimization Algorithm (WWPA) [39], and the dung beetle optimization (DBO) [25].

The Multi-Verse Optimizer (MVO) simulates the behavior of multiverse populations under the combined influence of white holes, black holes, and wormholes. Similar to other swarm intelligence optimization algorithms, MVO includes two main phases: the exploration phase, governed by white and black holes, and the exploitation phase, managed by wormholes. The Ant Lion Optimizer (ALO), proposed by Mirjalili in 2015, is a novel metaheuristic swarm intelligence algorithm that achieves global optimization by mimicking the hunting mechanism of ant lions. Before hunting, an ant lion digs a funnel-shaped trap in sandy soil and waits for prey. Once an ant falls into the trap, the ant lion quickly captures it and repairs the trap for the next hunt. The Sine–Cosine Algorithm (SCA), introduced by Seyedali Mirjalili in 2016, is a new bio-inspired optimization algorithm that solves optimization problems using sine and cosine mathematical models and random candidate solutions. Although SCA is simple in structure, has few parameters, and is easy to implement, it suffers from low optimization precision, susceptibility to local optima, and slow convergence speed. The Grey Wolf Optimizer (GWO), developed by Mirjalili et al. in 2014, is a swarm intelligence optimization algorithm that mimics the leadership hierarchy and hunting mechanism of grey wolves. GWO boasts strong convergence performance, few parameters, and ease of implementation, finding extensive application in job scheduling, parameter optimization, and image classification. The Whale Optimization Algorithm (WOA), introduced by Mirjalili and Lewis in 2016, is a novel swarm intelligence optimization method inspired by the hunting behavior of humpback whales. WOA includes searching for prey, encircling, and spiral position updating stages, with its population update mechanism being independent and not requiring manual setting of various control parameter values. Compared to other swarm intelligence optimization algorithms, WOA's novel structure and fewer control parameters make it perform well in solving numerous numerical optimization and engineering problems. Bayesian Optimization (BO) is a method for finding the maximum of a black-box function by guessing its shape to locate an acceptable maximum. The Moth–Flame Optimization (MFO), proposed by Mirjalili in 2015, is a swarm intelligence optimization algorithm inspired by the night flight paths of moths. MFO is characterized by strong parallel optimization capabilities and overall performance, demonstrating high global search ability in solving non-convex functions.

The Water Hyacinth and Water Lily-based Optimization Algorithm (WWPA) simulates the natural behavior of water hyacinth plants during hunting expeditions. WWPA uses plants as search agents to solve optimization problems and evaluates its performance using 23 different types of unimodal and multimodal objective functions. For unimodal functions, WWPA shows strong exploitation ability, approaching optimal solutions, while for multimodal functions, WWPA demonstrates strong exploration capability, identifying major optimal regions in the search space.

The corresponding parameters of the tested algorithms are listed in Table 1.

Table 1. Parameters of each tested algorithm.

MVO	ALO	SCA	GWO	WOA	BO	MFO	WWPA	DBO	IDBO
Max_iter = 500 S = 30 WEP_Max = 1 WEP_Min = 0.2	Max_iter = 500 S = 30	Max_iter = 500 S = 30 a = 2	Max_iter = 500 S = 30	Max_iter = 500 S = 30 b = 1	Max_iter = 500 S = 30 p = 0.8 power_exponent = 0.1 sensory_modality = 0.01	Max_iter = 500 S = 30 b = 1	Max_iter = 500 S = 30	Max_iter = 500 S = 30 b = 1 Subpopulation Proportion = 6: 6; 7; 11	Max_iter = 500 S = 30 b = 1 Subpopulation Proportion = 6: 6; 7; 11

4.2. CEC 2005 Function Testing

The CEC 2005 test functions, as shown in Table 2, were used to compare the convergence accuracy and convergence speed of various algorithms. For each test function, the algorithm with the highest convergence accuracy had its final value bolded. If multiple algorithms achieved the optimal value, their convergence accuracy was compared. The test results are presented in Table 3.

Table 2. Test functions.

Function	Dimension	Dim	Function
Sphere Function	$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]$
Schwefel's Problem 2.22	$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$[-10, 10]$
Schwefel's Problem 1.2	$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	30	$[-100, 100]$
Schwefel's Problem 2.21	$f_4(x) = \max_i \{ \lvert x_i \rvert, 1 \le i \le n \}$	30	$[-100, 100]$
Generalized Rosenbrock's Function	$f_5(x) = \sum_{i=1}^{n-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + (x_i - 1)^2 \right]$	30	$[-30, 30]$
Step Function	$f_6(x) = \sum_{i=1}^{n} \left([x_i + 0.5] \right)^2$	30	$[-100, 100]$
Quartic Function i.e., Noise	$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$	30	$[-1.28, 1.28]$
Generalized Schwefel's Problem 2.26	$f_8(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert} \right)$	30	$[-500, 500]$
Generalized Rastrigin's Function	$f_9(x) = \sum_{i=1}^{n} \left[x_i^2 - 10 \cos(2\pi x_i) + 10 \right]$	30	$[-5.12, 5.12]$
Ackley's Function	$f_{10}(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2} \right) - \exp\left(\frac{1}{n} \sum_{i=1}^{n} \cos(2\pi x_i) \right) + 20 + e$	30	$[-32, 32]$
Generalized Griewank's Function	$f_{11}(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1$	30	$[-600, 600]$
Generalized Penalized Function 1	$f_{12}(x) = \frac{\pi}{n} \left\{ 10 \sin(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 \left[1 + 10 \sin^2(\pi y_{i+1}) \right] + (y_n - 1)^2 \right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$	30	$[-50, 50]$
Generalized Penalized Function 2	$f_{13}(x) = 0.1 \{ \sin^2(3\pi x_i) + \sum_{i=1}^{n} (x_i - 1)^2 \left[1 + \sin^2(3\pi x_i + 1) \right] + (x_n - 1)^2 \left[1 + \sin^2(2\pi x_n) \right] \} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30	$[-50, 50]$
Shekel's Foxholes Function	$f_{14}(x) = \left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6} \right)^{-1}$	2	$[-65, 65]$

Table 2. *Cont.*

Function	Dimension	Dim	Function
Kowalik's Function	$f_{15}(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_i(b_i^2 + b_1 x_2)}{b_i^2 + b_1 x_3 + x_4}\right]^2$	4	$[-5, 5]$
Six-Hump Camel-Back Function	$f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5, 5]$
Branin Function	$f_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	2	$[-5, 5]$
Goldstein-Price Function	$f_{18}(x) = \left[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \times \left[30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$	2	$[-2, 2]$
Hartman's Family	$f_{19}(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$	3	$[1, 3]$
	$f_{20}(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$	6	$[0, 1]$
Shekel's Family	$f_{21}(x) = -\sum_{i=1}^{5}\left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4	$[0, 10]$
	$f_{22}(x) = -\sum_{i=1}^{7}\left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4	$[0, 10]$
	$f_{23}(x) = -\sum_{i=1}^{10}\left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4	$[0, 10]$

Table 3. Algorithm testing results.

Function	MVO	ALO	SCA	GWO	WOA	BO	MFO	WWPA	DBO	IDBO
F1	0.006957	1.381×10^{-8}	4.124×10^{-13}	3.627×10^{-58}	6.439×10^{-78}	1.163×10^{-11}	8.742×10^{-16}	1.3791	4.58×10^{-110}	$\mathbf{1.82 \times 10^{-141}}$
F2	0.043117	0.63247	1.773×10^{-9}	6.832×10^{-34}	5.230×10^{-58}	5.317×10^{-9}	2.856×10^{-9}	0.214	1.218×10^{-57}	$\mathbf{2.346 \times 10^{-74}}$
F3	0.07451	0.033541	0.021413	3.288×10^{-25}	883.8813	1.191×10^{-11}	0.050041	0.89166	2.75×10^{-115}	$\mathbf{4.64 \times 10^{-125}}$
F4	0.071496	0.001579	1.832×10^{-5}	3.341×10^{-18}	8.377×10^{-6}	5.650×10^{-9}	3.3256	0.42419	1.557×10^{-42}	$\mathbf{2.779 \times 10^{-68}}$
F5	8.6316	5.1402	7.4734	7.1753	6.6719	8.9023	6.8714	22.5056	4.9417	**4.9083**
F6	0.0043393	2.478×10^{-9}	0.20331	1.242×10^{-6}	0.0006893	1.4444	1.047×10^{-12}	3.2038	8.811×10^{-28}	$\mathbf{1.217 \times 10^{-28}}$
F7	0.002397	0.02933	0.003236	0.0005233	0.001397	0.001025	0.006826	0.08167	0.0003603	**0.0003000**
F8	−3063.084	−2087.107	−2112.222	−2865.371	−3832.167	−2711.595	−4071.390	−3832.167	−3971.477	**−4092.493**
F9	20.8998	12.9344	1.190×10^{-7}	3.1729	0	31.368	17.9092	16.5128	0	**0**
F10	1.164	0.0001155	5.109×10^{-8}	4.440×10^{-15}	7.993×10^{-15}	2.365×10^{-9}	1.682×10^{-7}	0.77348	8.881×10^{-16}	$\mathbf{8.881 \times 10^{-16}}$
F11	0.30571	0.12298	9.103×10^{-15}	0.043854	0.24782	2.068×10^{-13}	0.076316	0.99407	0	**0**
F12	0.0005551	2.177×10^{-8}	0.058596	8.214×10^{-7}	0.030224	0.16319	4.270×10^{-14}	7.785×10^{-5}	1.032×10^{-9}	$\mathbf{1.912 \times 10^{-26}}$
F13	0.0012285	4.591×10^{-8}	0.37296	4.38×10^{-6}	0.015184	0.26016	0.010987	0.1332	0.097371	$\mathbf{3.647 \times 10^{-31}}$
F14	0.998	1.992	2.9821	1.992	2.9821	1.0839	0.998	12.6706	0.998	**0.998**
F15	0.0006549	0.0007640	0.001443	0.0004542	0.0004067	0.0004929	0.0007826	0.009032	0.0004176	**0.0003498**
F16	−1.0316	−1.0316	−1.0315	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316
F17	0.39789	0.39789	0.39883	0.39789	0.3979	0.39864	0.39789	0.96041	0.40243	**0.39789**
F18	3	3	3	3	3	3.1595	3	36.015	3	**3**
F19	−3.8628	−3.8628	−3.8522	−3.8627	−3.8563	−3.8563	−3.8628	−3.2298	−3.8549	**−3.8628**
F20	−3.322	−3.322	−1.9183	−3.322	−3.3147	−3.0459	−3.2031	−1.566	−3.1376	**−3.322**
F21	−5.1007	−5.1008	−0.49652	−10.1502	−10.1391	−4.4941	−2.6829	−5.0449	−2.6305	**−10.1532**
F22	−10.4027	−10.4029	−0.9061	−10.3975	−3.7215	−2.8792	−10.4029	−3.4918	−10.4029	**−10.4029**
F23	−10.5362	−5.1285	−4.754	−10.5338	−5.1283	−3.301	−3.8354	−2.9159	−10.5364	**−10.5364**

The analysis of the test functions included unimodal test functions, multimodal test functions, and composite benchmark test functions. The iterative processes for these three types of functions are shown in Figures 6–8, respectively.

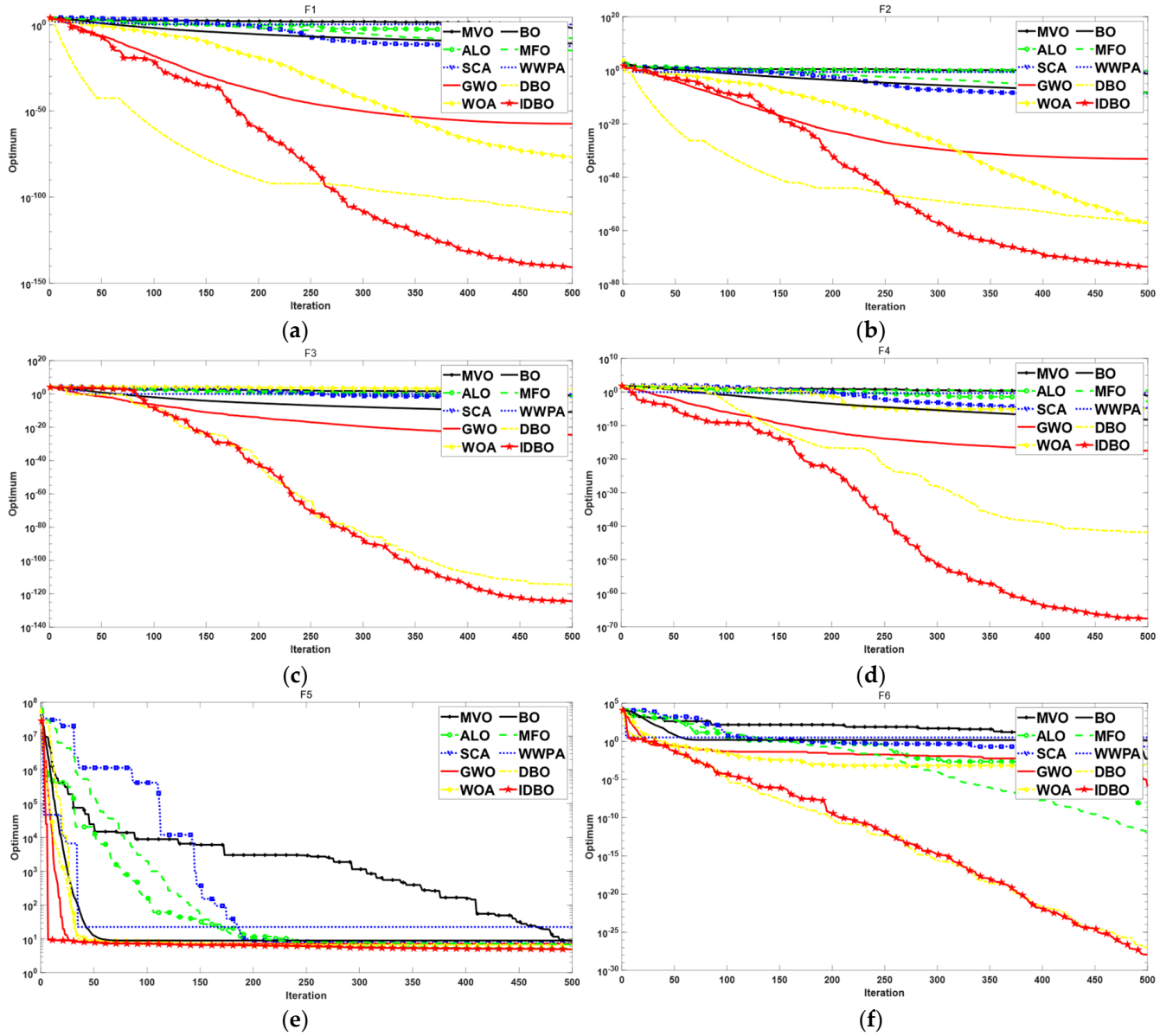

Figure 6. Convergence trend of unimodal test functions: (**a**) Sphere Function, (**b**) Schwefel's Problem 2.22, (**c**) Schwefel's Problem 1.2, (**d**) Schwefel's Problem 2.21, (**e**) Generalized Rosenbrock's Function, (**f**) Step Function.

As shown in the figures, for unimodal test functions, the improved dung beetle optimization (DBO) algorithm demonstrated significantly better performance than other algorithms over 500 iterations across seven functions. It not only converged faster but also achieved higher accuracy.

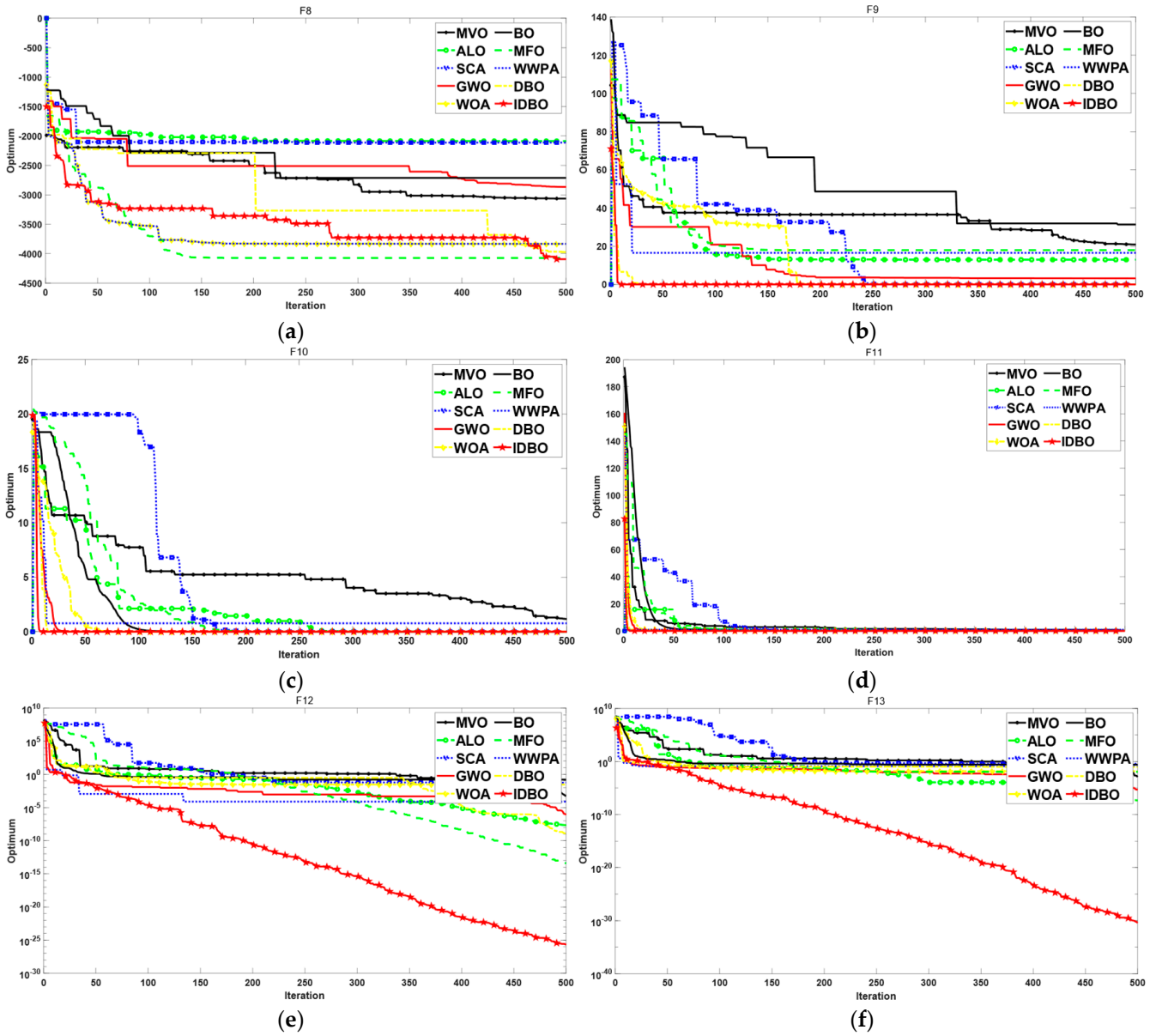

Figure 7. Convergence trend of multimodal test functions: (**a**) Generalized Schwefel's Problem 2.26, (**b**) Generalized Rastrigin's Function, (**c**) Ackley's Function, (**d**) Generalized Griewank's Function, (**e**) Generalized Penalized Function 1, (**f**) Generalized Penalized Function 2.

The performance of the improved cockroach optimization algorithm (ICROA) on six multimodal test functions was significantly superior to other algorithms when subjected to 500 iterations, as illustrated in the above figures. In the case of functions F9 and F11, both the cockroach optimization algorithm (COA) and ICROA converged to the optimal solution; however, ICROA exhibited notably faster convergence rates compared to the original COA. Moreover, for the remaining multimodal functions, ICROA demonstrated higher convergence accuracy than other algorithms. Even when other functions were trapped in local optima, ICROA continued to search for the global optimum, showcasing superior convergence performance.

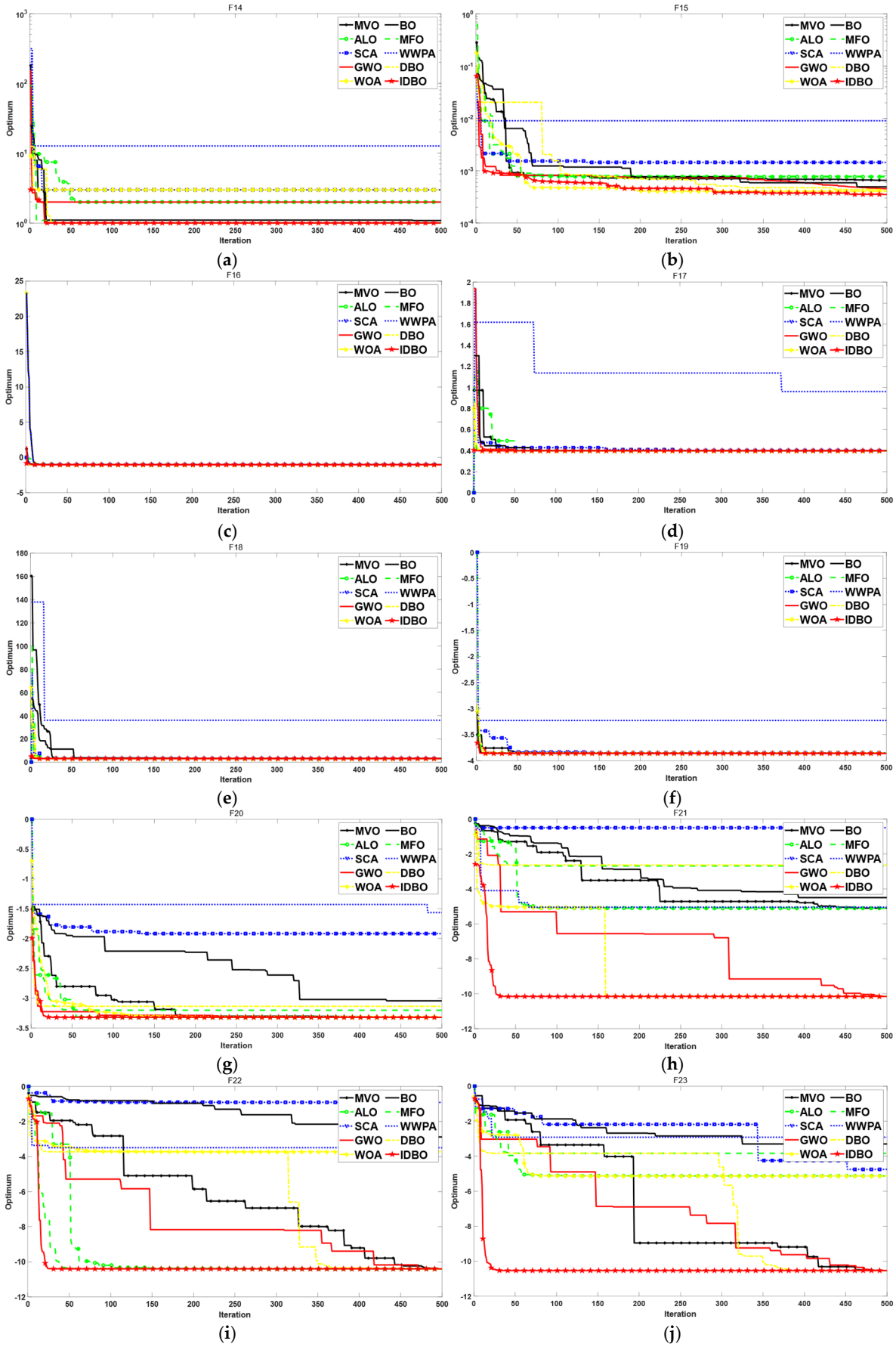

Figure 8. Convergence trend of composite benchmark test functions: (**a**) Shekel's Foxholes Function, (**b**) Kowalik's Function, (**c**) Six-Hump Camel-Back Function, (**d**) Branin Function, (**e**) Goldstein–Price

Function, (**f**) Hartman's Family, (**g**) Hartman's Family 2, (**h**) Shekel's Family 1, (**i**) Shekel's Family 2, and (**j**) Shekel's Family 3.

As depicted in the above figures, the improved cockroach optimization algorithm (ICROA) consistently converged to the optimal solution in most cases of composite benchmark test functions. While some other algorithms also achieved convergence to the optimum in certain functions, ICROA consistently exhibited the fastest convergence speed, thus demonstrating superior convergence performance.

A comparative analysis involving 10 different algorithms revealed that ICROA outperformed the other algorithms in 22 out of 23 benchmark test functions. For the first seven single-modal benchmark test functions and the subsequent eight multimodal benchmark test functions, ICROA demonstrated the best convergence performance in each function, confirming its adaptability in handling these two types of functions. Across the 10 composite benchmark test functions, ICROA generally obtained superior solutions compared to other algorithms. Although there was a slight deficiency in convergence speed observed in function F16, overall, it proved to be an effective algorithm for solving composite functions.

4.3. CEC 2020 Engineering Problem Testing

The CEC 2020 test set consists of real-world optimization problems, mathematical models, and artificially constructed benchmark problems, characterized by a certain level of complexity and diversity, covering different types of optimization problems. This study selected six common engineering problem models from the CEC 2020 dataset to test various optimization algorithms. The parameter settings for all algorithms were consistent with those described earlier. The six CEC 2020 models are detailed as follows.

(1) Shifted and Rotated Bent Cigar Function

Objective functions:

$$\min\begin{cases} f_1(x) = gx_1 \\ f_2(x) = g\sqrt{1-(f_1/g)^2} \end{cases}$$

where $g = 1 + \sum_{i=2}^{n}\left(1 - \exp\left(-10\left(z_i - 0.5 - \frac{i-1}{2n}\right)^2\right)\right)$, $z_i = x_i^{n-2}$, and n is the number of decision variables and is set to 25.

Objective constraints:

$$\begin{aligned} C_1 &: f_1^2 + f_2^2 - (1.7 - 0.2\sin(2l))^2 \le 0, \\ C_2 &: \left(1 + 0.5\sin\left(6(0.5\pi - 2|l - 0.25\pi|)^3\right)\right)^2 - f_1^2 - f_2^2 \le 0, \\ C_3 &: \left(1 - 0.45\sin\left(6(0.5\pi - 2|l - 0.25\pi|)^3\right)\right)^2 - f_1^2 - f_2^2 \le 0, \end{aligned}$$

where $l = \arctan(f_2/f_1)$.

The search space is $0 \le x_i \le 1, i = 1, 2, \ldots, 25$.

(2) Shifted and Rotated Lunacek bi-Rastrigin Function

Objective functions:

$$\min\begin{cases} f_1(x) = gx_1^n \\ f_2(x) = g\left(1-(f_1/g)^2\right) \end{cases}$$

where $g = 1 + \sum_{i=2}^{n}\left(1.5 + \frac{0.1}{n}z_i^2 - 1.5\cos(2\pi z_i)\right)$, $z_i = 1 - \exp\left(-10\left(x_i - \frac{i-1}{n}\right)^2\right)$, and n is the number of decision variables and is set to 25.

Objective constraints:

$$C_1 : (2 - 4f_1^2 - f_2)(2 - 8f_1^2 - f_2) \geq 0,$$
$$C_1 : (2 - 2f_1^2 - f_2)(2 - 16f_1^2 - f_2) \leq 0,$$
$$C_1 : (1 - f_1^2 - f_2)(1.2 - 1.2f_1^2 - f_2) \leq 0.$$

The search space is $0 \leq x_i \leq 1, i = 1, 2, \ldots, 25$.

(3) Expanded Rosenbrock's plus Griewangk's Function

Objective functions:

$$\min\begin{cases} f_1(x) = gx_1 \\ f_2(x) = g(5 - \exp(f_1/g) - 0.5|\sin(3\pi f_1/g)|) \end{cases}$$

where $g = 1 + \sum_{i=2}^{n}\left(1.5 + \frac{0.1}{n}z_i^2 - 1.5\cos(2\pi z_i)\right)$, $z_i = 1 - \exp\left(-10\left(x_i - \frac{i-1}{n}\right)^2\right)$, and n is the number of decision variables and is set to 25.

Objective constraints:

$$C_1 : (5 - \exp(f_1) - 0.5\sin(3\pi f_1) - f_2)(5 - (1 + 0.4f_1) - 0.5\sin(3\pi f_1) - f_2) \leq 0,$$
$$C_2 : \left(5 - \left(1 + f_1 + 0.5f_1^2\right) - 0.5\sin(3\pi f_1) - f_2\right)(5 - (1 + 0.7f_1) - 0.5\sin(3\pi f_1) - f_2) \geq 0.$$

The search space is $0 \leq x_i \leq 1.5,\ i = 1, 2, \ldots, 25$.

(4) Composition Function 1

Objective functions:

$$\min\begin{cases} f_1(x) = x_1 \\ f_2(x) = g\left(1 - \sqrt{f_1}/g\right) \end{cases}$$

where $g = (x_2 - 10)^2 + 5(x_3 - 12)^2 + x_4^4 + 3(x_5 - 11)^2 + 10x_6^6 + 7x_7^2 + x_8^4 - 4x_7x_8 - 10x_7 - 8x_8 - 679.6300573745$.

Objective constraints:

$$C_1 = f_1 + f_2 - 1 \geq 0;$$
$$C_2 = f_1 + f_2 - 1 - |\sin(10\pi(f_1 - f_2 + 1))| \geq 0.$$

Decision constraints:

$$C_3 = -127 + 2x_2^2 + 3x_3^4 + x_4 + 4x_5^2 + 5x_6 \leq 0;$$
$$C_4 = -282 + 7x_2 + 3x_3 + 10x_4^2 + x_5 - x_6 \leq 0;$$
$$C_5 = -196 + 23x_2 + x_3^2 + 6x_7^2 - 8x_8 \leq 0;$$
$$C_6 = 4x_2^2 + x_3^2 - 3x_2x_3 + 2x_4^2 + 5x_7 - 11x_8 \leq 0.$$

- The search space is $0 \leq x_1 \leq 1, -10 \leq x_i \leq 10,\ i = 2, 3, \ldots, 8$.

(5) Composition Function 2

Objective functions:

$$\min\begin{cases} f_1(x) = x_1 \\ f_2(x) = g\left(1 - \sqrt{f_1}/g\right) \end{cases}$$

where $g = x_2 - 192.724510070035$.

Objective constraints:

$$C_1 : f_1 + f_2 - 1 \geq 0;$$
$$C_2 : f_1 + f_2 - 1 - |\sin(10\pi(f_1 - f_2 + 1))| \geq 0;$$
$$C_3 : (f_1 - 0.8)(f_2 - 0.6) \leq 0.$$

Decision constraints:

$$C_4 : -x_2 + 35x_3^{0.6} + 35x_4^{0.6} \leq 0;$$
$$C_5 : -300x_4 + 7500x_6 - 7500x_7 - 25x_5x_6 + 25x_5x_7 + x_4x_5 = 0;$$
$$C_6 : 100x_3 + 155.365x_5 + 2500x_8 - x_3x_5 - 25x_5x_8 - 15536.5 = 0;$$
$$C_7 : -x_6 + \log(-x_5 + 900) = 0;$$
$$C_8 : -x_7 + \log(x_5 + 300) = 0;$$
$$C_9 : -x_8 + \log(-2x_5 + 700) = 0.$$

The search space is $0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1000, 0 \leq x_3, x_4 \leq 40, 100 \leq x_5 \leq 300, 6.3 \leq x_6 \leq 6.7, 5.9 \leq x_7 \leq 6.4, 4.5 \leq x_8 \leq 6.25.$

(6) Composition Function 3

Objective functions:

$$\min \begin{cases} f_1(x) = g x_1 x_2 \\ f_2(x) = g x_1 (1 - x_2) \\ f_3(x) = g(1 - x_1) \end{cases}$$

where $g = x_3 + x_4 + x_5 - 7048.2480205286$.

Objective constraints:

$$C_1 : (f_3 - 0.4)(f_3 - 0.6) \geq 0.$$

Decision constraints:

$$C_2 : -1 + 0.0025(x_6 + x_8) \leq 0;$$
$$C_3 : -1 + 0.0025(x_7 + x_9 - x_6) \leq 0;$$
$$C_4 : -1 + 0.01(x_{10} - x_7) \leq 0;$$
$$C_5 : -x_3x_8 + 833.33252x_6 + 100x_3 - 8333.333 \leq 0;$$
$$C_6 : -x_4x_9 + 1250x_7 + x_4x_6 - 1250x_6 \leq 0;$$
$$C_7 : -x_5x_{10} + 1250000 + x_5x_7 - 2500x_7 \leq 0.$$

- The search space is $0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1, 500 \leq x_3 \leq 1000, 1000 \leq x_4 \leq 2000, 5000 \leq x_5 \leq 6000$, and $100 \leq x_i \leq 500,\ i = 6, 7, \ldots, 10$.

The aforementioned models were used to test the algorithms, with the results presented in Table 4 and the iteration process illustrated in Figure 9.

Table 4. CEC 2020 algorithm test results.

Function	MVO	ALO	SCA	GWO	WOA	BO	MFO	WWPA	DBO	IDBO
F1	100	100	100	100	100	100	100	100	100	100
F2	720.01	720	700.0001	740.1549	700	700	720.0001	700.0964	700	**700**
F3	1900	1900	1900	1900	1900	1900	1900	1900	1900	1900
F4	2300.003	2300	2300	2300	2300	2300	2300.0011	2300	2300	**2300**
F5	2550.0032	2550	2550	2550	2550	2550	2550	2550	2550	**2550**
F6	2700.0034	2700	2700	2700	2700	2700	2700	2700	2700	**2700**

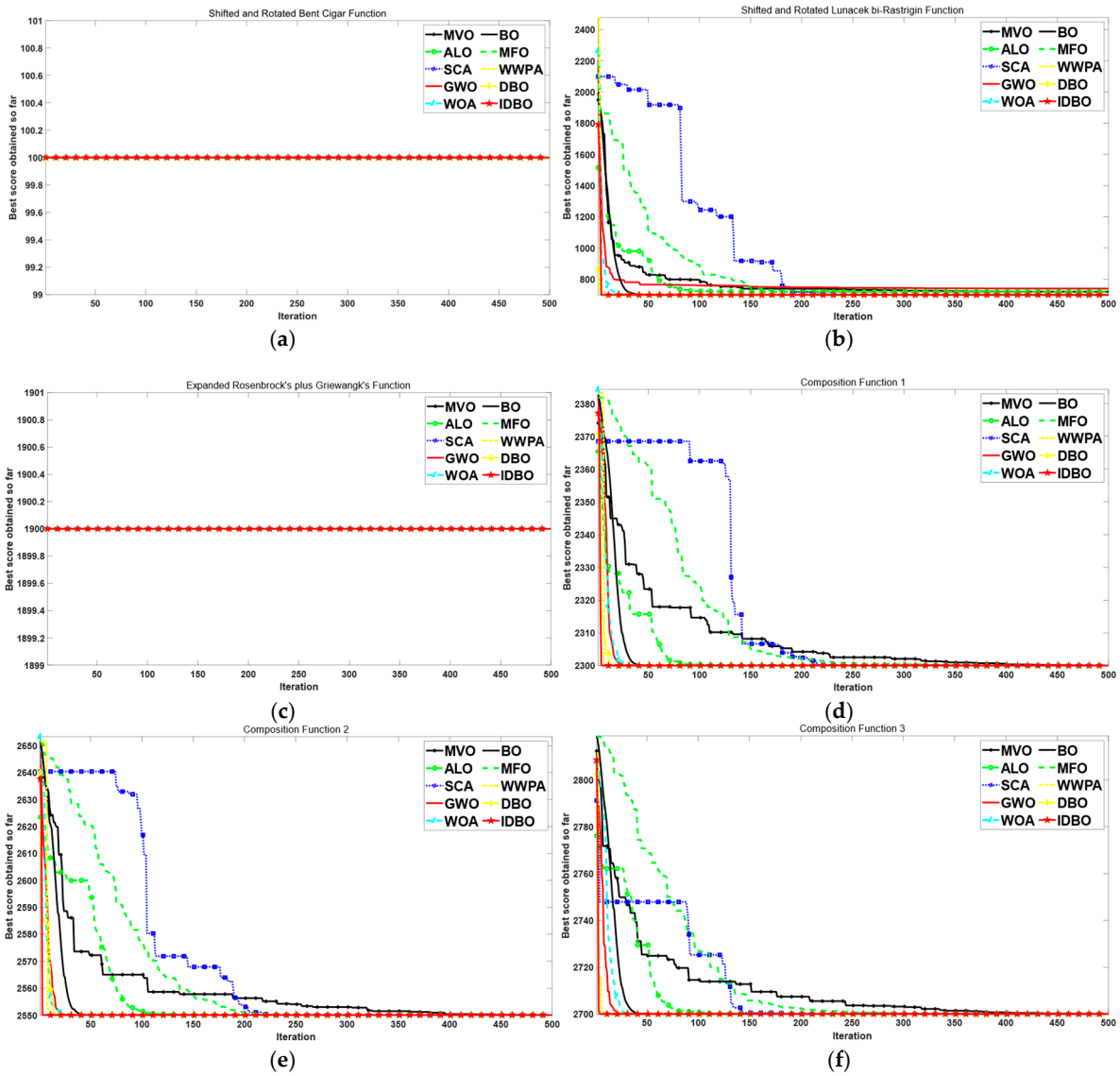

Figure 9. Convergence trends of CEC 2020 models: (**a**) Shifted and Rotated Bent Cigar Function, (**b**) Shifted and Rotated Lunacek bi-Rastrigin Function, (**c**) Expanded Rosenbrock's plus Griewangk's Function, (**d**) Composition Function 1, (**e**) Composition Function 2, (**f**) Composition Function 3.

As shown in the figure above, most of the algorithms tested in each model converged to the optimal value, with IDBO consistently achieving the fastest convergence speed.

In summary, by testing and comparing the improved dung beetle optimization (IDBO) algorithm with other algorithms using 23 benchmark functions of different dimensions and six engineering problem models, the high accuracy and fast convergence performance of IDBO were effectively validated. When applied to multi-drone path planning and task allocation in dynamic environments, IDBO effectively avoided the drawbacks of slow convergence speed and low convergence accuracy.

5. Drone Flight Path Planning Simulation Experiment

Using Matlab 2021a, a simulation platform was constructed to initialize the terrain map for unmanned aerial vehicle (UAV) trajectory planning. The terrain map spanned 1000 m × 1000 m × 400 m, with the starting coordinates at (200,100,150) and the endpoint coordinates at (800,800,150). The number of waypoints was set to 10. Relevant constraint parameters included a flight height range of [100, 200], a population size of 30, and a

maximum iteration count of 500. The experiments involved comparing the performance of the MVO, ALO, WOA, DBO, and IDBO algorithms. The algorithm parameters were consistent with those mentioned earlier. Testing was conducted in three scenarios. In Scenario 1 and Scenario 2, the terrain was the same, with three and nine threats, respectively, and the three cost weights were 5, 10, and 1. In Scenario 3, the terrain was different from the first two scenarios, with nine threats and the three cost weights set to 5, 10, and 100. The specific terrain maps are shown in Figures 10–12. The red area in the figure represents the threat range.

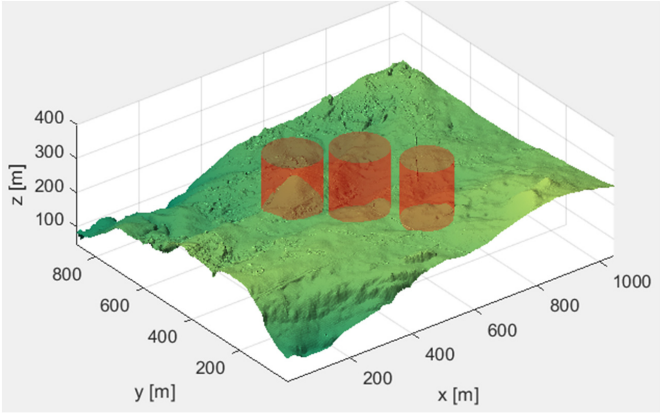

Figure 10. Experimental scene 1.

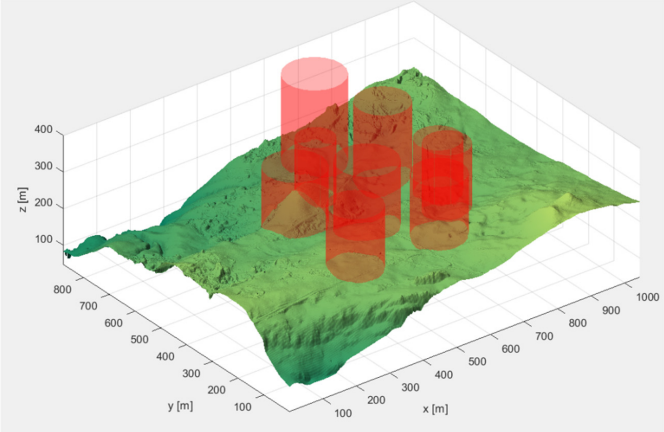

Figure 11. Experimental scene 2.

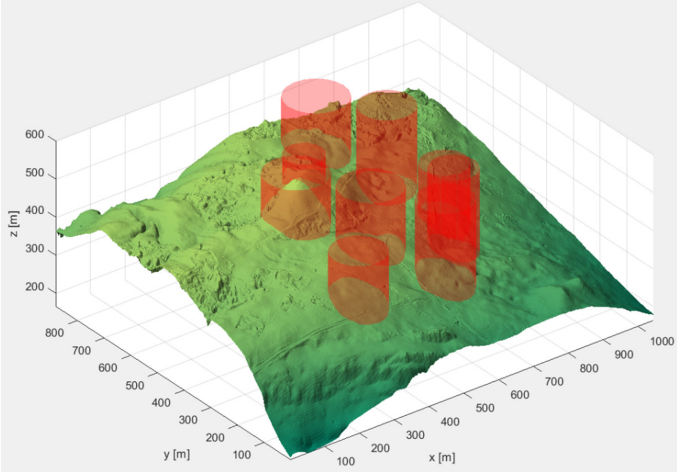

Figure 12. Experimental scene 3.

5.1. Test Results for Scenario 1

In Scenario 1, with three threats, a UAV path planning test was conducted, where the weights for flight distance cost, flight altitude cost, and threat/obstacle cost were set to 5, 10, and 1, respectively. The test results are shown in Figures 13–16, which include the iteration charts for each algorithm, the 3D view of the UAV path, the top-down view of the UAV path, and the side view of the UAV path. The correspondence between the different colored lines in each figure and the algorithms is consistent with the legend in Figure 13.

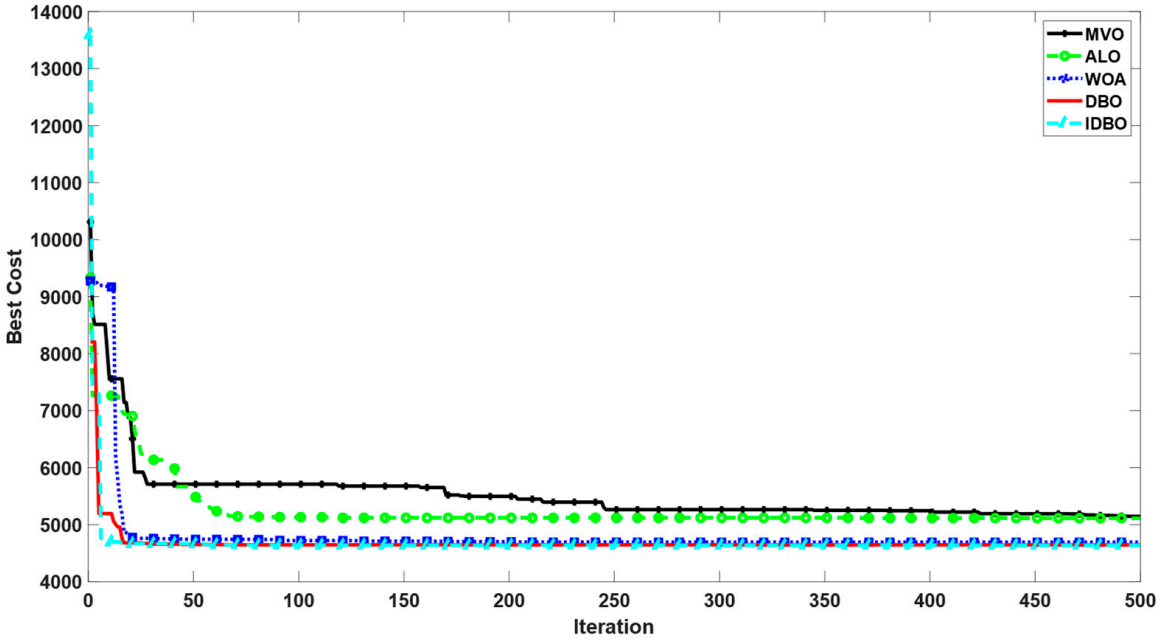

Figure 13. Experiment 1—algorithm iteration chart.

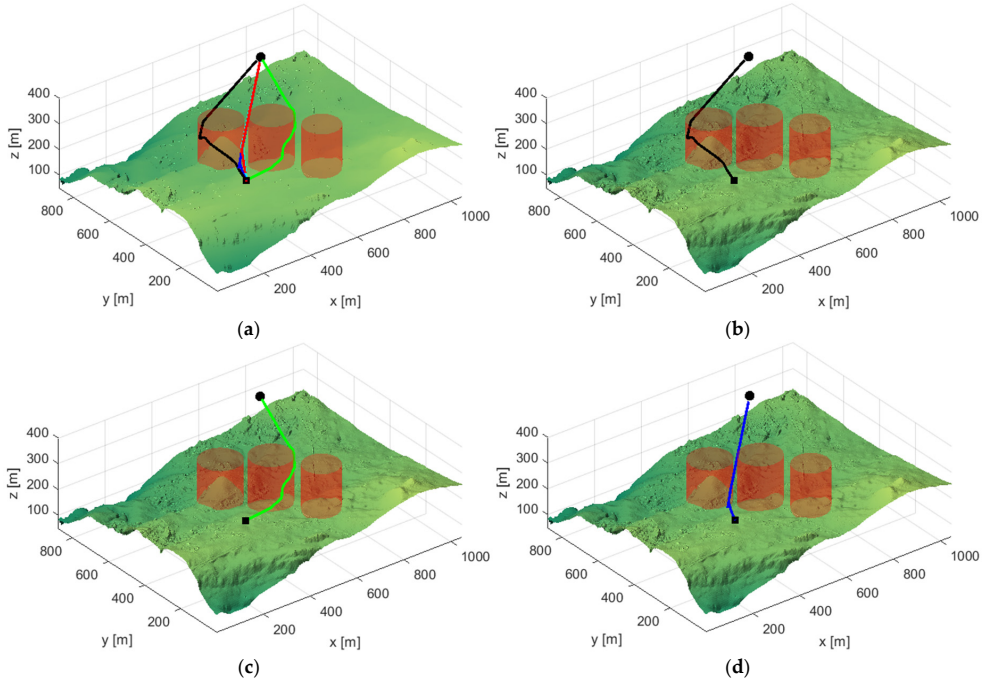

Figure 14. *Cont.*

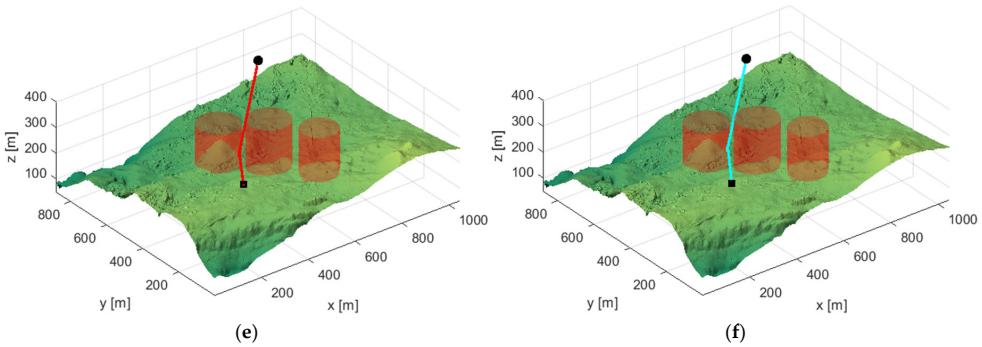

Figure 14. Experiment 1—UAV path 3D view. (**a**) 3D view—overall, (**b**) 3D view—MVO, (**c**) 3D view—ALO, (**d**) 3D view—WOA, (**e**) 3D view—DBO, (**f**) 3D view—IDBO.

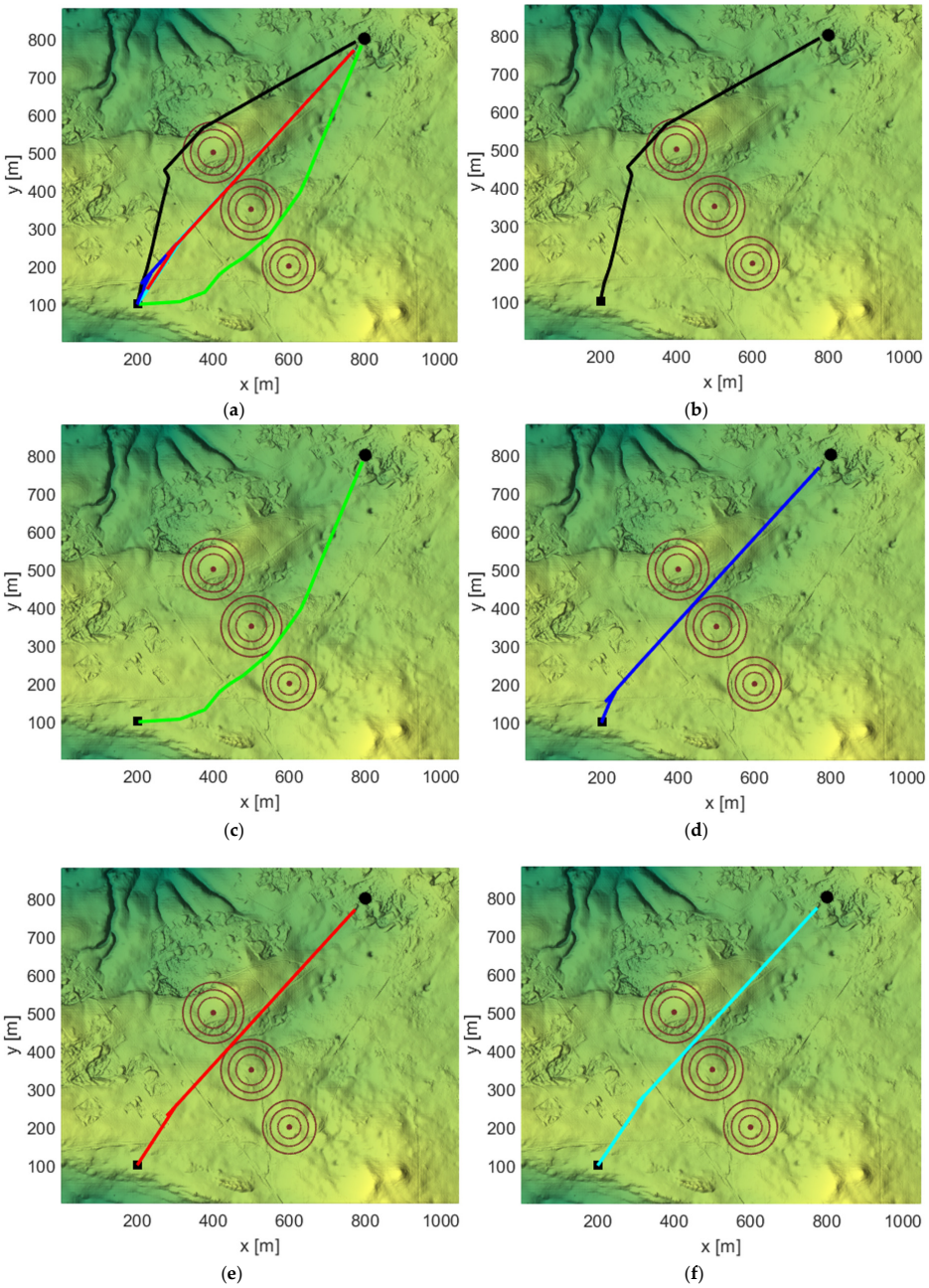

Figure 15. Experiment 1—UAV path top-down view. (**a**) Top-down view—overall, (**b**) top-down view—MVO, (**c**) top-down view—ALO, (**d**) top-down view—WOA, (**e**) top-down view—DBO, (**f**) top-down view—IDBO.

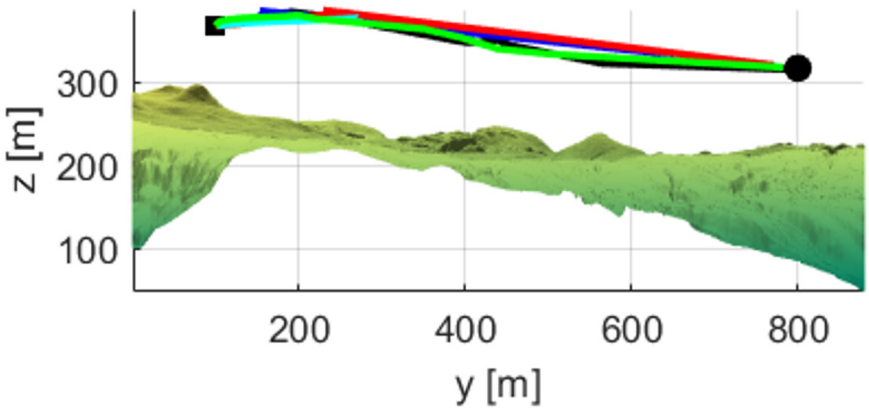

Figure 16. Experiment 1—UAV path side view.

The optimal values obtained by each algorithm are shown in Table 5.

Table 5. Experiment 1—optimal values for each algorithm.

Algorithm	MVO	ALO	WOA	DBO	IDBO
Flight Distance Cost	1014.4	1003.8	932.1	927.6	**926.7**
Flight Altitude Cost	5.6306	8.5913	2.4675	$\mathbf{5.1159 \times 10^{-13}}$	6.0083×10^{-7}
Threat/Obstacle Cost	15.681	10	7.315	8.437	**6.928**
Total Cost	5143.9	5114.7	4692.4	4646.4	**4640.5**
Total Path Deviation Angle	179.0531	256.3046	153.9947	84.5057	**37.2063**
Computational Time	1.3999 s	1.2816 s	**0.6512 s**	0.6858 s	1.3201 s

IDBO not only exhibited the fastest convergence speed but also provided the optimal solution among the five algorithms. Its flight distance cost, threat/obstacle cost, and total cost were the lowest of all the solutions, and its total path deviation angle was significantly smaller than that of the other solutions. This indicates that IDBO outperformed the other four algorithms in UAV path planning when the number of obstacles was relatively low.

5.2. Test Results for Scenario 2

In Scenario 2, with nine threats, a UAV path planning test was conducted, where the weights for flight distance cost, flight altitude cost, and threat/obstacle cost were set to 5, 10, and 1, respectively. The test results are shown in Figures 17–20, which include the iteration charts for each algorithm, the 3D view of the UAV path, the top-down view of the UAV path, and the side view of the UAV path. The correspondence between the different colored lines in each figure and the algorithms is consistent with the legend in Figure 17.

The optimal values obtained by each algorithm are shown in Table 6.

Table 6. Experiment 2—optimal values of each algorithm.

Algorithm	MVO	ALO	WOA	DBO	IDBO
Flight Distance Cost	1084.7	1190.8	**924.6**	1271.3	951.1
Flight Altitude Cost	15.7153	**4.9325**	26.3965	200.0000	22.2238
Threat/Obstacle Cost	24.326	18.551	Inf	**10**	12.773
Total Cost	5605.1	6021.7	Inf	8366.7	**4990.3**
Total Path Deviation Angle	226.0586	136.7557	**0**	226.3384	152.4937
Computational Time	2.4139 s	2.2997 s	**0.9613 s**	1.2121 s	2.0064 s

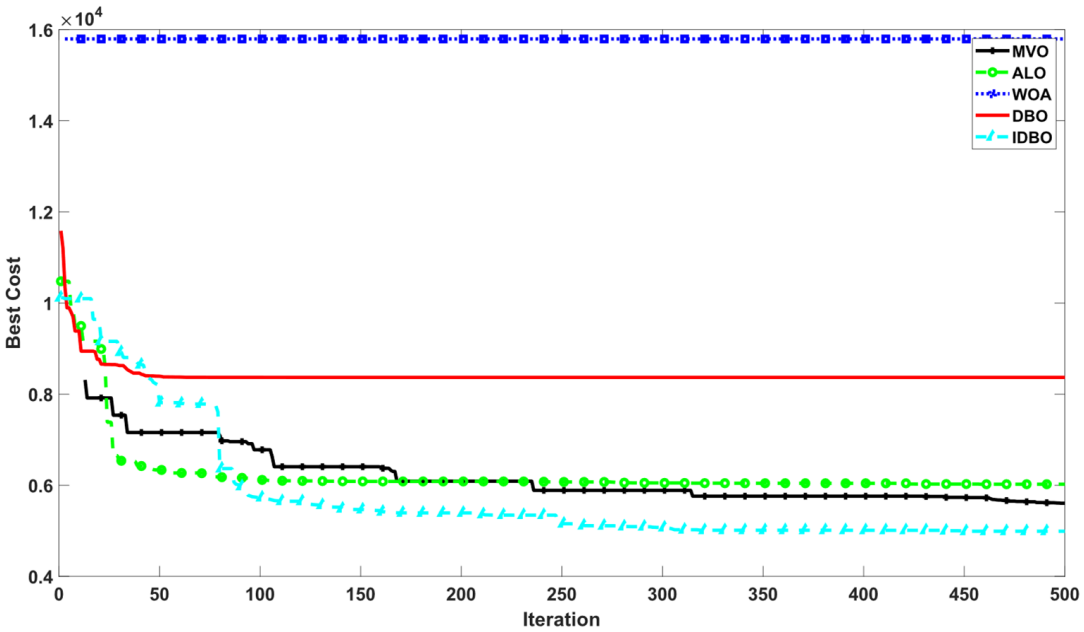

Figure 17. Experiment 2—algorithm iteration chart.

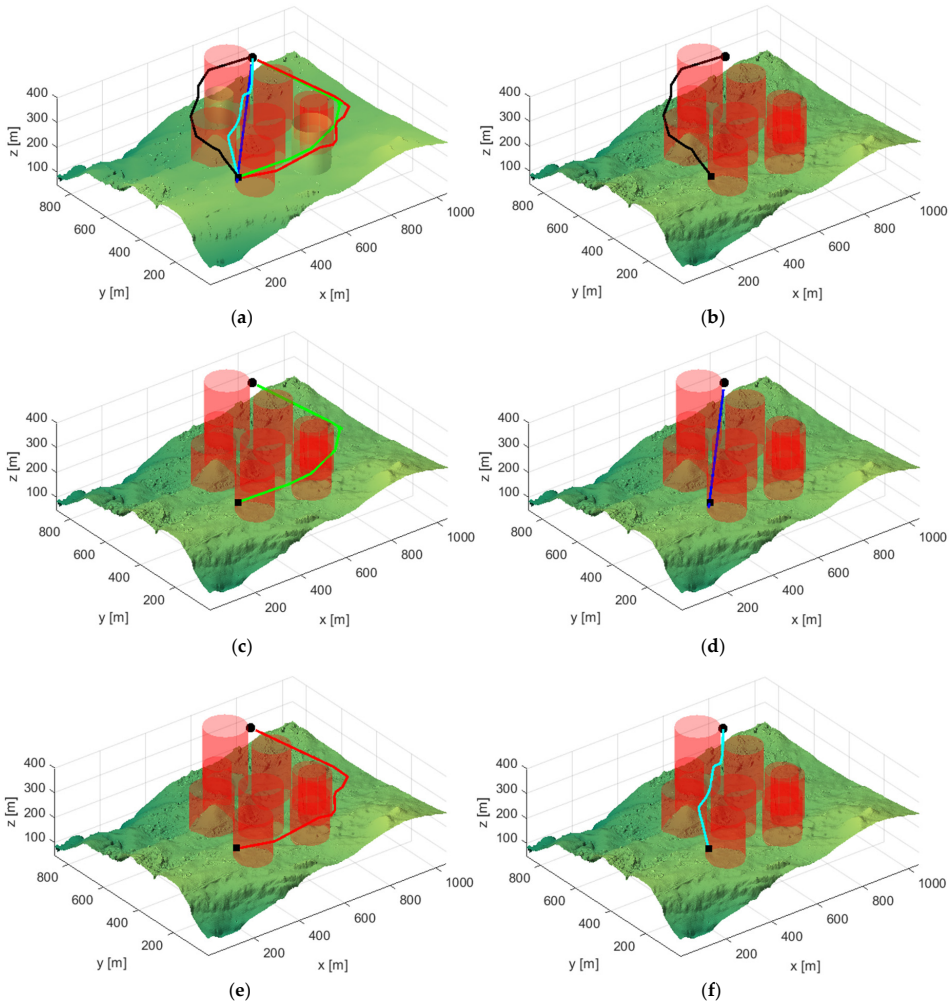

Figure 18. Experiment 2—UAV path 3D view. (**a**) 3D view—overall, (**b**) 3D view—MVO, (**c**) 3D view—ALO, (**d**) 3D view—WOA, (**e**) 3D view—DBO (**f**) 3D view—IDBO.

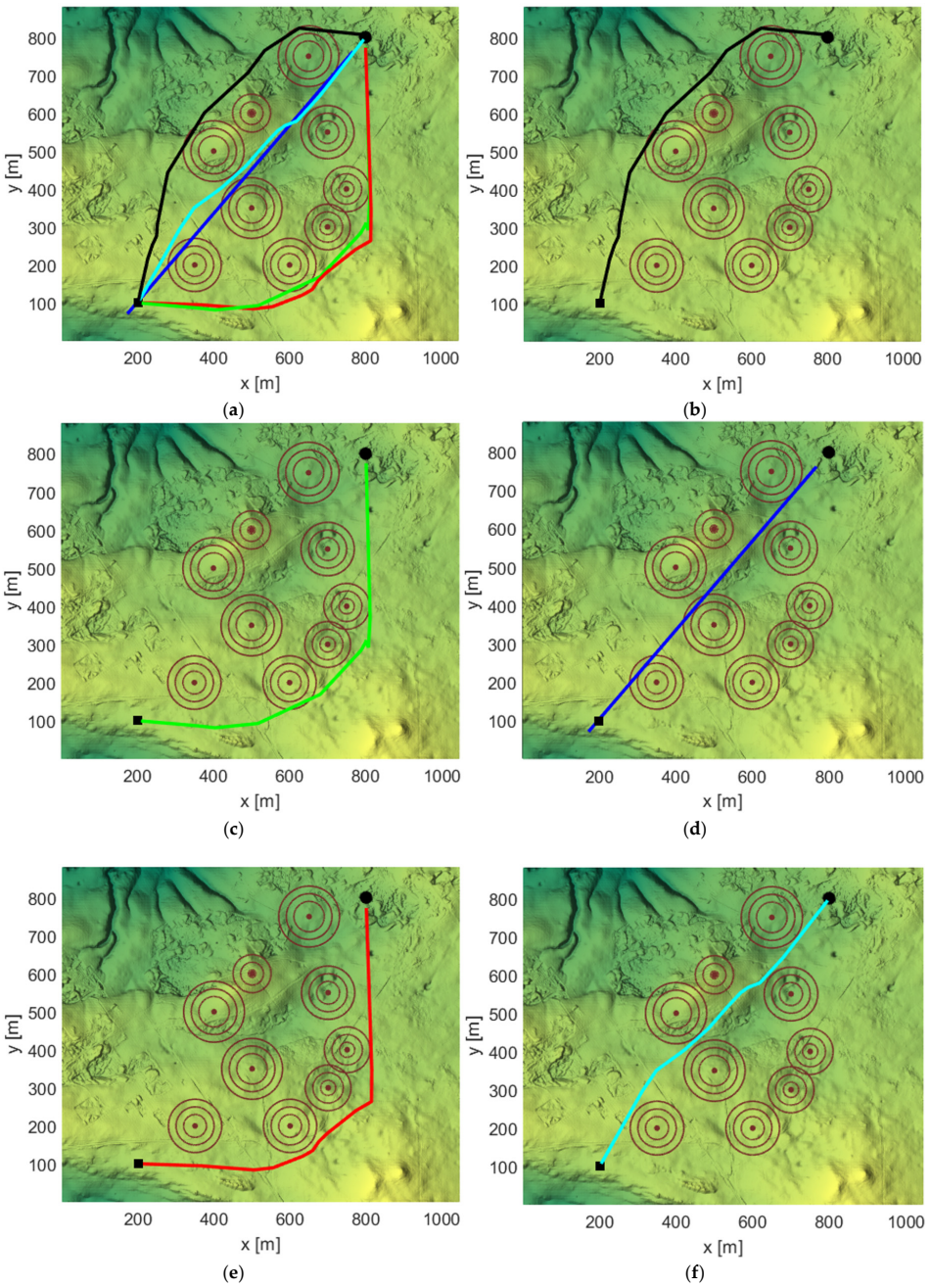

Figure 19. Experiment 2—UAV path top-down view. (**a**) Top-down view—overall, (**b**) top-down view—MVO, (**c**) top-down view—ALO, (**d**) top-down view—WOA, (**e**) top-down view—DBO, (**f**) top-down view—IDBO.

The path obtained by WOA directly collided with obstacles, making the solution infeasible, with the total cost being infinite (Inf). Among the other four algorithms, IDBO provided the solution with the lowest total cost. This demonstrates that in scenarios with a higher number of obstacles, IDBO outperformed the other four algorithms for UAV path planning.

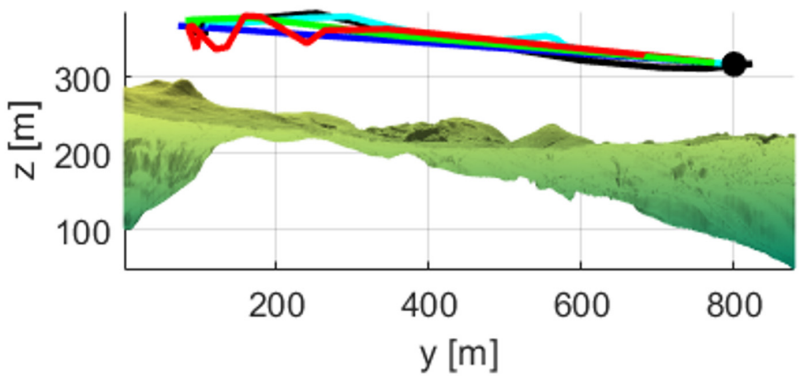

Figure 20. Experiment 2—UAV path side view.

5.3. Test Results of Scenario 3

In Scenario 3, with nine threats, a UAV path planning test was conducted, where the weights for flight distance cost, flight altitude cost, and threat/obstacle cost were set to 5, 10, and 100, respectively. The test results are shown in Figures 21–24, which include the iteration charts for each algorithm, the 3D view of the UAV path, the top-down view of the UAV path, and the side view of the UAV path. The correspondence between the different colored lines in each figure and the algorithms is consistent with the legend in Figure 21.

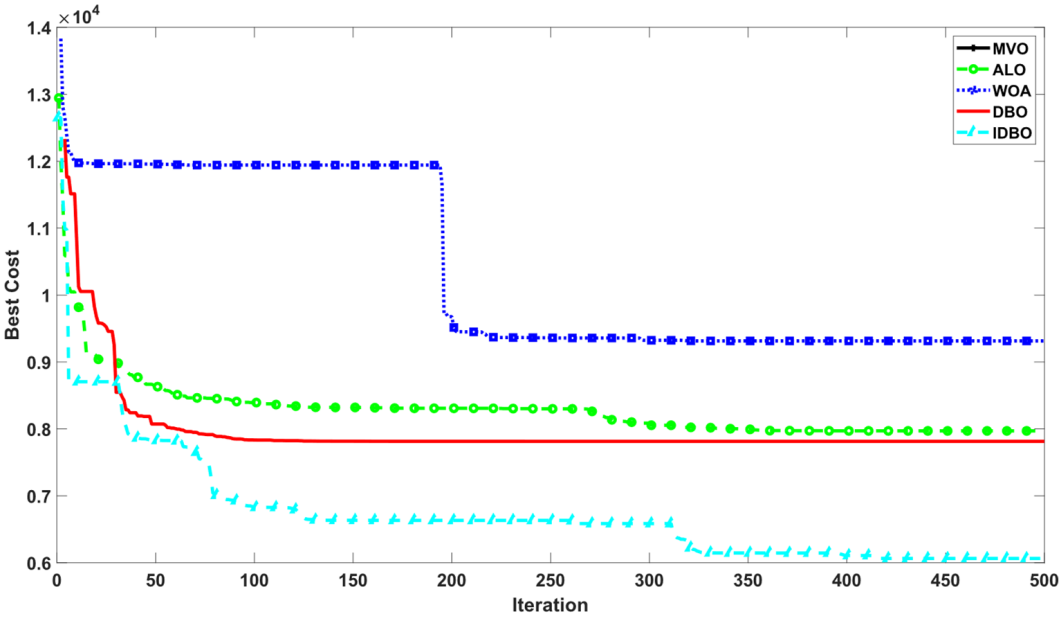

Figure 21. Experiment 3—algorithm iteration chart.

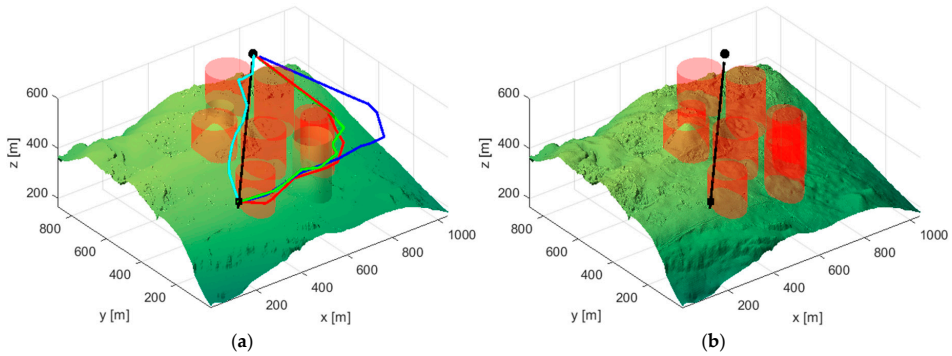

Figure 22. *Cont.*

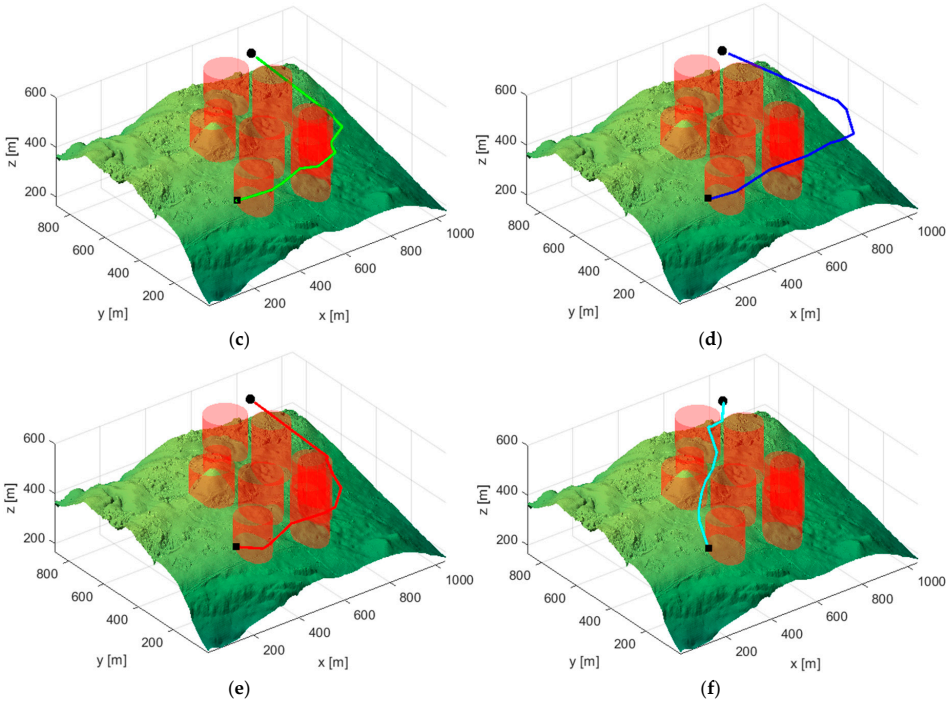

Figure 22. Experiment 3—UAV path 3D view. (**a**) 3D view—overall, (**b**) 3D view—MVO, (**c**) 3D view—ALO, (**d**) 3D view—WOA, (**e**) 3D view—DBO, (**f**) 3D view—IDBO.

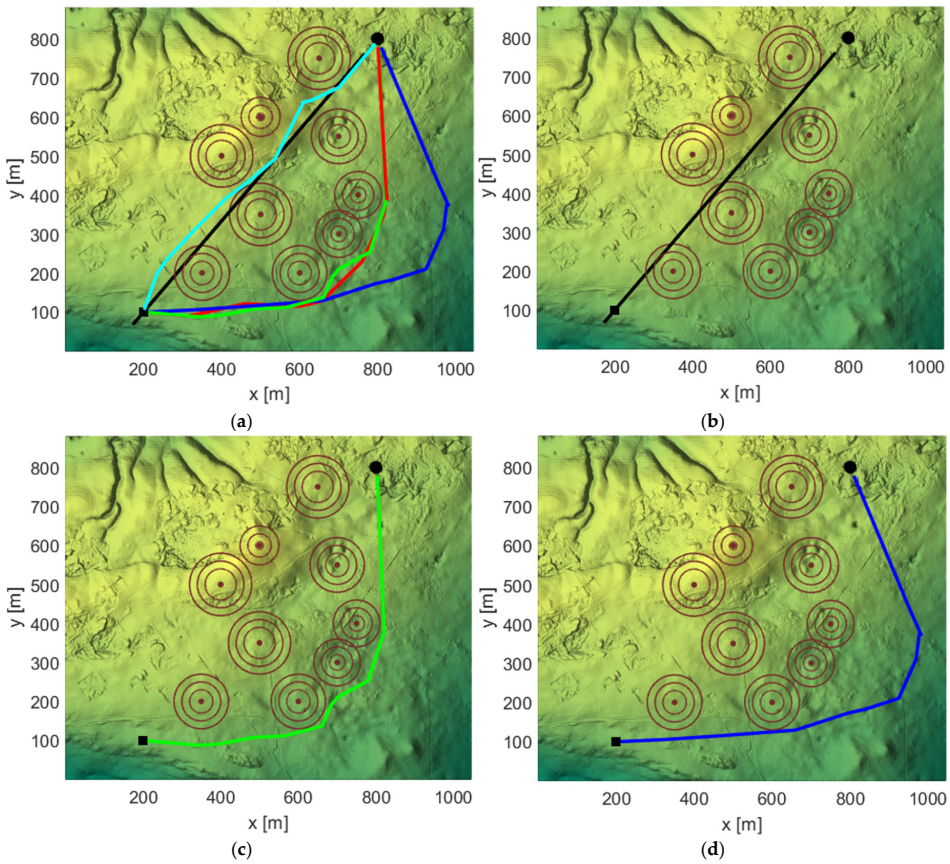

Figure 23. *Cont*.

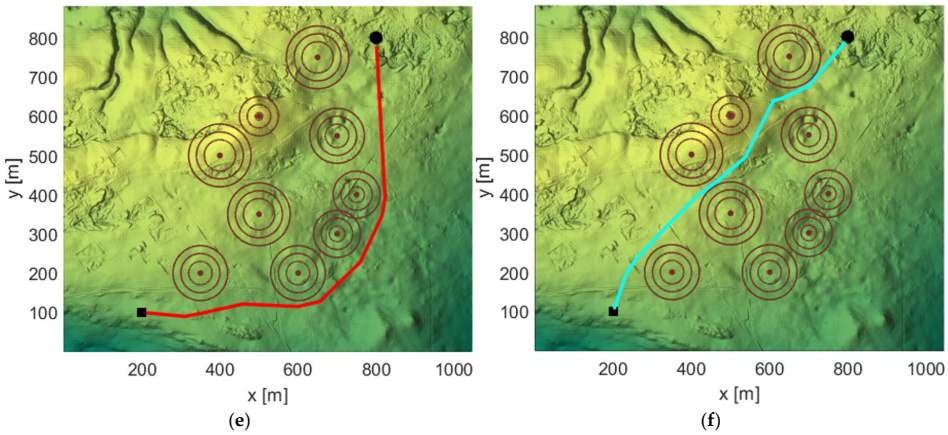

Figure 23. Experiment 3—UAV path top-down view. (**a**) Top-down view—overall, (**b**) top-down view—MVO, (**c**) top-down view—ALO, (**d**) top-down view—WOA, (**e**) top-down view—DBO, (**f**) top-down view—IDBO.

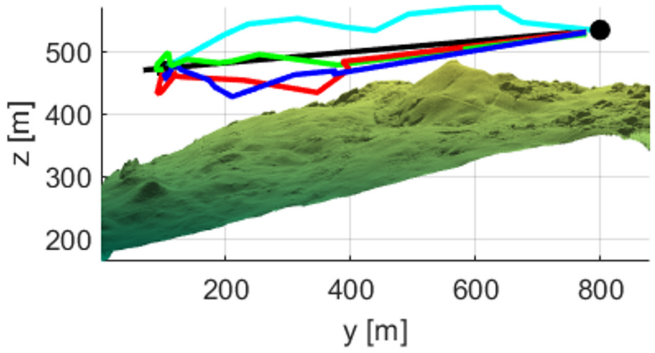

Figure 24. Experiment 3—UAV path side view.

The optimal values obtained by each algorithm are shown in Table 7.

Table 7. Experiment 3—optimal values of each algorithm.

Algorithm	MVO	ALO	WOA	DBO	IDBO
Flight Distance Cost	**924**	1283.6	1439.8	1262.5	993.4
Flight Altitude Cost	**0**	155.2348	211.5553	150.1933	88.2500
Threat/Obstacle Cost	Inf	**0**	**0**	**0**	2.148
Total Cost	Inf	7970.3	9314.7	7814.2	**6064.2**
Total Path Deviation Angle	**0**	296.4471	430.7547	231.3566	230.0188
Computational Time	2.3013 s	2.3766 s	1.2102 s	**1.1885 s**	2.0289 s

The solution obtained by MVO was infeasible, with the total cost being infinite (Inf). Among the other four algorithms, IDBO provided the solution with the lowest flight distance cost, flight altitude cost, and total cost. Additionally, its total path deviation angle was smaller than that of the other three solutions. This demonstrates that under different scenarios and varying weight configurations, IDBO outperformed the other four algorithms in terms of performance.

6. Summary and Outlook

This paper addresses the complexity of real-world airspace environments by constructing a more realistic unmanned aerial vehicle (UAV) trajectory planning model. Simulation experiments conducted under various environmental conditions and parameter settings

demonstrated the feasibility of applying the improved dung beetle optimization algorithm in UAV trajectory planning.

To solve this model, the dung beetle optimization algorithm was further enhanced by improving the population initialization method, introducing the sine–cosine movement strategy, and implementing population evolution strategies. These enhancements significantly enhanced the algorithm's search efficiency and the likelihood of finding global optimal solutions. Testing using the 23 benchmark functions provided by the International Conference on Evolutionary Computation (CEC) indicated that the convergence performance of the improved dung beetle optimization algorithm was significantly stronger than that of similar optimization algorithms, making it better suited for solving various problems.

For future research, the authors could continue exploring the potential of more advanced optimization algorithms in UAV trajectory planning and compare them with IDBO, continuously improving the algorithms. Additionally, research could focus on dynamic three-dimensional path planning for UAVs in unknown environments. By enhancing existing online path planning algorithms and integrating environmental information with UAV characteristics, efficient path planning and task allocation for UAVs in dynamic environments could be achieved. Furthermore, exploring techniques for dynamic UAV path planning could address the need for UAVs to rapidly evade enemy threats during flight, providing comprehensive technical support for UAV applications in real-world missions.

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

Funding: This work was supported by the National Natural Science Foundation of China under Grant No. 62272239, No. 62302237, and No. 62303214 and by Department of Science and Technology of Zhejiang Province Soft Science Research Program Project under Grant No. 2025C25009. The funders are Zhixin Sun for the first three grants and Fengjun Lv for the last one.

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

Conflicts of Interest: Author Kai Yuan was employed by the company Anhui Yugu Express intelligent Technology Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Khosravi, M.; Arora, R.; Enayati, S.; Pishro-Nik, H. A Search and Detection Autonomous Drone System: From Design to Implementation. *IEEE Trans. Autom. Sci. Eng.* **2025**, *22*, 3485–3501. [CrossRef]
2. Hong, D.; Lee, S.; Cho, Y.H.; Baek, D.; Kim, J.; Chang, N. Energy-Efficient Online Path Planning of Multiple Drones Using Reinforcement Learning. *IEEE Trans. Veh. Technol.* **2021**, *70*, 9725–9740. [CrossRef]
3. Zhang, M.; Han, Y.; Chen, S.; Liu, M.; He, Z.; Pan, N. A Multi-Strategy Improved Differential Evolution Algorithm for UAV 3D Trajectory Planning in Complex Mountainous Environments. *Eng. Appl. Artif. Intell.* **2023**, *125*, 106672. [CrossRef]
4. Liu, X.; Shao, P.; Li, G.; Ye, L.; Yang, H. Complex Hilly Terrain Agricultural UAV Trajectory Planning Driven by Grey Wolf Optimizer with Interference Model. *Appl. Soft Comput.* **2024**, *160*, 111710. [CrossRef]
5. Bai, H.; Fan, T.; Niu, Y.; Cui, Z. Multi-UAV Cooperative Trajectory Planning Based on Many-Objective Evolutionary Algorithm. *Complex Syst. Model. Simul.* **2022**, *2*, 130–141. [CrossRef]
6. Wang, M.; Zhang, D.; Li, C.; Zhang, Z. Multiple Fixed-Wing UAVs Collaborative Coverage 3D Path Planning Method for Complex Areas. *Def. Technol.* 2025; *in press*. ISSN 2214-9147. [CrossRef]
7. Wang, W.; Li, X.; Tian, J. UAV Formation Path Planning for Mountainous Forest Terrain Utilizing an Artificial Rabbit Optimizer Incorporating Reinforcement Learning and Thermal Conduction Search Strategies. *Adv. Eng. Inform.* **2024**, *62*, 102947. [CrossRef]

8. Pan, Z.; Zhang, C.; Xia, Y.; Xiong, H.; Shao, X. An Improved Artificial Potential Field Method for Path Planning and Formation Control of Multi-UAV Systems. *IEEE Trans. Circuits Syst. II Express Briefs* **2022**, *69*, 1129–1133. [CrossRef]
9. Zhou, X.; Ling, M.; Lin, Q.; Tang, S.; Wu, J.; Hu, H. Effectiveness Analysis of Multiple Initial States Simulated Annealing Algorithm: A Case Study on the Molecular Docking Tool AutoDock Vina. *IEEE ACM Trans. Comput. Biol. Bioinform.* **2023**, *20*, 3830–3841. [CrossRef] [PubMed]
10. Martins, O.O.; Adekunle, A.A.; Olaniyan, O.M.; Bolaji, B.O. An Improved Multi-Objective A-Star Algorithm for Path Planning in a Large Workspace: Design, Implementation, and Evaluation. *Sci. Afr.* **2022**, *15*, e01068. [CrossRef]
11. Fan, J.; Chen, X.; Liang, X. UAV Trajectory Planning Based on Bi-Directional APF-RRT* Algorithm with Goal-Biased. *Expert Syst. Appl.* **2023**, *213*, 119137. [CrossRef]
12. Shin, Y.; Kim, E. Hybrid Path Planning Using Positioning Risk and Artificial Potential Fields. *Aerosp. Sci. Technol.* **2021**, *112*, 106640. [CrossRef]
13. Orozco-Rosas, U.; Montiel, O.; Sepúlveda, R. Mobile Robot Path Planning Using Membrane Evolutionary Artificial Potential Field. *Appl. Soft Comput.* **2019**, *77*, 236–251. [CrossRef]
14. Bayili, S.; Polat, F. Limited-Damage A*: A Path Search Algorithm That Considers Damage as a Feasibility Criterion. *Knowl. Based Syst.* **2011**, *24*, 501–512. [CrossRef]
15. Wang, H.; Lai, H.; Du, H.; Gao, G. IBPF-RRT*: An Improved Path Planning Algorithm with Ultra-Low Number of Iterations and Stabilized Optimal Path Quality. *J. King Saud Univ. Comput. Inform. Sci.* **2024**, *36*, 102146. [CrossRef]
16. Wang, W.; Li, J.; Bai, Z.; Wei, Z.; Peng, J. Toward Optimization of AGV Path Planning: An RRT*-ACO Algorithm. *IEEE Access* **2024**, *12*, 18387–18399. [CrossRef]
17. Haris, M.; Bhatti, D.M.S.; Nam, H. A Fast-Convergent Hyperbolic Tangent PSO Algorithm for UAVs Path Planning. *IEEE Open J. Veh. Technol.* **2024**, *5*, 681–694. [CrossRef]
18. Jiacheng, L.; Lei, L. A Hybrid Genetic Algorithm Based on Information Entropy and Game Theory. *IEEE Access* **2020**, *8*, 36602–36611. [CrossRef]
19. Li, B.; Qi, X.; Yu, B.; Liu, L. Trajectory Planning for UAV Based on Improved ACO Algorithm. *IEEE Access* **2020**, *8*, 2995–3006. [CrossRef]
20. Liu, X.; Li, G.; Yang, H.; Zhang, N.; Wang, L.; Shao, P. Agricultural UAV Trajectory Planning by Incorporating Multi-Mechanism Improved Grey Wolf Optimization Algorithm. *Expert Syst. Appl.* **2023**, *233*, 120946. [CrossRef]
21. Hu, G.; Huang, F.; Shu, B.; Wei, G. MAHACO: Multi-Algorithm Hybrid Ant Colony Optimizer for 3D Path Planning of a Group of UAVs. *Inf. Sci.* **2025**, *694*, 121714. [CrossRef]
22. Gu, T.; Zhang, Y.; Wang, L.; Zhang, Y.; Deveci, M.; Wen, X. A Comprehensive Analysis of Multi-Strategic RIME Algorithm for UAV Path Planning in Varied Terrains. *J. Ind. Inf. Integr.* **2025**, *43*, 100742. [CrossRef]
23. Pang, S.Y.; Chai, Q.W.; Liu, N.; Zheng, W.M. A Multi-Objective Cat Swarm Optimization Algorithm Based on Two-Archive Mechanism for UAV 3-D Path Planning Problem. *Appl. Soft Comput.* **2024**, *167*, 112306. [CrossRef]
24. Chang, B.; Xi, W.; Lin, J.; Shao, Z. UAV Path Planning Based on Improved Dung Beetle Algorithm with Multiple Strategy Integration. *Proc. Inst. Mech. Eng. G J. Aerosp. Eng.* **2024**. [CrossRef]
25. Jue, X.; Shen, B. Dung Beetle Optimizer: A New Meta-Heuristic Algorithm for Global Optimization. *J. Supercomput.* **2022**, *79*, 7305–7336.
26. Mirjalili, S.; Mirjalili, M.S.; Hatamlou, A. Multi-Verse Optimizer: A Nature-Inspired Algorithm for Global Optimization. *Neural Comput. Appl.* **2016**, *27*, 495–513. [CrossRef]
27. Cao, B.; Zhao, S.; Li, X.; Wang, B. K-Means Multi-Verse Optimizer (KMVO) Algorithm to Construct DNA Storage Codes. *IEEE Access* **2020**, *8*, 29547–29556. [CrossRef]
28. Mirjalili, S. The Ant Lion Optimizer. *Adv. Eng. Softw.* **2015**, *83*, 80–98. [CrossRef]
29. Li, Y.; Yao, Y.; Hu, S.; Wen, Q.; Zhao, F. Coverage Enhancement Strategy for WSNs Based on Multiobjective Ant Lion Optimizer. *IEEE Sens. J.* **2023**, *23*, 13762–13773. [CrossRef]
30. Mirjalili, S. SCA: A Sine Cosine Algorithm for Solving Optimization Problems. *Knowl. Based Syst.* **2016**, *96*, 120–133. [CrossRef]
31. Wang, M.; Lu, G. A Modified Sine Cosine Algorithm for Solving Optimization Problems. *IEEE Access* **2021**, *9*, 27434–27450. [CrossRef]
32. Wan, Y.; Ma, A.; Zhang, L.; Zhong, Y. Multiobjective Sine Cosine Algorithm for Remote Sensing Image Spatial-Spectral Clustering. *IEEE Trans. Cybern.* **2022**, *52*, 11172–11186. [CrossRef] [PubMed]
33. Ghorpade, S.N.; Zennaro, M.; Chaudhari, B.S. GWO Model for Optimal Localization of IoT-Enabled Sensor Nodes in Smart Parking Systems. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 1217–1224. [CrossRef]
34. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
35. Malla, S.G.; Malla, P.; Malla, J.M.R.; Singla, R.; Choudekar, P.; Koilada, R.; Sahu, M.K. Whale Optimization Algorithm for PV Based Water Pumping System Driven by BLDC Motor Using Sliding Mode Controller. *IEEE J. Emerg. Sel. Top. Power Electron.* **2022**, *10*, 4832–4844. [CrossRef]

36. Chen, X. Research on New Adaptive Whale Algorithm. *IEEE Access* **2020**, *8*, 90165–90201. [CrossRef]
37. Rani, S.; Babbar, H.; Kaur, P.; Alshehri, M.D.; Shah, S.H. An Optimized Approach of Dynamic Target Nodes in Wireless Sensor Network Using Bio-Inspired Algorithms for Maritime Rescue. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 2548–2555. [CrossRef]
38. Li, X.; Qi, X.; Liu, X.; Gao, C.; Wang, Z.; Zhang, F.; Liu, J. A Discrete Moth-Flame Optimization with an l_2-Norm Constraint for Network Clustering. *IEEE Trans. Netw. Sci. Eng.* **2022**, *9*, 1776–1788. [CrossRef]
39. Alhussan, A.A.; Abdelhamid, A.A.; El-Kenawy, E.S.M.; Ibrahim, A.; Eid, M.M.; Khafaga, D.S.; Ahmed, A.E. A Binary Waterwheel Plant Optimization Algorithm for Feature Selection. *IEEE Access* **2023**, *11*, 94227–94251. [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.