symmetry

MDPI

Article

Multi-Strategy Enhanced Dung Beetle Optimizer and Its Application in Three-Dimensional UAV Path Planning

Qianwen Shen, Damin Zhang *, Mingshan Xie and Qing He

School of Big Data and Information Engineering, Guizhou University, Guiyang 550025, China; gs.qwshen22@gzu.edu.cn (Q.S.); msxie@gzu.edu.cn (M.X.); qhe@gzu.edu.cn (Q.H.)
* Correspondence: dmzhang@gzu.edu.cn; Tel.: +86-13-98-5413242

Abstract: Path planning is a challenging, computationally complex optimization task in high-dimensional scenarios. The metaheuristic algorithm provides an excellent solution to this problem. The dung beetle optimizer (DBO) is a recently developed metaheuristic algorithm inspired by the biological behavior of dung beetles. However, it still has the drawbacks of poor global search ability and being prone to falling into local optima. This paper presents a multi-strategy enhanced dung beetle optimizer (MDBO) for the three-dimensional path planning of an unmanned aerial vehicle (UAV). First, we used the Beta distribution to dynamically generate reflection solutions to explore more search space and allow particles to jump out of the local optima. Second, the Levy distribution was introduced to handle out-of-bounds particles. Third, two different cross operators were used to improve the updating stage of thief beetles. This strategy accelerates convergence and balances exploration and development capabilities. Furthermore, the MDBO was proven to be effective by comparing seven state-of-the-art algorithms on 12 benchmark functions, the Wilcoxon rank sum test, and the CEC 2021 test suite. In addition, the time complexity of the algorithm was also analyzed. Finally, the performance of the MDBO in path planning was verified in the three-dimensional path planning of UAVs in oil and gas plants. In the most challenging task scenario, the MDBO successfully searched for feasible paths with the mean and standard deviation of the objective function as low as 97.3 and 32.8, which were reduced by 39.7 and 14, respectively, compared to the original DBO. The results demonstrate that the proposed MDBO had improved optimization accuracy and stability and could better find a safe and optimal path in most scenarios than the other metaheuristics.

Keywords: metaheuristic; dung beetle optimizer (DBO); dynamic reflective learning strategy; Beta distribution; Levy distribution; crisscross optimizer; UAV path planning; constrained optimization

Citation: Shen, Q.; Zhang, D.; Xie, M.; He, Q. Multi-Strategy Enhanced Dung Beetle Optimizer and Its Application in Three-Dimensional UAV Path Planning. *Symmetry* **2023**, *15*, 1432. https://doi.org/10.3390/sym15071432

Academic Editors: Peng-Yeng Yin, Ray-I Chang and Jen-Chun Lee

Received: 2 June 2023
Revised: 4 July 2023
Accepted: 13 July 2023
Published: 17 July 2023

1. Introduction

Artificial intelligence is one of the most promising technologies for solving problems in various fields. UAV is a powerful auxiliary tool in artificial intelligence technology with high flexibility, low cost, and high efficiency [1]. It is also considered as a candidate technology for future 6G networks and has wide applications in various scenarios [2]. For example, in hazardous areas such as oil and gas plants, UAVs can replace humans to perform specific tasks and improve work efficiency. However, ensuring the safety of autonomous flight UAVs in complex environments is a significant challenge. Path search has essential research significance as a supporting technology for UAV flight. Path planning allows the UAV to find the optimal safe path from a start point to an endpoint according to specific optimization criteria such as the minimal path length or the lowest energy consumption [3]. The UAV path searching problem is based on the case of task points and prior maps. Therefore, the mission, the surrounding environment, and the UAV's physical constraints should be clarified first. On this basis, the task requirements, environmental states, and constraints related to UAVs are expressed through a mathematical model and applied to the path-planning problem. It has been proven that finding the optimal path is

an NP-hard problem [4], and the complexity of the problem increases rapidly with the size of the problem.

Many researchers have been trying to develop solutions to path-planning problems, for example, conventional methods such as the artificial potential algorithm [5] and the Voronoi diagram search method [6]; cell-based methods such as Dijkstra [7] and the A-Star algorithm [8]; model-based methods such as rapidly exploring random tree (RRT) [9]; learning-based methods such as neural networks [10] and the evolutionary computation technique [11]. However, the limitations of the traditional and cell-based methods lie in poor flexibility and fault tolerance. The model-based methods typically have complex modeling and cannot work in real-time path planning. Neural networks have low complexity, flexibility, and fault tolerance advantages. Nevertheless, most neural networks require a learning process, which is time-consuming. Due to the limitations of traditional methods to successfully solve optimal paths, researchers have proposed metaheuristic algorithms for solving path-planning problems inspired by natural phenomena or laws, for example, genetic algorithm (GA) [12], particle swarm optimization (PSO) [13], ant colony optimization (ACO) [14], differential evolution (DE) [15], fruit fly optimization algorithm (FOA) [16], and the grey wolf optimization (GWO) algorithm [17]. The dung beetle optimization algorithm [18] is a recently developed metaheuristic algorithm that imitates the biological behavior of dung beetles. However, a determined metaheuristic cannot perform well in all categories of optimization problems, a hypothesis already proven by the "No Free Lunch" theorem [19]. This paper aimed to develop an efficient optimization algorithm for solving three-dimensional path-planning problems in oil and gas plants.

Nowadays, researchers have proposed many improved metaheuristic algorithms for solving various engineering application problems including path-planning problems. Yu X. et al. [15] modeled UAV path planning as a constraint satisfaction problem, in which the fitness function included the travel distance and risk of the UAV. Three constraints considered in the problem were UAV height, angle, and limited UAV slope. Jain et al. [20] proposed the multiverse optimizer (MVO) algorithm to deal with the coordination of UAVs in a three-dimensional environment. The results showed that the MVO algorithm performed better in most testing scenarios with the minimum average execution time. Li K. et al. [21] focused on multi-UAV collaborative task assignment problems with changing tasks. They proposed an improved fruit fly optimization algorithm (ORPFOA) to solve the real-time path-planning problem. Pehlivanoglu et al. [12] considered the path-planning problem of autonomous UAVs in target coverage problems and proposed initial population enhancement methods in GA for efficient path planning. These techniques aim to reduce the chances of collisions between UAVs.

Zhang X. et al. [22] proposed an improved FOA (MCFOA) by introducing two new strategies, Gaussian mutation and chaotic local search. Then, the MCFOA was applied to the feature selection problem to verify its optimization performance. Song S. et al. [23] used two different Gaussian variant strategies and dimension decision strategies to design an enhanced HHO (GCHHO) algorithm. The GCHHO algorithm has successfully optimized engineering design problems (such as tensile/compression springs and welded beam design problems). Gupta S. et al. [24] presented an improved random walk grey wolf algorithm and applied it to the parameter estimation of the frequency modulation (FM) unconstrained optimization problem, the optimal capacity of production facilities, and the design of pressure vessels. Pichai [25] introduced asymmetric chaotic sequences into the competitive swarm optimization (CSO) algorithm and applied it to the feature selection of high-dimensional data. The results showed that the algorithm reduced the number of features and improved the accuracy. Mikhalev [26] proposed a cyclic spider algorithm to optimize the weights of recurrent neural networks, and the results showed that the algorithm reduced the prediction errors. Almotairi [27] proposed a hybrid algorithm based on the reptile search algorithm and the remora algorithm, improving the original algorithm's search capability. Then, the hybrid algorithm was applied to the cluster analysis problem.

Previous work has deeply studied the defects of intelligent algorithms and improved the convergence ability of algorithms. In this paper, we improved the optimization ability of the DBO and the solving efficiency of the UAV path-planning problem. The following work was conducted to improve the search performance of the DBO:

- Adding a reflective learning strategy and using Beta distribution function mapping to generate reflective solutions, improving the algorithm's search ability;
- For particles that exceeded the search space range, Levy distribution mapping was used to handle particle boundaries, enhancing the probability of global search reaching the optimal position;
- Individual crossover mechanism and dimension crossover mechanism were used to update the position of individual thief beetles, increasing the population diversity and avoiding falling into local optima;
- Applying the improved MDBO to solve the three-dimensional UAV path-planning problem, and design sets of scene experiments to verify the efficiency of the MDBO.

The rest of this paper is organized as follows. Section 2 illustrates the basic DBO. In Section 3, the proposed MDBO is described in detail. In Section 4, the experimental results of the proposed MDBO are analyzed. Section 5 describes the UAV path-planning model. Section 6 conducts the UAV planning simulation experiments and discusses the comparison results between the MDBO and other metaheuristic algorithms. Finally, Section 7 provides a summary of the paper.

2. Dung Beetle Optimizer (DBO)

The basic DBO is based on population, mainly inspired by the ball-rolling, dancing, foraging, stealing, and reproduction behaviors of dung beetles. The DBO divides the population into four search agents: ball-rolling dung beetle, brood ball, small dung beetle, and thief. Specifically, each search agent has its own unique updating rules. Note that specific details will be described as follows.

(1) Ball-rolling dung beetle

During the rolling process, dung beetles must navigate through celestial cues to keep the dung ball rolling in a straight line. Thus, the position of the ball-rolling dung beetle is updated and can be expressed as:

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

$$\Delta x = |x_i(t) - X^w| \tag{2}$$

where t represents the current number of iterations; $x_i(t)$ represents the position information of the ith dung beetle during the tth iteration; $k \in (0, 0.2]$ is a constant value representing the deflection coefficient; b is a constant value belonging to (0,1); α is a natural coefficient assigned to 1 or -1; X^w is the global worst position; Δx simulates changes in light intensity.

When a dung beetle encounters obstacles that prevent it from progressing, it will redirect itself through dancing to obtain a new route. A tangent function simulates the dancing behavior to obtain a new rolling direction. Therefore, the location of the ball-rolling dung beetle is updated as follows:

$$x_i(t+1) = x_i(t) + \tan(\theta)|x_i(t) - x_i(t-1)| \tag{3}$$

where $\theta \in [0, \pi]$ is the deflection angle.

(2) Brood ball

Choosing a suitable spawning site is crucial for dung beetles to provide a safe environment for their offspring. Hence, in DBO, a boundary selection strategy is proposed to simulate the spawning area of female dung beetles, which is defined as follows:

$$\begin{aligned} Lb^* &= \max(X^* \times (1-R),\ Lb) \\ Ub^* &= \min(X^* \times (1-R),\ Ub) \end{aligned} \tag{4}$$

where X^* represents the current local optimal position; Lb^* and Ub^* represent the lower and upper bounds of the spawning area; $R = 1 - t/T_{\max}$; $T_{\max}$ denotes the maximum number of iterations; Lb and Ub are the lower and upper bounds of the search space, respectively.

It is assumed that each female dung beetle will only lay one egg in each iteration. The boundary range of the spawning area changes dynamically with values, so the position of the egg ball also changes dynamically during the iteration process expressed as follows:

$$B_i(t+1) = X^* + b_1 \times (B_i(t) - Lb^*) + b_2 \times (B_i(t) - Ub^*) \tag{5}$$

where $B_i(t)$ is the position of the ith sphere at the tth iteration; b_1 and b_2 are two independent random vectors with a size of $1 \times D$; D is the dimension.

(3) Small dung beetle

Some adult dung beetles will burrow out of the ground in search of food, and this type of dung beetle is called the small dung beetle. The boundaries of the optimal foraging area for small dung beetles are defined as follows:

$$\begin{aligned} Lb^b &= \max(X^b \times (1-R),\ Lb) \\ Ub^b &= \min(X^b \times (1-R),\ Ub) \end{aligned} \tag{6}$$

where X^b represents the global optimal position; Lb^b and Ub^b are the lower and upper bounds of the optimal foraging area, respectively. Therefore, the location of the small dung beetle is updated as follows:

$$x_i(t+1) = x_i(t) + C_1 \times (x_i(t) - Lb^b) + C_2 \times (x_i(t) - Ub^b) \tag{7}$$

where $x_i(t)$ represents the position of the ith dung beetle at the tth iteration; C_1 is the random number subject to normal distribution; C_2 is the random vector within the range of (0,1).

(4) Thief

Some dung beetles, known as thieves, steal dung balls from other beetles. From Equation (5), it can be observed that X^b is the optimal food source. Assume that the area around X^b is the optimal location for competing food. During the iteration process, the location information of the thief is updated as follows:

$$x_i(t+1) = X^b + S \times g \times (|x_i(t) - X^*| + \left|x_i(t) - X^b\right|) \tag{8}$$

where $x_i(t)$ represents the location information of the ith thief at the tth iteration; g is a random vector subject to normal distribution with the size of $1 \times D$; S is a constant value.

3. The Proposed Method

The MDBO proposed in this section mainly includes three strategies: generating reflective solutions, handling position boundary violations, and improving thief dung beetle position updates.

3.1. Dynamic Reflective Learning Strategy Based on Beta Distribution

The dynamic region selection strategy proposed by DBO strictly limits the search scope to this region, which is not conducive to searching other regions. The search scope can be expanded by dynamically calculating the solution in the opposite direction, helping the original algorithm break away from the local optimal region. The traditional method of randomly generating initial positions according to uniform distribution is simple and feasible. It is adverse to forming an "encircling" to the optimal solution. This paper proposes generating reflective solutions using Beta distribution to effectively surround the optimal solution with the initial position.

The Beta distribution function is defined as:

$$f(x) = \frac{1}{B(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1},\ x \in [0,1] \tag{9}$$

where $B(\alpha,\beta)$ is the Beta function, defined as:

$$B(\alpha,\beta) = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1}dt \tag{10}$$

Figure 1 shows the probability density function of the Beta distribution. From the figure, it can be seen that when $\alpha = \beta = 0.5$, the shape of the density function is symmetric in a U-shape. The candidate solution generated is most likely located near the boundary of the search space, so the global optimal solution is better "surrounded" within the initial particle swarm.

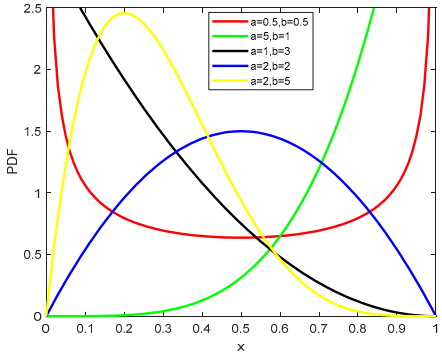


Figure 1. Probability density function image of Beta distribution.

Assume that the number of particles is N, the dimension is D, the maximum number of iterations is T, the upper and lower bounds of the jth dimension are $[Lb^j, Ub^j]$, and the reverse solution is u, generated according to the following formulas:

$$u_{i,j}(t+1) = \begin{cases} r_{i,j}(t+1)\ if(x_{i,j}(t+1) \geq s_j) \\ p_{i,j}(t+1)\ if(x_{i,j}(t+1) < s_j) \end{cases},\ i \in [1,N],\ j \in [1,D],\ t \in [1,T] \tag{11}$$

$$r_{i,j}(t+1) = \begin{cases} \Phi(Ub^j + Lb^j - x_{i,j}(t+1),\ s_j)\ \ if(\frac{|x_{i,j}(t+1)-s_j|}{|Ub^j-Lb^j|} < \Phi(0,1)) \\ \Phi(Lb^j,\ Ub^j + Lb^j - x_{i,j}(t+1))\ otherwise \end{cases} \tag{12}$$

$$p_{i,j}(t+1) = \begin{cases} \Phi(s_j,\ Ub^j + Lb^j - x_{i,j}(t+1))\ \ if(\frac{|x_{i,j}(t+1)-s_j|}{|Ub^j-Lb^j|} < \Phi(0,1)) \\ \Phi(Ub^j + Lb^j - x_{i,j}(t+1),\ Ub^j)\ otherwise \end{cases} \tag{13}$$

$$s_j = \frac{1}{2}(Ub^j + Lb^j) \tag{14}$$

$$\Phi(a,b) = a + (b-a) \times \text{betarnd}(\alpha,\beta) \tag{15}$$

where $x_{i,j}(t+1)$ is the jth value of the ith solution at the t + 1th iteration; $u_{i,j}(t+1)$ is the jth value of the ith reflective solution at the t + 1th iteration; s_j is the domain center of the upper and lower boundaries; $\Phi(a,b)$ generated a random number within (a,b) that follows the Beta distribution; betarnd is a function to randomly generate a number that obeys the Beta distribution. In this article, $\alpha = \beta = 0.5$.

3.2. Cross Boundary Limits Method Based on Levy Distribution

The simple method of "if the particle exceeds the boundary, it is equal to the boundary" is used to deal with the individual position. The advantage of this method is that it is simple to implement. However, because the boundary point is not the global optimal solution, this method is unfavorable to the optimization process. In this paper, a new boundary processing method based on Levy distribution mapping was adopted. The specific operations are as follows:

$$x'_{ij} = \begin{cases} \min(Ub^j \otimes Levy(\lambda), Ub^j), & x_{ij} > Ub^j \\ \max(Lb^j, Lb^j \otimes Levy(\lambda)), & x_{ij} < Lb^j \end{cases} \tag{16}$$

where x'_{ij} is the particle position after boundary processing; λ is a constant number; *Levy* is a random search path, and its random step size is a Levy distribution. The notation $\otimes$ is entry-wise multiplications. Figure 2 visually shows the method for handling particle boundaries. When the particle x_{ij} jumps out of the upper and lower boundaries, it re-enters the search region using Levy steps with random jumps.

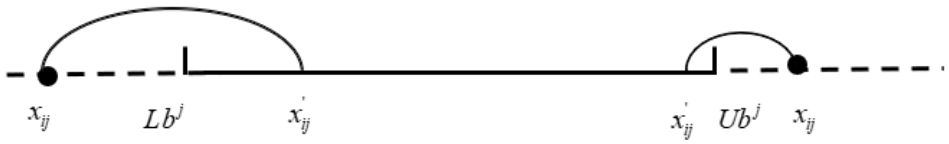


Figure 2. Cross boundary method based on Levy distribution.

3.3. Cross Operators for Updating the Location of Thieves

As can be seen from Equation (8), the thief's position is affected by the global optimal solution. This mechanism enables the optimal solution to be generated by the thieves. However, once the optimal individual falls into the local optimal solution, the solving efficiency of the algorithm will be greatly reduced. Inspired by the crisscross optimizer [28], this paper introduced horizontal and vertical crossover operators to perform crosstalk operations on thieves to improve the convergence ability of the algorithm. Note that specific details are described as follows.

(1) Horizontal crossover search (HCS)

HCS is similar to the crossover operation in genetic algorithms, which involves performing crossover operated on all dimensions between two individuals. Assuming that the dth dimension of the parent individual x_i and x_j are used to perform HCS, their kids can be reproduced as:

$$MS_{i,d} = \varepsilon_1 \times x_{i,d} + (1-\varepsilon_1) \times x_{j,d} + c_1 \times (x_{i,d} - x_{j,d}) \tag{17}$$

$$MS_{j,d} = \varepsilon_2 \times x_{j,d} + (1-\varepsilon_2) \times x_{i,d} + c_2 \times (x_{j,d} - x_{i,d}) \tag{18}$$

where ε_1 and ε_2 are random numbers uniformly distributed in the range of (0,1); c_1 and c_2 are random numbers uniformly distributed in the range of $(-1,1)$. $MS_{i,d}$ and $MS_{j,d}$ are the kids generated by $x_{i,d}$ and $x_{j,d}$, respectively.

(2) Vertical crossover search (VCS)

VCS is a crossover operation performed between two dimensions of the corresponding individuals, which has a lower probability of occurrence than HCS. Assume that the d1th

and d2th dimensions of the individual i are used to perform VCS, which is calculated as follows:

$$MS_{i,d_1} = \varepsilon \times x_{i,d_1} + (1 - \varepsilon) \times x_{i,d_2} \tag{19}$$

where ε is random number uniformly distributed in the range of (0,1), and MS_{i,d_1} is the offspring of x_{i,d_1} and x_{i,d_2}.

3.4. The Detailed Process of the MDBO

By introducing the above three strategies into the MDBO, the convergence speed and convergence accuracy of the algorithm can be effectively improved to balance the global exploration and local exploitation while enhancing the performance of the original DBO.

Figure 3 shows the process of MDBO implementation.

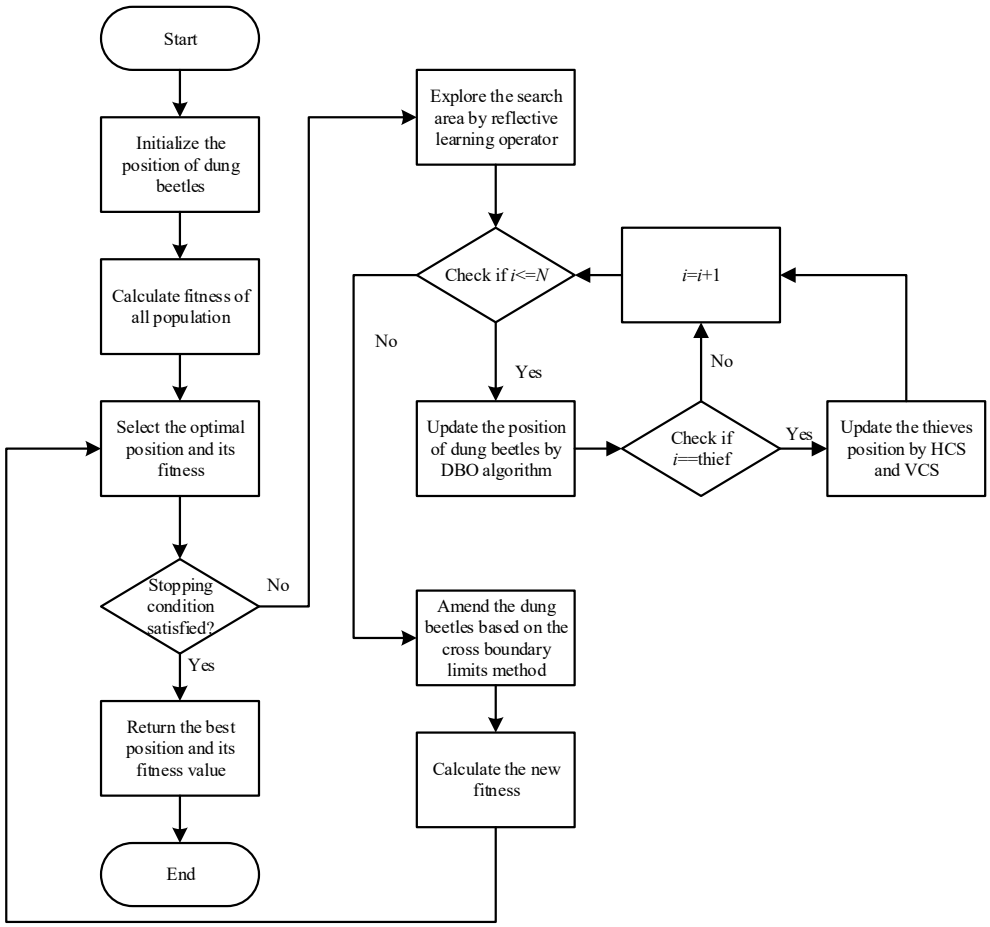


Figure 3. Flowchart of the MDBO.

The pseudo-code of MDBO is shown in Algorithm 1.

3.5. Computational Complexity Analysis

The pseudo-code of the proposed algorithm is shown in Algorithm 1. The complexity of the optimizer is one of the key indicators that can reflect the algorithm's efficiency. According to the pseudo-code in Algorithm 1, it can be seen that the complexity of MDBO mainly includes these several processes: initialization; calculating the reflective solutions by Beta distribution; HCS and VCS; updating the dung beetles' position with new boundary handling method. For convenience, it was assumed that N solutions and T iterations are involved in the optimization process of a D-variable problem. First, the complexity in the initialization phase is $O(N)$. Then, the complexity level of calculating N reflective solutions is $O(T * N)$. The complexity level of HCS and VCS is $O(T * N * D + T * N * N)$.

In MDBO, the total number of evaluations in updating the particles' position is $4 * T * N$ because N group solutions are evaluated per cycle. Thus, the time complexity of the dung beetles to look for the best position is about $O(4 * T * N)$, which can be abbreviated as $O(T * N)$. Based on the above discussions, the overall time complexity of MDBO is $O(MDBO) = O(N + T * N * (1 + D + N))$. Compared to DBO, although it increases the time overhead, the performance is optimized.

Algorithm 1: The pseudo code of MDBO

```
Initialize the particle's population N; the maximum iterations T; the dimensions D.
Initialize the positions of the dung beetles
While t ≤ T do
  Calculate the current best position and its fitness
  Obtain N reflective solutions by Equations (11)–(15)
  Update the positions of N individuals
  For i = 1:N do
    if i == ball-rolling dung beetle then
     Generate a random number p ∈ (0, 1)
    if p < 0.9 then
     Update search position by Equation (1)
    Else
     Update search position by Equation (3)
    end if
    if i == brood ball then
     Update search position by Equation (5)
    end if
    if i == small dung beetle then
       Update search position by Equation (7)
    end if
    if i == thief then
     Update search position by Equation (8)
     while t ≤ T/4 do
      Perform HCS using Equations (17)–(18)
      Perform VCS using Equation (19)
     end while
    end if
  end for
  Update the best position and its fitness
  t = t + 1
end while
Return the optimal solution X^b and its fitness f_b.
```

4. Analysis of Simulation Experiments

4.1. Experimental Design

In this section, a series of numerical experiments on MDBO were conducted to validate the effectiveness of MDBO. The statistical analysis and convergence analysis were performed on 12 basic benchmark functions [29–31] and CEC2021 competition functions [32]. Four well-established optimization techniques (the POA [33], SCSO [34], HBA [35], and DBO [18]), and three improved metaheuristics in recent years (the ISSA [36], MCFOA [22], GCHHO [23]) were compared with our proposed MDBO. Table 1 shows the parameter settings of all algorithms. For fairness, the population size N = 30 for all algorithms, and the maximum iterations T = 500. Mean (Mean) and standard deviation (Std) are used as statistical indicators to assess the optimization performance. All experiments were achieved on MatlabR2019a version.

Table 1. Parameter settings of all algorithms.

Algorithm	Parameters
MDBO	$k = 0.1$, $b = 0.3$, $\alpha = \beta = 0.5$
POA	$I = 1$ or 2, $R = 0.2$
SCSO	$S = 2$
HBA	$\beta = 6$, $C = 2$
DBO	$k = 0.1$, $b = 0.3$
ISSA	$w = 0.7$
MCFOA	$M = 4$, $c_1 \in (0,1)$
GCHHO	$E_0 \in [-1,1]$, $\theta = 0$

4.2. Sensitivity Analysis of MDBO's Parameters

This section analyzes the sensitivity of the control parameters (k, b, α, β) employed in the MDBO. The parameters k and b are consistent with the values taken in the DBO. Xue [18] has demonstrated that the algorithm performance is the most robust when $k = 0.1$, $b = 0.3$. In this subsection, we focused on analyzing the effect of the added parameters α and β on the search performance of the algorithm. Four CEC test functions (including the unimodal function CEC-1, the basic function CEC-3, the hybrid function CEC-7, and the composition function CEC-10) were used to test the design of these control parameters. Five parameter combinations, $\{\alpha = 0.5,\ \beta = 0.5\}$, $\{\alpha = 5,\ \beta = 1\}$, $\{\alpha = 1,\ \beta = 3\}$, $\{\alpha = 2,\ \beta = 2\}$, and $\{\alpha = 2,\ \beta = 5\}$, were set according to different distributions.

The sensitivity analysis is shown in Figure 4, where the horizontal coordinates indicate the five parameter combinations, and the vertical coordinates indicate the mean values. It can be seen from Figure 4a,c that the algorithm performed optimally when $\alpha = 0.5$ and $\beta = 0.5$. For CEC1, α and β showed a high sensitivity to different inputs. Noting that $a = 0.5$ and b = 0.5 had the best performance. Moreover, we could see that $\alpha = 0.5$ and $\beta = 0.5$ could obtain the best search performance in CEC7. It is observable in Figure 4b,d that $\alpha = 0.5$ and $\beta = 0.5$ demonstrated the robust behavior of different inputs. For CEC3, from the values displayed in the vertical coordinates, the optimized objective function values obtained for various combinations of a and b were relatively close. Note that $\alpha = 0.5$ and $\beta = 0.5$ obtained second place after $\alpha = 2$ and $\beta = 2$. In CEC10, the performance of $\alpha = 2$ and $\beta = 5$ was better than that of $\alpha = 2$ and $\beta = 2$. However, $\alpha = 0.5$ and $\beta = 0.5$ still achieved second place, showing their robust performance. Therefore, $\alpha = 0.5$ and $\beta = 0.5$ were selected as the recommended parameter values for MDBO.

4.3. Comparison of Performance on 12 Benchmark Functions

The 12 basic benchmark functions included seven unimodal functions and six multimodal functions (details in Appendix A, Table A1). It is worth noting that F1–F7 are typical unimodal test functions since they have one and only one global optimal value. The multimodal functions F8–F12 have features with multiple local optimums. Therefore, F1–F7 are widely used to estimate the convergence accuracy and speed, and F8–F12 are more suitable for testing the global exploration ability. Moreover, to fairly compare the comprehensive search ability of each metaheuristic, all algorithms were independently run 30 times in each experiment. We also tested the scalability and compared the performance of MDBO on dimensions of 30, 50, and 100. Two indicators were used to measure their optimization performance: the mean value (Mean) and the standard deviation (Std).

Table 2 shows the mean and standard deviation data of all algorithms on the 12 benchmark test functions tested, and the table also provides the comparison results of the algorithms when the test functions were taken in 30, 50, and 100 dimensions, respectively. As can be seen from Table 2, MDBO achieved the best results in all of the test functions. From the perspective of vertical comparison, MDBO ranked first in the test of other functions, except for some algorithms that showed a comparable performance to MDBO in the F9, F10, and F11 functions. For functions F1–F6, MDBO converged to the theoretical optimal value of 0. For F7 and F8, although MDBO could not have the best value, it showed significant

advantages compared with other algorithms. Next, from the results of the dimension comparison, the results of 100 dimensions, 30 dimensions, and 50 dimensions were similar, with little change in the order of optimal value. Based on the above analysis, it can be seen that MDBO has more significant competitive advantages in solving unimodal and multimodal functions. The results reflect the ability to jump out of the local optima by the reflective learning operator and crisscross optimizer.

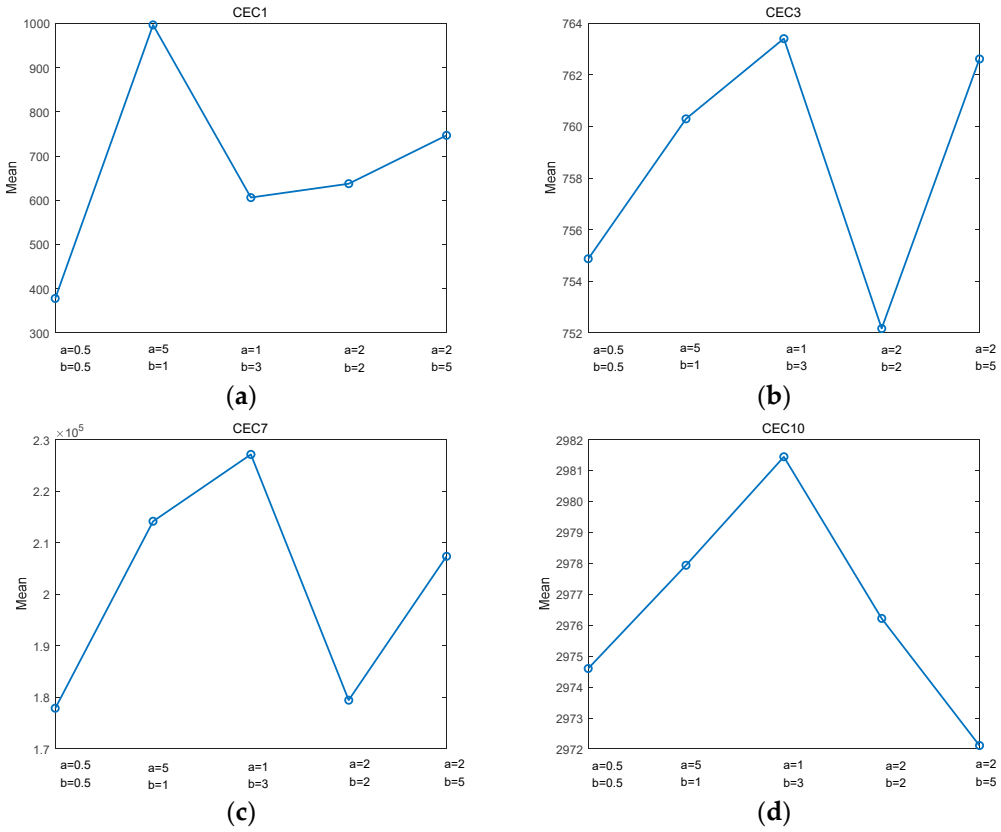


Figure 4. Sensitivity analysis of the MDBO's parameters. (**a**) CEC-1; (**b**) CEC-3; (**c**) CEC-7; (**d**) CEC-10.

Table 2. Twelve benchmark test function results in different dimensions.

Fun No.	Name	30 *dim*		50 *dim*		100 *dim*	
		Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**
F1	DBO	7.74×10^{-114}	3.88×10^{-113}	3.93×10^{-97}	2.15×10^{-96}	1.07×10^{-111}	5.84×10^{-111}
	POA	8.32×10^{-103}	4.44×10^{-102}	1.64×10^{-99}	8.31×10^{-99}	9.87×10^{-100}	5.39×10^{-99}
	HBA	2.58×10^{-134}	1.39×10^{-133}	1.56×10^{-128}	6.28×10^{-128}	7.48×10^{-122}	2.34×10^{-121}
	SCSO	9.44×10^{-111}	4.99×10^{-110}	2.70×10^{-109}	1.22×10^{-108}	1.44×10^{-103}	5.79×10^{-103}
	GCHHO	2.28×10^{-91}	8.84×10^{-91}	3.95×10^{-92}	2.16×10^{-91}	2.28×10^{-94}	1.15×10^{-93}
	ISSA	4.45×10^{-14}	1.08×10^{-14}	7.07×10^{-14}	1.46×10^{-14}	1.40×10^{-13}	1.77×10^{-14}
	MCFOA	6.53×10^{-11}	1.14×10^{-10}	1.90×10^{-10}	3.69×10^{-10}	7.94×10^{-10}	1.22×10^{-9}
	MDBO	**0**	**0**	**0**	**0**	**0**	**0**
F2	DBO	9.65×10^{-57}	5.28×10^{-56}	3.97×10^{-57}	2.18×10^{-56}	5.75×10^{-59}	2.40×10^{-58}
	POA	1.11×10^{-49}	6.05×10^{-49}	1.92×10^{-51}	8.90×10^{-51}	4.73×10^{-51}	1.71×10^{-50}
	HBA	7.02×10^{-72}	3.11×10^{-71}	2.61×10^{-69}	5.42×10^{-69}	8.24×10^{-65}	2.58×10^{-64}
	SCSO	3.31×10^{-60}	1.13×10^{-59}	6.72×10^{-59}	1.24×10^{-58}	5.81×10^{-55}	2.40×10^{-54}
	GCHHO	2.03×10^{-47}	1.11×10^{-46}	3.83×10^{-49}	1.65×10^{-48}	6.52×10^{-48}	2.78×10^{-47}
	ISSA	8.83×10^{-8}	1.34×10^{-8}	1.46×10^{-7}	1.64×10^{-8}	2.97×10^{-7}	2.15×10^{-8}
	MCFOA	3.16×10^{-4}	2.91×10^{-4}	5.40×10^{-4}	4.63×10^{-4}	1.22×10^{-3}	1.19×10^{-3}
	MDBO	**0**	**0**	**0**	**0**	**0**	**0**

Table 2. *Cont.*

Fun No.	Name	30 *dim*		50 *dim*		100 *dim*	
		Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**
F3	DBO	2.70×10^{-29}	1.48×10^{-28}	1.93×10^{-39}	1.06×10^{-38}	4.75×10^{-11}	2.60×10^{-10}
	POA	6.02×10^{-99}	3.30×10^{-98}	3.62×10^{-100}	1.81×10^{-99}	6.01×10^{-98}	2.24×10^{-97}
	HBA	7.39×10^{-96}	3.12×10^{-95}	4.41×10^{-87}	2.36×10^{-86}	4.22×10^{-74}	2.29×10^{-73}
	SCSO	8.22×10^{-99}	2.38×10^{-98}	2.65×10^{-94}	7.68×10^{-94}	5.04×10^{-89}	2.20×10^{-88}
	GCHHO	6.90×10^{-58}	3.35×10^{-57}	6.52×10^{-49}	3.55×10^{-48}	7.71×10^{-38}	4.22×10^{-37}
	ISSA	4.81×10^{-13}	5.24×10^{-13}	1.16×10^{-12}	1.47×10^{-12}	4.20×10^{-12}	3.91×10^{-12}
	MCFOA	1.62×10^{-8}	2.07×10^{-8}	9.54×10^{-8}	1.32×10^{-7}	5.32×10^{-7}	8.46×10^{-7}
	MDBO	**0**	**0**	**0**	**0**	**0**	**0**
F4	DBO	7.33×10^{-58}	3.94×10^{-57}	2.15×10^{-50}	1.18×10^{-49}	6.11×10^{-50}	1.86×10^{-49}
	POA	5.98×10^{-52}	2.82×10^{-51}	1.61×10^{-51}	7.11×10^{-51}	6.17×10^{-50}	2.84×10^{-49}
	HBA	1.53×10^{-56}	7.64×10^{-56}	7.69×10^{-50}	1.86×10^{-49}	4.23×10^{-39}	1.71×10^{-38}
	SCSO	3.79×10^{-51}	1.45×10^{-50}	4.92×10^{-49}	1.58×10^{-48}	1.24×10^{-47}	5.76×10^{-47}
	GCHHO	1.60×10^{-46}	7.28×10^{-46}	8.34×10^{-45}	3.60×10^{-44}	6.73×10^{-45}	2.56×10^{-44}
	ISSA	8.34×10^{-8}	1.33×10^{-8}	9.09×10^{-8}	1.47×10^{-8}	1.02×10^{-7}	9.84×10^{-9}
	MCFOA	3.17×10^{-6}	3.01×10^{-6}	6.04×10^{-6}	7.66×10^{-6}	8.58×10^{-6}	6.83×10^{-6}
	MDBO	**0**	**0**	**0**	**0**	**0**	**0**
F5	DBO	3.13×10^{-23}	1.70×10^{-22}	8.97×10^{-2}	4.91×10^{-1}	4.91×10^{1}	1.74×10^{2}
	POA	6.38×10^{-103}	3.49×10^{-102}	2.68×10^{-103}	1.44×10^{-102}	3.59×10^{-97}	1.97×10^{-96}
	HBA	8.52×10^{-61}	3.29×10^{-60}	1.33×10^{-24}	7.27×10^{-24}	3.42×10^{-4}	1.52×10^{-3}
	SCSO	1.67×10^{-92}	6.08×10^{-92}	1.05×10^{-86}	5.00×10^{-86}	1.09×10^{-72}	5.90×10^{-72}
	GCHHO	1.26×10^{-33}	6.88×10^{-33}	5.00×10^{-29}	2.69×10^{-28}	1.92×10^{-5}	1.05×10^{-4}
	ISSA	5.35×10^{-15}	1.64×10^{-14}	1.32×10^{-14}	3.03×10^{-14}	4.18×10^{-14}	1.10×10^{-13}
	MCFOA	9.13×10^{-6}	1.01×10^{-5}	8.72×10^{-5}	9.45×10^{-5}	1.09×10^{-3}	1.71×10^{-3}
	MDBO	**0**	**0**	**0**	**0**	**0**	**0**
F6	DBO	9.15×10^{-3}	4.53×10^{-2}	2.97×10^{-1}	2.69×10^{-1}	4.68×10^{0}	7.99×10^{-1}
	POA	2.78×10^{0}	5.92×10^{-1}	5.59×10^{0}	8.21×10^{-1}	1.46×10^{1}	1.08×10^{0}
	HBA	8.62×10^{-3}	4.56×10^{-2}	8.89×10^{-1}	3.71×10^{-1}	8.27×10^{0}	9.39×10^{-1}
	SCSO	2.06×10^{0}	5.98×10^{-1}	4.93×10^{0}	7.74×10^{-1}	1.43×10^{1}	1.34×10^{0}
	GCHHO	7.08×10^{-7}	6.46×10^{-7}	1.65×10^{-5}	1.12×10^{-5}	2.50×10^{-4}	1.61×10^{-4}
	ISSA	3.03×10^{0}	4.35×10^{-1}	7.19×10^{0}	6.75×10^{-1}	1.87×10^{1}	8.98×10^{-1}
	MCFOA	6.75×10^{0}	9.29×10^{-2}	1.12×10^{1}	1.66×10^{-1}	2.26×10^{1}	2.59×10^{-1}
	MDBO	**0**	**0**	**0**	**0**	**0**	**0**
F7	DBO	1.04×10^{-3}	6.94×10^{-4}	1.21×10^{-3}	1.01×10^{-3}	1.59×10^{-3}	1.02×10^{-3}
	POA	2.26×10^{-4}	1.62×10^{-4}	1.97×10^{-4}	1.42×10^{-4}	1.56×10^{-4}	8.67×10^{-5}
	HBA	3.02×10^{-4}	1.99×10^{-4}	3.91×10^{-4}	3.25×10^{-4}	5.31×10^{-4}	4.21×10^{-4}
	SCSO	8.98×10^{-5}	8.64×10^{-5}	1.79×10^{-4}	3.71×10^{-4}	2.29×10^{-4}	2.99×10^{-4}
	GCHHO	2.90×10^{-4}	2.73×10^{-4}	2.49×10^{-4}	2.28×10^{-4}	4.19×10^{-4}	3.78×10^{-4}
	ISSA	9.43×10^{-5}	7.38×10^{-5}	1.08×10^{-4}	9.09×10^{-5}	1.04×10^{-4}	1.46×10^{-4}
	MCFOA	2.15×10^{-3}	1.54×10^{-3}	2.61×10^{-3}	2.80×10^{-3}	3.73×10^{-3}	3.52×10^{-3}
	MDBO	$\mathbf{2.81 \times 10^{-5}}$	$\mathbf{2.13 \times 10^{-5}}$	$\mathbf{3.04 \times 10^{-5}}$	$\mathbf{2.18 \times 10^{-5}}$	$\mathbf{3.50 \times 10^{-5}}$	$\mathbf{2.40 \times 10^{-5}}$
F8	DBO	2.85×10^{1}	1.02×10^{2}	3.44×10^{3}	3.74×10^{3}	8.97×10^{4}	2.61×10^{4}
	POA	8.65×10^{2}	4.93×10^{2}	6.20×10^{3}	1.63×10^{3}	8.63×10^{4}	1.57×10^{4}
	HBA	2.84×10^{2}	3.21×10^{2}	5.67×10^{3}	2.29×10^{3}	1.16×10^{5}	2.35×10^{4}
	SCSO	2.17×10^{3}	1.05×10^{3}	1.19×10^{4}	3.20×10^{3}	1.41×10^{5}	3.39×10^{4}
	GCHHO	1.74×10^{0}	4.61×10^{0}	2.61×10^{1}	2.57×10^{1}	1.87×10^{3}	5.35×10^{2}
	ISSA	3.18×10^{3}	4.46×10^{2}	1.80×10^{4}	1.19×10^{3}	1.73×10^{5}	1.05×10^{4}
	MCFOA	9.36×10^{3}	1.59×10^{2}	4.27×10^{4}	2.68×10^{2}	3.37×10^{5}	1.79×10^{3}
	MDBO	$\mathbf{2.29 \times 10^{-7}}$	$\mathbf{1.99 \times 10^{-7}}$	$\mathbf{1.29 \times 10^{-5}}$	$\mathbf{1.07 \times 10^{-5}}$	$\mathbf{3.07 \times 10^{-3}}$	$\mathbf{4.26 \times 10^{-3}}$

Table 2. *Cont.*

Fun No.	Name	30 *dim*		50 *dim*		100 *dim*	
		Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**
F9	DBO	9.96×10^{-2}	5.45×10^{-1}	**0**	**0**	2.32×10^{0}	1.27×10^{1}
	POA	**0**	**0**	**0**	**0**	**0**	**0**
	HBA	**0**	**0**	**0**	**0**	**0**	**0**
	SCSO	**0**	**0**	**0**	**0**	**0**	**0**
	GCHHO	**0**	**0**	**0**	**0**	**0**	**0**
	ISSA	**0**	**0**	**0**	**0**	**0**	**0**
	MCFOA	4.68×10^{-6}	8.78×10^{-6}	8.24×10^{-6}	1.61×10^{-5}	2.47×10^{-5}	3.87×10^{-5}
	MDBO	**0**	**0**	**0**	**0**	**0**	**0**
F10	DBO	1.01×10^{-15}	6.49×10^{-16}	8.88×10^{-16}	**0**	1.01×10^{-15}	6.49×10^{-16}
	POA	3.61×10^{-15}	1.53×10^{-15}	3.97×10^{-15}	1.23×10^{-15}	3.85×10^{-15}	1.35×10^{-15}
	HBA	6.64×10^{-1}	3.64×10^{0}	2.66×10^{0}	6.89×10^{0}	3.32×10^{0}	7.55×10^{0}
	SCSO	$\mathbf{8.88 \times 10^{-16}}$	**0**	$\mathbf{8.88 \times 10^{-16}}$	**0**	$\mathbf{8.88 \times 10^{-16}}$	**0**
	GCHHO	$\mathbf{8.88 \times 10^{-16}}$	**0**	$\mathbf{8.88 \times 10^{-16}}$	**0**	$\mathbf{8.88 \times 10^{-16}}$	**0**
	ISSA	4.84×10^{-8}	4.18×10^{-9}	4.69×10^{-8}	3.79×10^{-9}	4.75×10^{-8}	2.78×10^{-9}
	MCFOA	1.74×10^{-5}	1.45×10^{-5}	1.74×10^{-5}	1.88×10^{-5}	1.50×10^{-5}	1.58×10^{-5}
	MDBO	$\mathbf{8.88 \times 10^{-16}}$	**0**	$\mathbf{8.88 \times 10^{-16}}$	**0**	$\mathbf{8.88 \times 10^{-16}}$	**0**
F11	DBO	1.80×10^{-3}	9.87×10^{-3}	**0**	**0**	**0**	**0**
	POA	**0**	**0**	**0**	**0**	**0**	**0**
	HBA	**0**	**0**	**0**	**0**	**0**	**0**
	SCSO	**0**	**0**	**0**	**0**	**0**	**0**
	GCHHO	**0**	**0**	**0**	**0**	**0**	**0**
	ISSA	9.89×10^{-14}	4.25×10^{-14}	1.01×10^{-13}	4.09×10^{-14}	1.24×10^{-13}	3.34×10^{-14}
	MCFOA	1.74×10^{-13}	3.89×10^{-13}	1.74×10^{-13}	3.92×10^{-13}	2.69×10^{-13}	4.75×10^{-13}
	MDBO	**0**	**0**	**0**	**0**	**0**	**0**
F12	DBO	5.13×10^{-4}	1.64×10^{-3}	4.77×10^{-3}	6.03×10^{-3}	6.45×10^{-2}	2.34×10^{-2}
	POA	1.61×10^{-1}	8.04×10^{-2}	2.81×10^{-1}	8.59×10^{-2}	4.74×10^{-1}	8.76×10^{-2}
	HBA	4.44×10^{-4}	1.64×10^{-3}	1.88×10^{-2}	8.56×10^{-3}	1.43×10^{-1}	5.33×10^{-2}
	SCSO	9.95×10^{-2}	4.05×10^{-2}	2.06×10^{-1}	5.90×10^{-2}	3.77×10^{-1}	7.31×10^{-2}
	GCHHO	1.34×10^{-7}	1.49×10^{-7}	5.88×10^{-7}	6.09×10^{-7}	1.66×10^{-6}	1.10×10^{-6}
	ISSA	2.35×10^{-1}	4.33×10^{-2}	4.13×10^{-1}	6.52×10^{-2}	6.38×10^{-1}	6.17×10^{-2}
	MCFOA	1.33×10^{0}	1.81×10^{-1}	1.23×10^{0}	7.90×10^{-2}	1.13×10^{0}	2.51×10^{-2}
	MDBO	$\mathbf{1.57 \times 10^{-32}}$	$\mathbf{5.57 \times 10^{-48}}$	$\mathbf{9.42 \times 10^{-33}}$	$\mathbf{2.78 \times 10^{-48}}$	$\mathbf{4.71 \times 10^{-33}}$	$\mathbf{1.39 \times 10^{-48}}$

4.4. Convergence Curve Analysis

Convergence analysis plays a vital role in evaluating the ability of the local exploitation and global exploration of the algorithm. Figure 5 shows the convergence curves of DBO, POA, HBA, SCSO, ISSA, GCHHO, MCFOA, and MDBO on functions F1–F12. As shown in Figure 5, MDBO's optimization ability was the best in 11 (F1–F6, F8–F12) out of 12 functions. For functions F1–F6, the MDBO's curves dropped rapidly in early iterations and kept a fast convergence rate to the optimal solution. Notably, F1–F5 are continuous unimodal test functions, and MDBO can quickly search for the optimal theoretical value of 0. Since the vertical coordinates of the convergence curve were generated by the logarithmic scale, the accuracy of the displayed magnitude was 10^{-300}. For F4, the function curve of MDBO showed an inflection point because crossover operators can help MDBO re-exploit the optimization precision. For function F7, although the progress of convergence to 500 generations was not as good as that of ISSA, the average number of iterations of function curve convergence to the optimal value was the least and converged to the optimal value in about 100 generations. In F9 and F11, MDBO obtained the optimal global value of 0. The curve broke during iterations because the figure showed an average best value in logarithmic. For functions F8, F10, and F12, MDBO exhibited a more competitive performance than the other comparison algorithms. In summary, convergence analysis proved that MDBO had a higher success ratio than the other optimization algorithms.

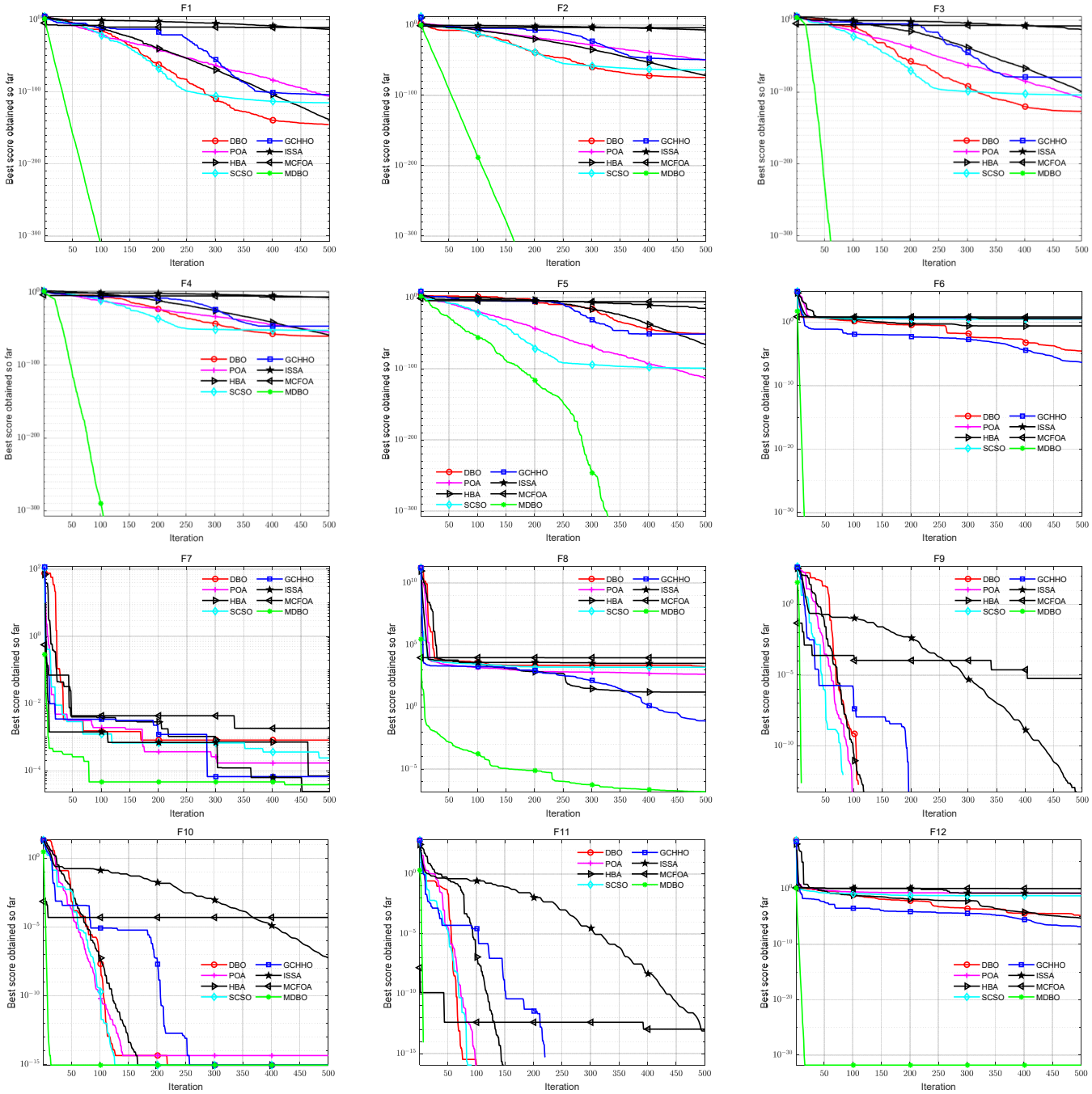


Figure 5. The convergence curves by MDBO on 12 benchmark test functions.

4.5. Wilcoxon Rank-Sum Test

To further test the effectiveness of the MDBO, the Wilcoxon rank sum test [37] was used to determine whether there was a statistical difference. Each algorithm was run 30 times independently, and the data volume met the requirement of statistical analysis. Table 3 shows the p-values of the Wilcoxon rank sum test at the α = 5% significance level. If $p < 0.05$, the original hypothesis is rejected, and the alternative hypothesis is accepted. In Table 3, "+/−/=" indicates the number of MDBO with better/worse/comparable performance compared with other algorithms, respectively, where the ''NaN" markers had comparable performance. In general, most of the p-values of the rank sum test were less than 0.05. This indicates that the performance of MDBO was significantly different from other algorithms. Therefore, it was considered that the proposed MDBO had excellent convergence performance.

Table 3. Wilcoxon rank sum test results.

Fun No.	DBO	POA	HBA	SCSO	GCHHO	ISSA	MCFOA
	p Value R	*p* Value R	*p* Value R	*p* Value R	*p* Value R	*p* Value R	*p* Value R
F1	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
F2	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
F3	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
F4	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
F5	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
F6	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
F7	3.02×10^{-11}	4.08×10^{-11}	5.07×10^{-10}	8.88×10^{-6}	2.83×10^{-8}	6.55×10^{-4}	3.34×10^{-11}
F8	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
F9	8.15×10^{-2}	NaN	NaN	NaN	NaN	NaN	1.21×10^{-12}
F10	NaN	8.99×10^{-11}	1.61×10^{-1}	NaN	NaN	1.21×10^{-12}	1.21×10^{-12}
F11	NaN	NaN	NaN	NaN	NaN	1.21×10^{-12}	1.66×10^{-11}
F12	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
+/−/=	9/1/2	10/0/2	10/0/2	9/0/3	9/0/3	11/0/1	12/0/0

4.6. MDBO's Performance on CEC2021 Suite

To further verify the performance of the MDBO, the challenging CEC 2021 test suite (details in Appendix A, Table A2) was used to test against the seven metaheuristics above-mentioned. Similarly, for CEC2021, the population size was set to 30, along with 500 maximum iterations, and the dimension was set to 20. Each metaheuristic was run independently 30 times to keep fairness and objectivity. The Wilcoxon signed-rank test (signed-rank test) [37] was also carried out. The symbol "gm" is the score representing the difference between the number of symbols "+" and "−".

Table 4 shows the experimental results of the MDBO and other optimizers. In Table 4, the proposed MDBO ranked first in CEC-1, CEC-3, CEC-4, and CEC-6. For CEC-2 and CEC-9, MDBO achieved the best mean results compared with other algorithms. For CEC-10, the mean value of MDBO was second only to HBA, but the standard deviation was smaller and more stable than HBA. The "gm" score showed that the MDBO differed significantly from other algorithms in the vast majority of cases. Overall, the results of the CEC2021 test functions show that the MDBO is effective and suitable for some engineering problems.

Table 4. CEC 2021 test function results.

Fun No.		DBO	POA	HBA	SCSO	GCHHO	ISSA	MCFOA	MDBO
CEC-1	Mean	3.62×10^{7}	6.57×10^{9}	5.71×10^{3}	2.49×10^{9}	4.20×10^{3}	1.04×10^{10}	5.05×10^{10}	$\mathbf{8.99 \times 10^{2}}$
	Std.	3.47×10^{7}	3.63×10^{9}	4.24×10^{3}	2.08×10^{9}	3.42×10^{3}	2.34×10^{9}	6.21×10^{8}	$\mathbf{1.54 \times 10^{3}}$
	p-value	3.02×10^{-11}	3.02×10^{-11}	3.08×10^{8}	3.02×10^{-11}	1.47×10^{-7}	3.02×10^{-11}	3.02×10^{-11}	N/A
	Signed-rank test	+	+	+	+	+	+	+	
CEC-2	Mean	3.50×10^{3}	3.12×10^{3}	2.86×10^{3}	3.67×10^{3}	3.02×10^{3}	5.79×10^{3}	9.27×10^{3}	$\mathbf{1.78 \times 10^{3}}$
	Std.	5.78×10^{2}	4.21×10^{2}	7.74×10^{2}	4.92×10^{2}	5.90×10^{2}	2.99×10^{2}	$\mathbf{1.86 \times 10^{2}}$	2.59×10^{2}
	p-value	3.02×10^{-11}	3.69×10^{-11}	3.96×10^{-8}	3.02×10^{-11}	5.07×10^{-10}	3.02×10^{-11}	3.02×10^{-11}	N/A
	Signed-rank test	+	+	+	+	+	+	+	
CEC-3	Mean	8.40×10^{2}	9.39×10^{2}	8.00×10^{2}	9.06×10^{2}	8.72×10^{2}	9.72×10^{2}	1.18×10^{3}	$\mathbf{7.58 \times 10^{2}}$
	Std.	4.23×10^{1}	2.99×10^{1}	2.89×10^{1}	3.52×10^{1}	3.53×10^{1}	3.14×10^{1}	5.25×10^{0}	$\mathbf{1.34 \times 10^{1}}$
	p-value	6.70×10^{-11}	3.02×10^{-11}	1.10×10^{-8}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	N/A
	Signed-rank test	+	+	+	+	+	+	+	
CEC-4	Mean	1.94×10^{3}	3.51×10^{3}	1.91×10^{3}	3.47×10^{3}	1.91×10^{3}	2.15×10^{4}	3.86×10^{7}	$\mathbf{1.91 \times 10^{3}}$
	Std.	6.81×10^{1}	1.78×10^{3}	4.80×10^{0}	3.37×10^{3}	5.67×10^{0}	1.48×10^{4}	2.19×10^{6}	$\mathbf{4.73 \times 10^{0}}$
	p-value	9.83×10^{-8}	3.02×10^{-11}	6.20×10^{-1}	3.02×10^{-11}	1.24×10^{-3}	3.02×10^{-11}	3.02×10^{-11}	N/A
	Signed-rank test	+	+	−	+	+	+	+	
CEC-5	Mean	1.22×10^{6}	1.35×10^{5}	$\mathbf{1.64 \times 10^{5}}$	7.79×10^{5}	4.75×10^{5}	2.04×10^{6}	4.81×10^{7}	3.21×10^{5}
	Std.	9.62×10^{5}	9.98×10^{4}	$\mathbf{1.21 \times 10^{5}}$	5.78×10^{5}	2.66×10^{5}	6.98×10^{5}	5.83×10^{6}	1.72×10^{5}
	p-value	3.26×10^{-7}	5.27×10^{-5}	4.98×10^{-4}	3.18×10^{-4}	1.99×10^{-2}	3.02×10^{-11}	3.02×10^{-11}	N/A
	Signed-rank test	+	+	+	+	+	+	+	

Table 4. *Cont.*

Fun No.		DBO	POA	HBA	SCSO	GCHHO	ISSA	MCFOA	MDBO
CEC-6	Mean	2.24×10^{3}	2.24×10^{3}	2.10×10^{3}	2.18×10^{3}	2.00×10^{3}	2.92×10^{3}	7.66×10^{3}	$\mathbf{1.67 \times 10^{3}}$
	Std.	2.53×10^{2}	1.88×10^{2}	3.42×10^{2}	2.22×10^{2}	2.01×10^{2}	2.43×10^{2}	1.54×10^{2}	$\mathbf{6.62 \times 10^{1}}$
	p-value	4.08×10^{-11}	3.34×10^{-11}	1.96×10^{-10}	3.34×10^{-11}	4.50×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	N/A
	Signed-rank test	+	+	+	+	+	+	+	
CEC-7	Mean	6.19×10^{5}	$\mathbf{2.60 \times 10^{4}}$	9.52×10^{4}	2.56×10^{5}	1.37×10^{5}	1.18×10^{6}	7.69×10^{8}	2.03×10^{5}
	Std.	7.71×10^{5}	$\mathbf{3.27 \times 10^{4}}$	8.49×10^{4}	3.08×10^{5}	9.82×10^{4}	5.94×10^{5}	2.63×10^{7}	1.32×10^{5}
	p-value	1.33×10^{-2}	5.97×10^{-9}	6.20×10^{-4}	6.73×10^{-1}	5.37×10^{-2}	9.92×10^{-11}	3.02×10^{-11}	N/A
	Signed-rank test	+	+	+	−	−	+	+	
CEC-8	Mean	$\mathbf{2.38 \times 10^{3}}$	3.12×10^{3}	2.84×10^{3}	2.93×10^{3}	2.41×10^{3}	3.60×10^{3}	9.53×10^{3}	2.51×10^{3}
	Std.	3.15×10^{2}	8.44×10^{2}	1.48×10^{3}	9.68×10^{2}	5.57×10^{2}	6.70×10^{2}	$\mathbf{1.26 \times 10^{2}}$	6.20×10^{2}
	p-value	1.25×10^{-4}	5.09×10^{-8}	9.03×10^{-4}	6.01×10^{-8}	2.32×10^{-2}	8.35×10^{-8}	3.02×10^{-11}	N/A
	Signed-rank test	+	+	+	+	+	+	+	
CEC-9	Mean	3.00×10^{3}	3.01×10^{3}	2.96×10^{3}	2.94×10^{3}	2.94×10^{3}	3.04×10^{3}	4.55×10^{3}	$\mathbf{2.92 \times 10^{3}}$
	Std.	8.21×10^{1}	5.69×10^{1}	1.29×10^{2}	4.64×10^{1}	4.87×10^{1}	2.59×10^{1}	$\mathbf{1.86 \times 10^{1}}$	9.77×10^{1}
	p-value	1.11×10^{-3}	1.09×10^{-5}	6.52×10^{-1}	5.30×10^{-1}	5.49×10^{-1}	2.92×10^{-9}	3.02×10^{-11}	N/A
	Signed-rank test	+	+	−	−	−	+	+	
CEC-10	Mean	2.98×10^{3}	3.12×10^{3}	$\mathbf{2.96 \times 10^{3}}$	3.07×10^{3}	2.98×10^{3}	3.56×10^{3}	1.12×10^{4}	2.99×10^{3}
	Std.	4.91×10^{1}	1.24×10^{2}	4.07×10^{1}	7.76×10^{1}	3.62×10^{1}	1.32×10^{2}	1.95×10^{2}	$\mathbf{2.24 \times 10^{1}}$
	p-value	9.93×10^{-2}	1.85×10^{-8}	6.91×10^{-4}	2.15×10^{-6}	2.23×10^{-1}	3.02×10^{-11}	3.02×10^{-11}	N/A
	Signed-rank test	−	+	+	+	−	=	+	
+/−/=/gm					62/8/0/54				

5. UAV Path-Planning Model

5.1. Environment Model

In the route planning of an UAV inspection in oil and gas plants, it is vital to create an appropriate environment model as it will improve the efficiency of the optimization algorithm. Considering some objects as obstacles in oil and gas plants such as complex pipelines, oil wells, signal towers, and so on, this paper adopted the geometric description method to establish the three-dimensional environment model. In this model, the obstacles are described by cuboids of different sizes. The transformation process of the model is shown in Figure 6. In addition, compared with the military UAV, the inspection UAV does not need to consider threats such as missiles, radar, and anti-aircraft guns. After ensuring the limitations of the UAV, the final optimal path is generated according to the surrounding environment and the task requirements to complete the inspection task. In order to prevent the UAV from colliding with obstacles during the flight, the safety threshold R_{safe} is added to the length, width, and height of different cuboids.

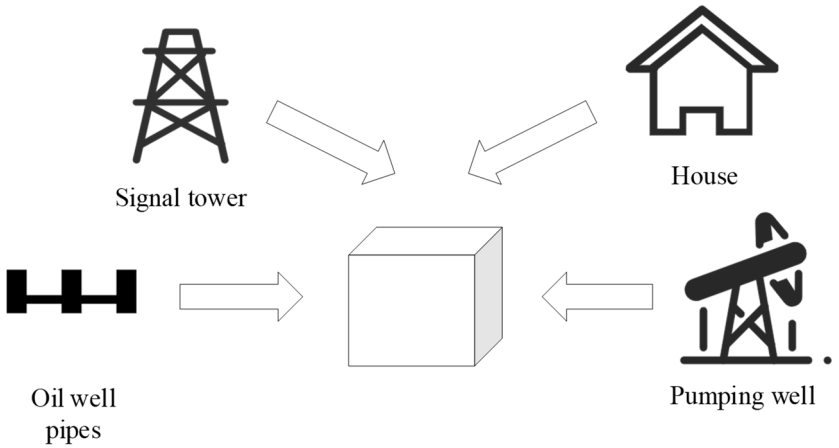


Figure 6. The environment model of oil and gas plants.

5.2. Path Representation

In this paper, it was assumed that the planning path had a start point *S* and an end point *E*. During the inspection, the UAV needs to traverse some task points with no collisions. Assume that the path can be represented as a series of discrete points such as

$(p_0,\ p_1,\ p_2,\ \cdots,\ p_k, \cdots,\ p_{n+1}\ )$, where the first and last waypoints are the given start and target point. The coordinates of p_k are $(x_k,\ y_k,\ z_k)$. The generation of the initial path is introduced as follows:

Step 1, confirm the direction of the next point to the starting point S. The direction of the UAV in three-dimensional space can be generated as:

$$DIR = (dir_x, dir_y, dir_z),\ dir_x,\ dir_y,\ dir_z \in [1, 0, -1] \tag{20}$$

where DIR represents the removable direction, and $dir_x,\ dir_y,\ dir_z$ are the direction on the x-axis, y-axis, and z-axis, respectively.

Then, the next movable waypoint is expressed as:

$$\begin{cases} x_{i+1} = x_i + dir_x \cdot rand() \\ y_{i+1} = y_i + dir_y \cdot rand() \\ z_{i+1} = z_i + dir_z \cdot rand() \end{cases} \tag{21}$$

where $(x_i,\ y_i,\ z_i)$ is the position of the ith discrete point.

Step 2, make sure that the generated waypoints stay within the map and do not collide with the obstacles. Therefore, it is necessary to penalize infeasible solutions. The penalty function h_1 is introduced as:

$$h_1 = p_{InObstacles} + p_{OutMap} = 0 \tag{22}$$

where $p_{InObstacles}$ is the point of collision with the obstacles, and p_{OutMap} denotes the point outside the map.

Step 3, find the optimal waypoint that satisfies the conditions and add it to the path, which can be described as follows:

$$\min h_2 = \sqrt{(D_1 + D_2) \cdot pri} \tag{23}$$

$$D_1 = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2} \tag{24}$$

$$D_2 = \sqrt{(x_i - x_{n+1})^2 + (y_i - y_{n+1})^2 + (z_i - z_{n+1})^2} \tag{25}$$

where i is the index of discrete points from 0 to n; pri is a constant value that depends on the scale of the search space; D_1 and D_2 represent the Euclidean distance between the current path point and the next path point and the end point, respectively.

Step 4, repeat the above operation until the UAV flies to the first task point.

Step 5, finally, repeat the above steps until the UAV flies over all task points and generates a complete initial path.

5.3. Cost Function and Performance Constraints

The objective function that evaluates a candidate route should take account of the cost of the path and the performance, which is expressed as follows:

$$\min f(x) = \sum_{m=1}^{3} w_m f_m(x) \tag{26}$$

$$L_i \le x_i \le U_i,\ i = 1,\ 2,\ \cdots,\ D \tag{27}$$

where $f(x)$ represents the overall cost function; w_m is the weight coefficient; $f_1(x)$ to $f_3(x)$ are respectively the costs associated with the path length, flight height, and smoothness. The decision variable is $x = \{x_1,\ x_2,\ \cdots, x_D\}$, and D is the dimension of the problem

space. L_i and U_i are the lower and upper bounds of the search space. The cost function and constraints of the UAV path are described as follows:

(1) Length cost

In the inspection of UAVs, the shorter the path, the less the time and energy consumption. The cost function f_1 is the length of the UAV path, which can be calculated by the Euclidean operator as follows:

$$f_1 = \sum_{k=0}^{N} d_k \tag{28}$$

$$d_k = \sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2 + (z_{k+1} - z_k)^2} \tag{29}$$

where k is the index of discrete points from 0 to N.

(2) Flight altitude cost

During the flight, maintaining a steady altitude can reduce the power consumption. In order to ensure the safety of the flight, the altitude of the UAV is usually limited between two given extrema, the minimum height $h_{\min}$ and the maximum height $h_{\max}$, respectively. Therefore, the cost function f_2 is computed as:

$$f_2 = \sum_{k=0}^{N} h_k \tag{30}$$

$$h_k = \begin{cases} \left| z_k - \frac{(h_{\max} + h_{\min})}{2} \right|, & if\ h_{\min} \le z_k \le h_{\max} \\ \infty, & otherwise \end{cases} \tag{31}$$

where z_k denotes the flight height with respect to the ground, and k is the index of waypoints. It should be mentioned that h_k maintains the average height and penalizes the values outside the range.

(3) Smooth cost

To ensure that the UAV can always maintain a good attitude during the working flight, this paper adopted a smooth cost, consisting of the climb turning angle and climbing rates. As shown in Figure 7, the turning angle ϕ_k is the angle between two adjacent consecutive path segments. If $\vec{e_3}$ is the unit vector in the z-axis direction, the projected vector is calculated as:

$$\overrightarrow{p'_k p'_{k+1}} = \vec{e_3} \times (\overrightarrow{p_k p_{k+1}} \times \vec{e_3}) \tag{32}$$

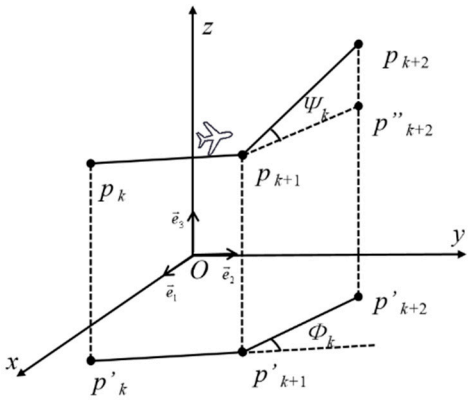


Figure 7. Turning angle and climbing angle calculation.

The turning angle is calculated as:

$$\phi_k = \arctan\left(\frac{\| \overrightarrow{p'_k p'_{k+1}} \times \overrightarrow{p'_{k+1} p'_{k+2}} \|}{\overrightarrow{p'_k p'_{k+1}} \cdot \overrightarrow{p'_{k+1} p'_{k+2}}}\right) \tag{33}$$

where the notation $\cdot$ is a dot product, and $\times$ is a cross product.

The climbing angle ψ_k is the angle between the $p'_k \vec{p}'_{k+1}$ and $p_k \vec{p}_{k+1}$ onto the horizontal plane, and the calculation formula is as follows:

$$\psi_k = \arctan\left(\frac{z_{k+1} - z_k}{\| \overrightarrow{p'_k p'_{k+1}} \|}\right) \tag{34}$$

Then, the smooth cost function f_3 is composed of the turning angle and the climbing rate, which can be calculated as:

$$f_3 = a_1 \cdot \sum_{k=1}^{N-1} \phi_k + a_2 \cdot \sum_{k=1}^{N-1} |\psi_k - \psi_{k-1}| \tag{35}$$

where a_1 and a_2 are the constants.

6. Simulation Experiments and Discussions on UAV Path Planning

6.1. Scenario Setup

The environment region was 1000 m long, 1000 m wide, and 12 m high, with several known obstacles (the coordinates and length, width, and height data of the obstacles are shown in Appendix A, Table A5). The start point and the destination point were [1,950,12], [950,1,1]. Six representative state-of-the-art metaheuristics (the PSO [38], GWO [39], DBO [18], FOA [40], GBO [41], and HPO [42]) were chosen to draw comparisons. For fairness, the population number was set to 30 for all algorithms, and the maximum iteration was set to 100. Each metaheuristic was independently run 20 times. The best value (Best), mean (Mean), and standard deviation (Std) were used as statistical indicators to assess the optimization performance.

6.2. Effect of the Cost Function Parameters

The objective function weights depend on the importance assigned to its different parts. The purpose of the path cost is to ensure that it generates an effective drone flight path. In many scenarios, the turning of drones during the flight is inevitable. Therefore, the weight of the smooth cost is lower than the other costs. This section mainly verifies the performance of MDBO in solving the cost function with different weight combinations. The value of w_3 is 0.1 or 0.2. When w_3 is 0.1, the combination of w_1 and w_2 is {0.7,0.2} or {0.2,0.7}, and when w_3 is 0.2, the combination of w_1 and w_3 is {0.4,0.4} or {0.5,0.3} or {0.3,0.5}. Thus, there are a total of five combinations of design.

The results are given in Table 5. It can be seen that MDBO achieved first place in 11 out of 15 indices in all index tests. For the Std index, MDBO ranked first among the two test combinations. When w_1 = 0.4, w_2 = 0.4, w_3 = 0.2, and w_1 = 0.3, w_2 = 0.5, w_3 = 0.2, and w_1 = 0.5, w_2 = 0.3, w_3 = 0.2, the performance of MDBO was second only to DBO. Although DBO had better standard deviations than MDBO under these three weight combinations, its optimal convergence solution and mean value were not as good as MDBO. In the case of the shortest path length, we believe that the performance of MDBO was still better than DBO. It is worth noting that when w_1 = 0.7, w_2 = 0.2, and w_3 = 0.1, MDBO ranked second only to GBO in terms of the mean value. In particular, when w_1 = 0.5, w_2 = 0.3, and w_3 = 0.2, the performance of MDBO was optimal. Overall, MDBO had good searchability and robustness in all testing scenarios.

Table 5. Comparison of the performance of the five algorithms under different weight combinations.

W		GBO	HPO	GWO	DBO	FOA	PSO	MDBO
$w_1 = 0.7$,	Best	3.12×10^1	8.01×10^1	1.05×10^2	3.01×10^1	4.09×10^1	1.81×10^2	$\mathbf{3.00 \times 10^1}$
$w_2 = 0.2$,	Mean	6.89×10^1	1.78×10^2	2.56×10^2	3.56×10^1	5.95×10^1	2.68×10^2	$\mathbf{3.37 \times 10^1}$
$w_3 = 0.1$	Std	4.54×10^1	6.03×10^1	1.15×10^2	5.36×10^0	1.27×10^1	5.65×10^1	$\mathbf{2.92 \times 10^0}$
$w_1 = 0.2$,	Best	$\mathbf{3.20 \times 10^1}$	1.15×10^2	1.39×10^2	3.56×10^1	4.70×10^1	1.93×10^2	3.36×10^1
$w_2 = 0.7$,	Mean	1.08×10^2	2.63×10^2	2.81×10^2	4.47×10^1	7.91×10^1	3.12×10^2	$\mathbf{4.33 \times 10^1}$
$w_3 = 0.1$	Std	7.53×10^1	8.33×10^1	1.24×10^2	5.74×10^0	1.38×10^1	6.32×10^1	$\mathbf{5.51 \times 10^0}$
$w_1 = 0.4$,	Best	3.09×10^1	1.85×10^2	8.88×10^1	2.97×10^1	4.71×10^1	2.87×10^2	$\mathbf{2.80 \times 10^1}$
$w_2 = 0.4$,	Mean	1.11×10^2	3.60×10^2	4.50×10^2	3.59×10^1	7.96×10^1	4.86×10^2	$\mathbf{3.40 \times 10^1}$
$w_3 = 0.2$	Std	8.11×10^1	1.11×10^2	1.40×10^2	$\mathbf{4.77 \times 10^0}$	2.08×10^1	8.36×10^1	7.62×10^0
$w_1 = 0.3$,	Best	3.56×10^1	1.66×10^2	1.82×10^2	3.10×10^1	5.67×10^1	3.53×10^2	$\mathbf{2.89 \times 10^1}$
$w_2 = 0.5$,	Mean	1.49×10^2	4.13×10^2	4.41×10^2	3.63×10^1	9.26×10^1	5.33×10^2	$\mathbf{3.47 \times 10^1}$
$w_3 = 0.2$	Std	8.66×10^1	1.30×10^2	1.64×10^2	$\mathbf{3.45 \times 10^0}$	2.37×10^1	1.02×10^2	4.39×10^0
$w_1 = 0.5$,	Best	2.99×10^1	1.41×10^2	1.83×10^2	2.85×10^1	5.86×10^1	3.30×10^2	$\mathbf{2.61 \times 10^1}$
$w_2 = 0.3$,	Mean	1.13×10^2	3.14×10^2	4.35×10^2	3.37×10^1	8.72×10^1	4.81×10^2	$\mathbf{3.32 \times 10^1}$
$w_3 = 0.2$	Std	1.11×10^2	1.05×10^2	1.65×10^2	$\mathbf{7.27 \times 10^0}$	1.96×10^1	1.04×10^2	8.63×10^0

6.3. Impact of the Count and Position of Tasks

During the flight of the UAV, the task requirements need to be considered. In other words, the UAV must pass through a series of task points from the starting point and fly to the destination after completing the corresponding task. The number of task points and complexity of the task will affect the execution efficiency of the algorithm. This section mainly compares the search performance of each algorithm with different numbers of task points and coordinates. The number of set targets and their coordinates in this experiment are given in Table 6. Three groups of experiments were set up, and the number of task points in each group was 2, 3, and 4. The constant values of the cost function were the optimal weight combination according to the above experiment. The experimental results are shown in Table 7.

Table 6. Details of the coordinates of the tasks.

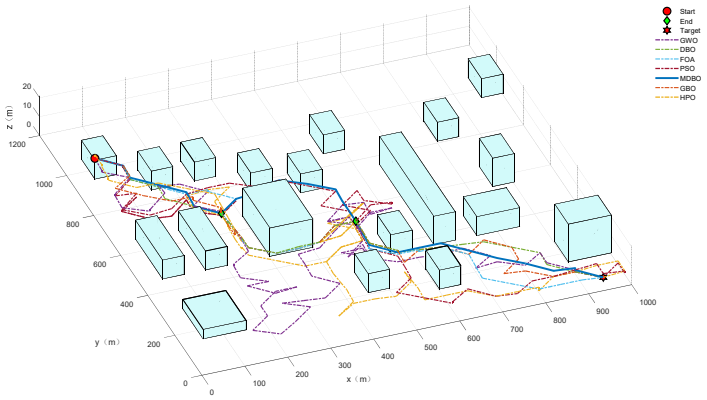
Tasks' Numbers	Target Coordinates
2	[250,650,5], [500,450,10]
3	[250,650,5], [300,300,7], [700,300,10]
4	[250,650,5], [300,300,7], [600,800,12], [900,400,2]

Table 7. Comparison of the performance of the five algorithms with different tasks.

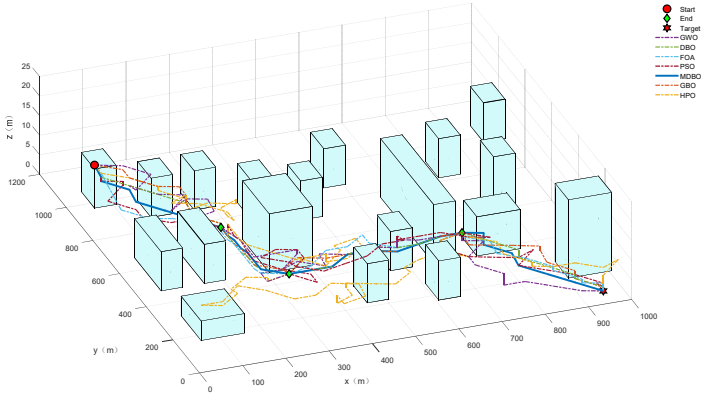
Task Numbers		GBO	HPO	GWO	DBO	FOA	PSO	MDBO
2	Best	5.22×10^1	4.58×10^2	3.22×10^2	5.22×10^1	5.52×10^1	4.96×10^2	$\mathbf{4.58 \times 10^1}$
	Mean	1.06×10^2	6.65×10^2	6.00×10^2	6.18×10^1	6.93×10^1	7.54×10^2	$\mathbf{5.50 \times 10^1}$
	Std	7.78×10^1	1.25×10^2	2.10×10^2	7.76×10^0	7.78×10^0	1.17×10^2	$\mathbf{5.02 \times 10^0}$
3	Best	$\mathbf{3.55 \times 10^1}$	3.28×10^2	2.62×10^2	3.61×10^1	3.85×10^1	4.91×10^2	$\mathbf{3.55 \times 10^1}$
	Mean	6.58×10^1	5.77×10^2	4.91×10^2	4.10×10^1	5.90×10^1	7.11×10^2	$\mathbf{3.60 \times 10^1}$
	Std	4.60×10^1	1.63×10^2	2.49×10^2	6.34×10^0	1.24×10^1	1.82×10^2	$\mathbf{7.38 \times 10^{-1}}$
4	Best	6.23×10^1	8.01×10^2	4.54×10^2	8.21×10^1	1.16×10^2	1.05×10^3	$\mathbf{6.16 \times 10^1}$
	Mean	1.62×10^2	1.04×10^3	8.55×10^2	1.37×10^2	1.50×10^2	1.51×10^3	$\mathbf{9.73 \times 10^1}$
	Std	8.28×10^1	2.08×10^2	2.18×10^2	4.68×10^1	2.16×10^1	2.77×10^2	$\mathbf{3.28 \times 10^1}$

From Table 7, it can be intuitively observed that MDBO achieved the best results in all indices. The planned path length will also increase as the number of tasks increases. Compared with the other algorithms, it still showed a superior performance. The results

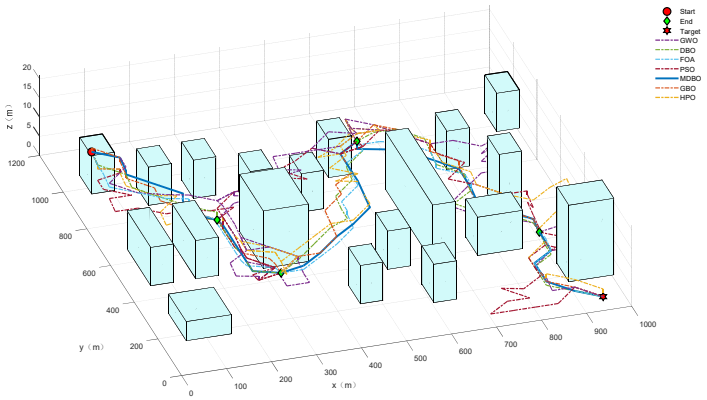
indicate that MDBO can handle both simple and complex tasks. Figure 8 shows a top view of the optimal path generated by all algorithms. From Figure 8, it can be seen that all of the algorithms could successfully find secure paths. As the number of tasks increases and the complexity increases, the path will undergo significant changes. This indicates the complexity of multitasking. Meanwhile, it is evident from the comparison of paths that MDBO had smoother paths and fewer corners at task points, which means that it had a stronger optimization performance than the other algorithms.



(**a**)



(**b**)



(**c**)

Figure 8. The UAV paths of all algorithms under a different number of targets. (**a**) The generated paths when the number of tasks was 2. (**b**) The generated paths when the number of tasks was 3. (**c**) The generated paths when the number of tasks was 4.

6.4. Influence of the Number and Arrangement of Obstacles

The number and arrangement of obstacles in the map will affect the algorithm's efficiency in finding the optimal solution. Moreover, the number of iterations required may increase if there are many obstacles in the presence of a line of sight path between the start and the destination of the UAV. Based on the previous experiments, this section assessed the performance of seven algorithms in different scenarios. We mainly set up three groups of map scenes. The number of obstacles distributed in each group of map scenes was 6, 13, and 19, respectively. The specific information of obstacles in Maps 1–3 including the coordinates and length, width, and height of obstacles are shown in Tables A3–A5 in Appendix A.

The results of the average convergence curves for 100 iterations are plotted in Figures 9–11. It can be seen that in all cases, MDBO always obtained the optimal solution compared to the other algorithms. For Map 1, DBO, GBO, FOA, and MDBO converged rapidly in the early stage and converged to the optimal value of around 50 iterations. The comparison from the final results found that MDBO had the highest solution accuracy, followed by the GBO. It was also found that MDBO converged to the optimal value very quickly at the beginning of the iteration, and the performance improvement was obvious compared to the original DBO. This proves that the proposed search strategies can accelerate the convergence speed and improve the convergence accuracy. For Map 2, it can be seen from Figure 10 that the GBO and FOA outperformed the DBO and MDBO at the beginning of the iterations. However, as the iterations began, DBO and MDBO converged quickly to a minimal value. At the same time, the GBO algorithm fell into a local optimum at a later stage. For Map 3, with more obstacles, the FOA and the MDBO performed the best, with the MDBO converging late to obtain the best accuracy.

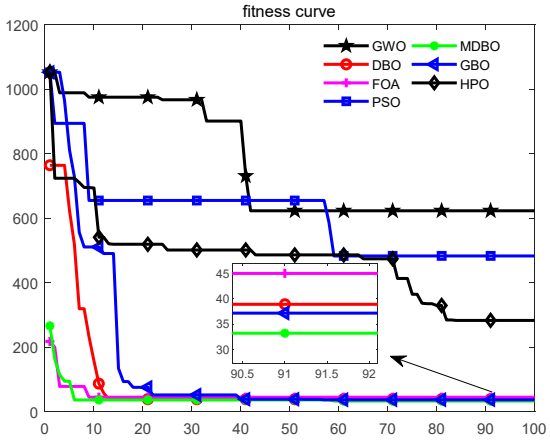


Figure 9. Avg. best cost vs. iteration for Map 1.

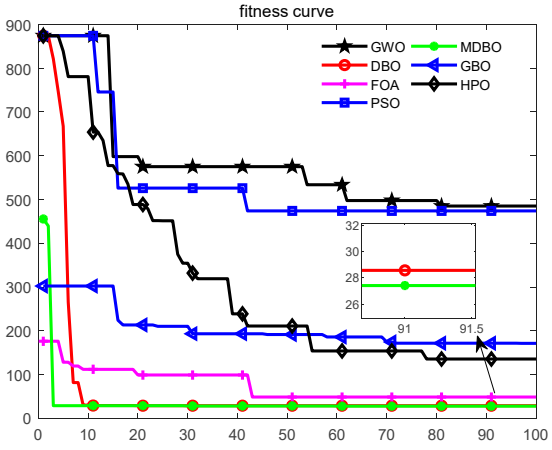


Figure 10. Avg. best cost vs. iteration for Map 2.

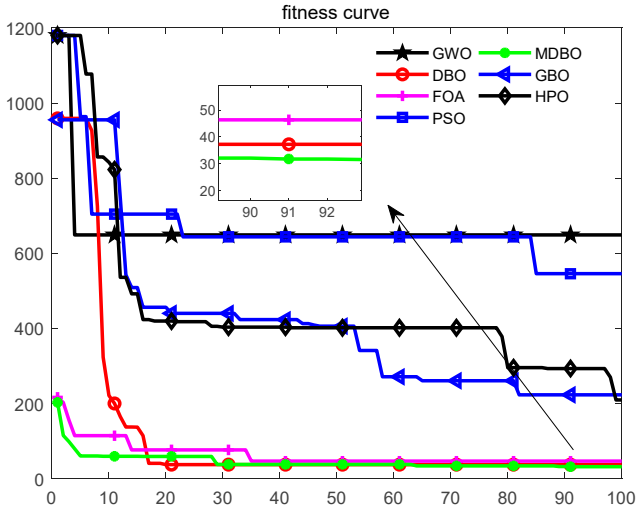


Figure 11. Avg. best cost vs. iteration for Map 3.

On average, the number of iterations for MDBO to converge to the optimal value was 50. In general, MDBO still has a higher convergence accuracy and stronger robustness than the state-of-the-art metaheuristic methods.

7. Conclusions

In this paper, a multi-strategy enhanced dung beetle optimizer (MDBO) was proposed to improve the original algorithm's performance using a reflective learning method, Levy boundary mapping processing, and two different cross-search mechanisms. The proposed MDBO was then used to successfully handle the three-dimensional route planning problem of oil and gas plants.

Tests for MDBO were conducted on 12 benchmark test functions, the Wilcoxon rank sum test, and the CEC2021 suite. The results showed that the MDBO is capable of handling a wide range of optimization problems and is competitive with some advanced metaheuristic algorithms. Second, based on the comparative experiments of UAV path-planning scenarios, the proposed method significantly outperformed other algorithms and achieved satisfactory results in UAV path planning. Finally, the time complexity analysis showed that the proposed MDBO increased in time complexity, so future research will focus on reducing the complexity of the algorithm. In addition, when solving the UAV path-planning problem, the UAVs are required to respond promptly when the mission dynamics change, and the number of UAVs needs to be increased if necessary, which is a limitation of this study.

Future work will focus on multi-UAV cooperation path planning in complex environments. We will also work further on reducing the running time of MDBO and applying it to more complicated optimization problems.

8. Discussion

This study provides a solution for path planning with an improved dung beetle optimizer (MDBO). It was found that the three proposed strategies effectively improved the performance of the original DBO, enabling it to solve various optimization problems. Simulation experiments in the UAV path-planning scenario demonstrated the superiority of the MDBO for path searching.

This paper examined the performance of metaheuristic algorithms such as the dung beetle optimization algorithm in solving various types of optimization problems, with a focus on the MDBO's performance in addressing path-planning problems. First, tests were performed on 12 benchmark functions, the Wilcoxon rank sum test, and the CEC2021 suite. The results show that the three enhancement strategies can improve the original DBO's performance and expand the algorithm's application capabilities. This helps to verify

that different improvement strategies can improve the performance of the original algorithm [23,24] and help achieve satisfactory results for specific engineering issues [22,25,26].

Second, tests on trajectory planning scenarios indicate that the DBO and the MDBO outperformed other advanced comparison algorithms. The results demonstrate the efficiency of the intelligence algorithms in solving path-planning problems [15,20,21]. In contrast to previous research, we focused on the flaws of the intelligent algorithm and aimed to develop a more reasonable search mechanism that is suitable for resolving path optimization problems.

While the findings are not surprising, it is important to understand the question of the performance gaps of the metaheuristic algorithm when applied to engineering problems. This study demonstrates the power of metaheuristic algorithms for a wide range of optimization problems and successful route planning in oil and gas plants provides theoretical support for practical navigation. However, considering the complex application environment of the real world, there will be many dynamic obstacles and changing tasks, which may require multiple UAVs to avoid dynamic obstacles. Therefore, further investigation will be considered with dynamic tasks. We will focus on designing a multi-objective beetle optimization algorithm to optimize the three-dimensional spatial navigation of several UAVs.

Author Contributions: Conceptualization, methodology, writing—original draft, software, writing—review, Q.S.; writing–review, editing, supervision, investigation, M.X.; writing–review, editing, investigation, D.Z.; writing–review, visualization, supervision, funding acquisition, Q.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China, grant number No.62062021, 61872034, 62166006; Natural Science Foundation of Guizhou Province, grant number [2020]1Y254; Guizhou Provincial Science and Technology Projects, grant number Guizhou Science Foundation-ZK [2021] General 335.

Data Availability Statement: Not applicable.

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

Appendix A

Table A1. Twelve benchmark functions.

No.	Function Name	Search Space	Dim	$f_{\min}$
F1	Sphere	[−100,100]	30/50/100	0
F2	Schwefel 2.22	[−10,10]	30/50/100	0
F3	Schwefel 1.2	[−100,100]	30/50/100	0
F4	Schwefel 2.21	[−100,100]	30/50/100	0
F5	Zakharov	[−5,10]	30/50/100	0
F6	Step	[−100,100]	30/50/100	0
F7	Quartic	[−1.28,1.28]	30/50/100	0
F8	Qing	[−500,500]	30/50/100	0
F9	Rastrigin	[−5.12,5.12]	30/50/100	0
F10	Ackley 1	[−32,32]	30/50/100	0
F11	Griewank	[−600,600]	30/50/100	0
F12	Penalized 1	[−50,50]	30/50/100	0

Table A2. Summary of the CEC2021 test suite [32].

	No.	Functions	F_i^*
Unimodal Function	CEC-1	Shifted and Rotated Bent Cigar Function	100
Basic Functions	CEC-2	Shifted and Rotated Schwefel's Function	1100
	CEC-3	Shifted and Rotated Lunacek bi-Rastrigin Function	700
	CEC-4	Expand Rosenbrock's plus Griewangk's Function	1900
Hybrid Functions	CEC-5	Hybrid Function 1 (N = 3)	1700
	CEC-6	Hybrid Function 2 (N = 4)	1600
	CEC-7	Hybrid Function 3 (N = 5)	2100
Composition Functions	CEC-8	Composition Function 1 (N = 3)	2200
	CEC-9	Composition Function 2 (N = 4)	2400
	CEC-10	Composition Function 3 (N = 5)	2500
Search range: $[-100,100]^D$			

Table A3. The data of obstacles on Map 1.

No.	X	Y	Z	L	W	H
1	550	100	0	50	100	10
2	0	400	0	50	200	10
3	300	320	0	50	380	15
4	800	150	0	50	100	15
5	500	350	0	50	100	10
6	50	800	0	50	100	10

Table A4. The data of obstacles on Map 2.

No.	X	Y	Z	L	W	H
1	40	100	0	100	150	5
2	450	350	0	50	100	10
3	850	100	0	100	100	20
4	0	400	0	50	200	10
5	100	400	0	50	200	10
6	260	430	0	100	180	15
7	600	320	0	50	380	15
8	800	500	0	50	100	15
9	430	650	0	50	100	10
10	20	900	0	50	100	10
11	500	800	0	50	100	10
12	450	200	0	50	100	10
13	750	200	0	50	100	10

Table A5. The data of obstacles on Map 3.

No.	X	Y	Z	L	W	H
1	40	100	0	100	150	5
2	400	150	0	50	100	10
3	550	100	0	50	100	10
4	850	100	0	100	100	20
5	0	400	0	50	200	10
6	100	400	0	50	200	10
7	260	430	0	100	180	15
8	500	320	0	50	100	10
9	600	320	0	50	380	15
10	700	300	0	100	100	10
11	800	500	0	50	100	15
12	300	700	0	50	100	10
13	430	650	0	50	100	10
14	20	900	0	50	100	10
15	100	800	0	50	100	10
16	200	800	0	50	100	10
17	500	800	0	50	100	10
18	750	750	0	50	100	10
19	900	900	0	50	100	10

References

1. Jordan, S.; Moore, J.; Hovet, S.; Box, J.; Perry, J.; Kirsche, K.; Lewis, D.; Tse, Z.T.H. State-of-the-art technologies for UAV inspections. *IET Radar Sonar Navig.* **2018**, *12*, 151–164. [CrossRef]
2. Hu, H.; Xiong, K.; Qu, G.; Ni, Q.; Fan, P.; Ben Letaief, K. AoI-Minimal Trajectory Planning and Data Collection in UAV-Assisted Wireless Powered IoT Networks. *IEEE Internet Things J.* **2021**, *8*, 1211–1223. [CrossRef]
3. Yu, X.; Li, C.; Yen, G.G. A knee-guided differential evolution algorithm for unmanned aerial vehicle path planning in disaster management. *Appl. Soft Comput.* **2020**, *98*, 106857. [CrossRef]
4. Lin, L.; Goodrich, M.A. UAV intelligent path planning for Wilderness Search and Rescue. In Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO, USA, 10–15 October 2009; pp. 709–714. [CrossRef]
5. Shin, Y.; Kim, E. Hybrid path planning using positioning risk and artificial potential fields. *Aerosp. Sci. Technol.* **2021**, *112*, 106640. [CrossRef]
6. Huang, S.-K.; Wang, W.-J.; Sun, C.-H. A Path Planning Strategy for Multi-Robot Moving with Path-Priority Order Based on a Generalized Voronoi Diagram. *Appl. Sci.* **2021**, *11*, 9650. [CrossRef]
7. Wang, J.; Li, Y.; Li, R.; Chen, H.; Chu, K. Trajectory planning for UAV navigation in dynamic environments with matrix alignment Dijkstra. *Soft Comput.* **2022**, *26*, 12599–12610. [CrossRef]
8. Zhang, Z.; Wu, J.; Dai, J.; He, C. A Novel Real-Time Penetration Path Planning Algorithm for Stealth UAV in 3D Complex Dynamic Environment. *IEEE Access* **2020**, *8*, 122757–122771. [CrossRef]
9. Lu, L.; Zong, C.; Lei, X.; Chen, B.; Zhao, P. Fixed-Wing UAV Path Planning in a Dynamic Environment via Dynamic RRT Algorithm. *Mech. Mach. Sci.* **2017**, *408*, 271–282. [CrossRef]
10. Liu, Y.; Zheng, Z.; Qin, F.; Zhang, X.; Yao, H. A residual convolutional neural network based approach for real-time path planning. *Knowl.-Based Syst.* **2022**, *242*, 108400. [CrossRef]
11. Chai, X.; Zheng, Z.; Xiao, J.; Yan, L.; Qu, B.; Wen, P.; Wang, H.; Zhou, Y.; Sun, H. Multi-strategy fusion differential evolution algorithm for UAV path planning in complex environment. *Aerosp. Sci. Technol.* **2021**, *121*, 107287. [CrossRef]
12. Pehlivanoglu, Y.V.; Pehlivanoglu, P. An enhanced genetic algorithm for path planning of autonomous UAV in target coverage problems. *Appl. Soft Comput.* **2021**, *112*, 107796. [CrossRef]
13. Phung, M.D.; Ha, Q.P. Safety-enhanced UAV path planning with spherical vector-based particle swarm optimization. *Appl. Soft Comput.* **2021**, *107*, 107376. [CrossRef]
14. Ali, Z.A.; Han, Z.; Hang, W.B. Cooperative Path Planning of Multiple UAVs by using Max–Min Ant Colony Optimization along with Cauchy Mutant Operator. *Fluct. Noise Lett.* **2021**, *20*, 2150002. [CrossRef]
15. Yu, X.; Li, C.; Zhou, J. A constrained differential evolution algorithm to solve UAV path planning in disaster scenarios. *Knowl.-Based Syst.* **2020**, *204*, 106209. [CrossRef]
16. Zhang, X.; Lu, X.; Jia, S.; Li, X. A novel phase angle-encoded fruit fly optimization algorithm with mutation adaptation mechanism applied to UAV path planning. *Appl. Soft Comput.* **2018**, *70*, 371–388. [CrossRef]
17. Dewangan, R.K.; Shukla, A.; Godfrey, W.W. Three dimensional path planning using Grey wolf optimizer for UAVs. *Appl. Intell.* **2019**, *49*, 2201–2217. [CrossRef]

18. Xue, J.; Shen, B. Dung beetle optimizer: A new meta-heuristic algorithm for global optimization. *J. Supercomput.* **2022**, *79*, 7305–7336. [CrossRef]
19. Peres, F.; Castelli, M. Combinatorial Optimization Problems and Metaheuristics: Review, Challenges, Design, and Development. *Appl. Sci.* **2021**, *11*, 6449. [CrossRef]
20. Jain, G.; Yadav, G.; Prakash, D.; Shukla, A.; Tiwari, R. MVO-based path planning scheme with coordination of UAVs in 3-D environment. *J. Comput. Sci.* **2019**, *37*, 101016. [CrossRef]
21. Li, K.; Ge, F.; Han, Y.; Wang, Y.A.; Xu, W. Path planning of multiple UAVs with online changing tasks by an ORPFOA algorithm. *Eng. Appl. Artif. Intell.* **2020**, *94*, 103807. [CrossRef]
22. Zhang, X.; Xu, Y.; Yu, C.; Heidari, A.A.; Li, S.; Chen, H.; Li, C. Gaussian mutational chaotic fruit fly-built optimization and feature selection. *Expert Syst. Appl.* **2020**, *141*, 112976. [CrossRef]
23. Song, S.; Wang, P.; Heidari, A.A.; Wang, M.; Zhao, X.; Chen, H.; He, W.; Xu, S. Dimension decided Harris hawks optimization with Gaussian mutation: Balance analysis and diversity patterns. *Knowl.-Based Syst.* **2021**, *215*, 106425. [CrossRef]
24. Gupta, S.; Deep, K. A novel Random Walk Grey Wolf Optimizer. *Swarm Evol. Comput.* **2019**, *44*, 101–112. [CrossRef]
25. Pichai, S.; Sunat, K.; Chiewchanwattana, S. An Asymmetric Chaotic Competitive Swarm Optimization Algorithm for Feature Selection in High-Dimensional Data. *Symmetry* **2020**, *12*, 1782. [CrossRef]
26. Mikhalev, A.S.; Tynchenko, V.S.; Nelyub, V.A.; Lugovaya, N.M.; Baranov, V.A.; Kukartsev, V.V.; Sergienko, R.B.; Kurashkin, S.O. The Orb-Weaving Spider Algorithm for Training of Recurrent Neural Networks. *Symmetry* **2022**, *14*, 2036. [CrossRef]
27. Almotairi, K.H.; Abualigah, L. Hybrid Reptile Search Algorithm and Remora Optimization Algorithm for Optimization Tasks and Data Clustering. *Symmetry* **2022**, *14*, 458. [CrossRef]
28. Meng, A.-B.; Chen, Y.-C.; Yin, H.; Chen, S.-Z. Crisscross optimization algorithm and its application. *Knowl.-Based Syst.* **2014**, *67*, 218–229. [CrossRef]
29. Yao, X.; Liu, Y.; Lin, G. Evolutionary programming made faster. *IEEE Trans. Evol. Comput.* **1999**, *3*, 82–102. [CrossRef]
30. Cai, L.; Qu, S.; Cheng, G. Two-archive method for aggregation-based many-objective optimization. *Inf. Sci.* **2018**, *422*, 305–317. [CrossRef]
31. Mohammadi-Balani, A.; Nayeri, M.D.; Azar, A.; Taghizadeh-Yazdi, M. Golden eagle optimizer: A nature-inspired metaheuristic algorithm. *Comput. Ind. Eng.* **2021**, *152*, 107050. [CrossRef]
32. Mohamed, A.W.; Sallam, K.M.; Agrawal, P.; Hadi, A.A.; Mohamed, A.K. Evaluating the performance of meta-heuristic algorithms on CEC 2021 benchmark problems. *Neural Comput. Appl.* **2023**, *35*, 1493–1517. [CrossRef]
33. Trojovský, P.; Dehghani, M. Pelican Optimization Algorithm: A Novel Nature-Inspired Algorithm for Engineering Applications. *Sensors* **2022**, *22*, 855. [CrossRef] [PubMed]
34. Seyyedabbasi, A.; Kiani, F. Sand Cat swarm optimization: A nature-inspired algorithm to solve global optimization problems. *Eng. Comput.* **2022**, 1–25. [CrossRef]
35. Hashim, F.A.; Houssein, E.H.; Hussain, K.; Mabrouk, M.S.; Al-Atabany, W. Honey Badger Algorithm: New metaheuristic algorithm for solving optimization problems. *Math. Comput. Simul.* **2022**, *192*, 84–110. [CrossRef]
36. Hegazy, A.E.; Makhlouf, M.A.; El-Tawel, G.S. Improved salp swarm algorithm for feature selection. *J. King Saud Univ.—Comput. Inf. Sci.* **2020**, *32*, 335–344. [CrossRef]
37. Derrac, J.; García, S.; Molina, D.; Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol. Comput.* **2011**, *1*, 3–18. [CrossRef]
38. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; pp. 1942–1948.
39. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
40. Pan, W.-T. A new Fruit Fly Optimization Algorithm: Taking the financial distress model as an example. *Knowl. Based Syst.* **2012**, *26*, 69–74. [CrossRef]
41. Ahmadianfar, I.; Bozorg-Haddad, O.; Chu, X. Gradient-based optimizer: A new metaheuristic optimization algorithm. *Inf. Sci.* **2020**, *540*, 131–159. [CrossRef]
42. Naruei, I.; Keynia, F.; Molahosseini, A.S. Hunter–prey optimization: Algorithm and applications. *Soft Comput.* **2022**, *26*, 1279–1314. [CrossRef]