drones

MDPI

Article

A Hybrid ARO Algorithm and Key Point Retention Strategy Trajectory Optimization for UAV Path Planning

Bei Liu [1], Yuefeng Cai [2,*], Duantengchuan Li [3,*], Ke Lin [4] and Guanghui Xu [1]

[1] School of Electrical and Electronics Engineering, Hubei University of Technology, Wuhan 430068, China; 102200242@hbut.edu.cn (B.L.)
[2] School of Information Management, Wuhan University, Wuhan 430072, China
[3] School of Computer Science, Wuhan University, Wuhan 430072, China
[4] Department of Control Science and Engineering, Harbin Institute of Technology (Shenzhen), Shenzhen 518055, China; schris_lin@stu.hit.edu.cn
* Correspondence: yuefengcai0127@gmail.com (Y.C.); dtclee1222@whu.edu.cn (D.L.)

Abstract: Path planning is a fundamental research issue for enabling autonomous flight in unmanned aerial vehicles (UAVs). An effective path planning algorithm can greatly improve the operational efficiency of UAVs in complex environments like urban and mountainous areas, thus offering more extensive coverage for various tasks. However, existing path planning algorithms often encounter problems such as high computational costs and a tendency to become trapped in local optima in complex 3D environments with multiple constraints. To tackle these problems, this paper introduces a hybrid multi-strategy artificial rabbits optimization (HARO) for efficient and stable UAV path planning in complex environments. To realistically simulate complex scenarios, we introduce spherical and cylindrical obstacle models. The HARO algorithm balances exploration and exploitation phases using a dual exploration switching strategy and a population migration memory mechanism, enhancing search performance and avoiding local optima. Additionally, a key point retention trajectory optimization strategy is proposed to reduce redundant path points, thus lowering flight costs. Experimental results confirm the HARO algorithm's superior search performance, planning more efficient and stable paths in complex environments. The key point retention strategy effectively reduces flight costs during trajectory optimization, thereby enhancing adaptability.

Keywords: UAV path planning; unmanned aerial vehicle; artificial rabbits optimization; hybrid algorithm; trajectory optimization

Citation: Liu, B.; Cai, Y.; Li, D.; Lin, K.; Xu, G. A Hybrid ARO Algorithm and Key Point Retention Strategy Trajectory Optimization for UAV Path Planning. *Drones* **2024**, *8*, 644. https://doi.org/10.3390/drones8110644

Academic Editor: Oleg Yakimenko

Received: 14 September 2024
Revised: 2 November 2024
Accepted: 4 November 2024
Published: 5 November 2024

1. Introduction

During the last ten years, unmanned aerial vehicles (UAVs) have shown substantial potential in diverse domains, including military applications [1,2], agricultural production [3], food delivery services [4,5], and urban environmental monitoring [6], owing to their high autonomy, flexibility, and adaptability. Path planning, a central research topic in UAV autonomous flight, seeks to determine the optimal trajectory from the origin to the destination to fulfill specific mission objectives. Additionally, trajectory optimization is a crucial component of path planning aimed at further improving flight paths. Effective UAV path planning must account for obstacles, energy consumption, flight altitude, and attitude parameters [7], making the computation of an ideal trajectory a highly complex Non-deterministic Polynomial-time hard (NP-hard) optimization problem [8,9]. Developing solutions to this challenge remains a focal point in current research on UAV path planning.

Traditional path planning algorithms, such as Dijkstra's algorithm [10], A* [11], the Probabilistic Roadmap Method (PRM) [12], and the Rapidly-exploring Random Tree (RRT) [13] encounter significant challenges, including high computational costs, slow convergence rates, and susceptibility to local optima when applied to complex, multi-constraint scenarios (NP-hard problems) [14]. Consequently, researchers have reformulated

UAV path planning as a multi-constrained optimization task. Metaheuristic algorithms, as intelligent optimization methods based on biomimicry and evolutionary theories, show significant advantages across various domains [15–19]. Widely recognized classic metaheuristic algorithms include the Genetic algorithm [20], Whale algorithm [21,22], Differential Evolution (DE) algorithm [23] (evolutionary-based), Particle Swarm Optimization (PSO) algorithm [24] and Ant Colony Optimization (ACO) algorithm [25] (swarm intelligence-based). These have been successfully applied in the field of UAV path planning [26–29].

In UAV path planning, optimization algorithms typically encode population individuals as a series of spherical vector coordinates [30], incorporating position and attitude information to represent the complete flight trajectory. However, as dimensionality rises with an increasing number of path nodes, the optimization task becomes exponentially more complex, making the pursuit of a globally stable optimal solution exceptionally challenging. To address this, the metaheuristic-based artificial rabbits optimization (ARO) algorithm [31] has attracted attention for its stable search performance in low- to high-dimensional optimization tasks. The algorithm emulates the survival strategies adopted by rabbit species, featuring a simple and easily implementable structure, and has been widely applied in fields such as prediction [32,33], photovoltaics [34], and natural language processing [35]. However, the ARO algorithm faces challenges, including a propensity towards becoming trapped in local optima, a relatively slow convergence speed, limited strategies, and insufficient information exchange within the population [36]. Therefore, applying the ARO algorithm for UAV path planning in complex scenarios holds significant potential but requires further optimization of algorithm performance.

Additionally, preliminary feasible paths generated by optimization algorithms are often smoothed and optimized using methods such as cubic B-spline interpolation [37] and Thin-plate spline (TPS) [38]. TPS, widely used in non-rigid point set registration, offers high flexibility [39]. However, these methods do not support differentiated optimization based on obstacle density in various regions. We aim to retain more key path points in densely populated obstacle areas to ensure flight safety. In contrast, in relatively sparse obstacle areas, reducing the number of path points appropriately achieves smoother flight trajectories and effectively reduces flight costs. The Douglas–Peucker (DP) algorithm [40], commonly employed for two-dimensional curve simplification, reduces the number of sample points while roughly preserving the trajectory shape. However, applying the DP algorithm directly to UAV path planning presents two challenges: first, the scarcity of specialized schemes tailored for UAV paths; second, its reliance on judgment thresholds, where excessive simplification could lead to the loss of crucial points in obstacle-dense areas, jeopardizing obstacle avoidance and flight safety.

To address these limitations, we propose an enhanced artificial rabbits optimization algorithm (HARO) that uses a hybrid multi-strategy approach, encoding population individuals as spherical vector coordinates for UAV path planning in complex environments. Firstly, we introduce a dual-exploration strategy switching mechanism to address the original algorithm's limitations with single exploration and exploitation strategies. Additionally, to tackle the algorithm's slow convergence, we introduce an elite-guided strategy. Furthermore, to enhance intra-population information exchange and strengthen the algorithm's ability to avoid local optima, we propose a population migration memory mechanism in conjunction with the algorithm [41]. Moreover, previous research often used cylindrical models to simulate environmental obstacles that fail to capture the actual characteristics of obstacles in complex scenarios fully [42,43]. Therefore, we introduce a novel spherical obstacle model in three-dimensional environments, as depicted in Figure 1. While cylindrical models represent buildings and trees in real scenes, spherical obstacle models emulate objects like radar towers, rocks, and certain types of vegetation, providing a more accurate simulation of real-world scenarios. Furthermore, to overcome the shortcomings of existing methods in dealing with varying obstacle density regions, we propose an enhanced algorithm, HARO+. By preserving key points in dense areas and moderately simplifying

paths in sparse regions, HARO+ balances the requirements of safety and smoothness when generating flight trajectories.

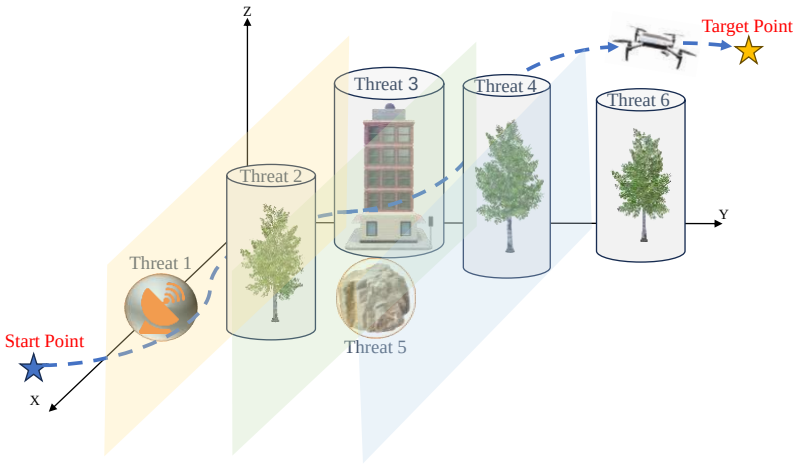

Figure 1. Schematic diagram of the 3D environment simulation for UAVs.

The main contributions of this work are as follows:

- Considering more complex and realistic scenarios, we introduce a spherical obstacle model to better replicate scene characteristics. Subsequently, we propose a hybrid multi-strategy artificial rabbits optimization (HARO+) that utilizes spherical vector coordinates to enhance the efficiency of UAV path planning in intricate environments.
- To enhance early exploration capabilities and flexibility while ensuring better candidate solutions for the development phase, we propose a dual-exploration strategy switching mechanism, balancing exploration and exploitation stages. Additionally, we introduce a population migration memory mechanism to maintain population diversity during iterations, enhancing the ability to avoid falling into local optima.
- Considering the differential treatment of preliminary paths generated by HARO based on obstacle density, we propose the key point retention trajectory optimization strategy, HARO+. This approach effectively generates safe, smooth, and cost-effective UAV flight paths in complex 2D and 3D environments.
- HARO's superior search performance is validated through comparisons with other methods using the CEC2017 test functions and various complex 2D/3D UAV flight scenarios. Additionally, the incorporation of the key point retention trajectory optimization strategy significantly reduces the fitness cost for both HARO+ and other methods (up to 4.5%), with a path point compression rate of approximately 50–80%, further validating the adaptability and compatibility of this optimization strategy in complex environments.

The remainder of this article is organized as follows. Section 2 discusses related work, Section 3 describes the relevant problem definitions, and Section 4 elaborates on the HARO+ algorithm. The numerical experiments and results analysis are provided in Section 5, with conclusions and discussions summarized in Section 6.

2. Related Work

This section reviews key developments in UAV path planning and the artificial rabbits optimization algorithm (ARO), highlighting existing challenges and advancements. We aim to address these limitations by proposing a hybrid multi-strategy algorithm tailored for complex UAV path planning scenarios.

2.1. UAV Path Planning

To improve the efficiency and performance of UAV path planning, researchers have been developing diverse methods to address this complex challenge. Traditional cost-

based path planning algorithms, such as the A* algorithm [11], search for the optimal path by evaluating cost functions but are limited by environmental complexity and heuristic function selection. Sampling-based methods like PRM [12] and RRT algorithms [13] search by randomly generating path points and connecting grids. However, they require numerous samples in high-dimensional space, leading to computational inefficiency. The artificial potential field method [44] guides navigation by simulating physical field effects but suffers from local minima issues. With the rapid development of metaheuristic algorithms, they have found wide applications in path planning and other fields. These algorithms possess strong global search capabilities and adaptability to complex environments, effectively addressing multi-criteria challenges in UAV path planning. Integrating optimization algorithms with reinforcement learning methods [45,46] has provided new optimization approaches for addressing path planning challenges. Accordingly, Qu et al. [14] attempted to generate feasible flight paths in a complex search space by combining reinforcement learning with an improved Grey Wolf algorithm. This method partially improved path planning performance, but due to the inherent complexity of the search space there is still room for further optimization of the generated paths. To address this, Phung et al. [30] proposed the Spherical Coordinate Vector Encoding Particle Swarm Optimization (SPSO) algorithm, which effectively integrates UAV dynamic constraints to transform the search space into configuration space. Although it satisfies multiple constraints and generates high-quality paths, it fails to effectively search solution spaces in high-dimensional environments with an increasing number of path points. Huang et al. [37] pointed out that an increase in waypoints degrades the optimization performance. Consequently, there is an urgent need to develop more robust and effective algorithms capable of rapidly generating safe and smooth UAV flight paths in high-dimensional environments (i.e., with many path points), and providing reliable solutions for UAV autonomous navigation in complex environments.

Additionally, the Douglas–Peucker (DP) algorithm [40], known for handling large numbers of redundant data points, has been recently used for compressing ship trajectories. Bai et al. [47] proposed an adaptive threshold fast DBSCAN algorithm to preserve trajectory feature points for ship trajectory clustering. Tang et al. [48] proposed an AIS trajectory data compression method based on the adaptive threshold DP algorithm. However, this simplification strategy is rarely used in UAV path planning. To address this gap and improve upon current shortcomings, our work focuses on providing a more efficient algorithm and proposes differentiated key point retention strategies for both 2D and 3D scenarios. These strategies can effectively simplify path points and retain key points in areas with varying obstacle complexity, reducing fitness costs and flight energy consumption while ensuring UAV flight safety.

2.2. Artificial Rabbits Optimization Algorithm

The artificial rabbits optimization algorithm [31] represents a newly introduced, biologically inspired metaheuristic algorithm that draws inspiration from the foraging and random hiding strategies observed in rabbits in nature. Because of its straightforward architecture, simplicity of deployment, and robust global search functions in high-dimensional challenges, ARO has been extensively utilized in numerous areas. However, ARO exhibits certain limitations, including the absence of diverse search tactics in the exploration and exploitation phases, restricted communication among the population, poor convergence ability, and susceptibility to local optima. Some research has introduced different enhancement methods to mitigate these shortcomings, generally classified into two groups. One group focuses on strategy improvement, which enhances the algorithm's optimization ability by adjusting control parameters or borrowing strategies from other algorithms. In [49], the authors incorporated methods like Levy flights and opposition-based learning into the binary ARO, enhancing population diversity and optimizing the balance between global exploration and local exploitation. They applied these improvements to the medical diagnosis field. However, opportunities for enhancing performance in complex optimization challenges remain. Cao et al. [36] introduced an approach that adaptively adjusts

the inertia weight based on the existing population distribution, enhancing ARO's performance in support vector machine optimization tasks but still encountering the issue of local optima. Another enhancement method is algorithm fusion, which enhances overall performance by integrating strengths from various algorithms or strategies. Researchers recognize that balancing exploration and exploitation capabilities is crucial in algorithm design. Luo et al. [50] incorporated reinforcement learning strategies into ARO, improving its capability to balance exploration and exploitation, and successfully utilized it for medical image registration. In addition, the improved artificial rabbits optimization (IARO) algorithm proposed by Hu et al. [51], which combines various development strategies of the Orca predation algorithm [52], improves ARO's performance in optimizing the multi-level reduction of spherical NURBS curves. In [32], the White Shark Optimizer [53] is combined with ARO for extracting parameters related to photovoltaic batteries, further demonstrating the advantages of hybrid algorithms.

Overall, these enhancement methods have somewhat improved the ARO algorithm's performance. However, in complex high-dimensional optimization problems, existing versions of the ARO algorithm and its improvements still suffer from limited exploration capabilities, insufficient population diversity, and susceptibility to local optima. Therefore, we introduce a hybrid multi-strategy enhanced ARO algorithm (HARO) to tackle the UAV path planning challenges in intricate environments.

3. Preliminaries

This section formalizes the issues related to UAV path planning and provides an overview of relevant fundamentals to facilitate a better grasp of the proposed approach algorithms.

3.1. Definition of the UAV Path Planning Problem

In intricate UAV path planning scenarios, the crucial task is to find safe, efficient, and logical flight routes through complex terrain and various obstacles. Thus, we represent this issue as a multi-constraint optimization problem, primarily encompassing energy consumption, safety, and flight posture constraints. To holistically address the mentioned constraints, we formulate a total cost function to delineate the optimization objective. The mathematical formulation of the aggregate cost function for the i-th path P is presented as follows [54]:

$$C(P_i) = \sum_{k=1}^{4} \lambda_k C_k(P_i), \tag{1}$$

where C_1 denotes energy cost, C_2 denotes altitude cost, C_3 denotes obstacle cost, and C_4 denotes flight angle cost, with $\lambda_{1\sim4}$ being the weighting coefficients for the corresponding cost factors. Minimizing this overall cost function, C, enables the identification of the optimal UAV flight path while adhering to various constraints. Typically, the flight path of a UAV consists of a starting point, W_s, a target point, W_t, and n path points, represented as follows:

$$\mathcal{P} = \{W_s, W_1, W_2, ..., W_n, W_t\}. \tag{2}$$

Each path point $W_j = (x_j, y_j, z_j)$ corresponds to a coordinate point in three-dimensional space. For two-dimensional plane path planning, the path points can be simplified to plane coordinates $W_j = (x_j, y_j)$. However, using the spherical coordinate system is more intuitive for representing points in three-dimensional space than the Cartesian coordinate system, making it advantageous for path planning and optimization. The spherical coordinate system, with parameters of radius r, inclination angle θ, and azimuth angle φ, better describes the position and attitude changes of the UAV in three-dimensional space. Therefore, we

transform the coordinates of the trajectory points from Cartesian to spherical coordinates using the following formulas:

$$\begin{cases} x_j = x_{j-1} + r_j \sin\theta_j \cos\phi_j \\ y_j = y_{j-1} + r_j \sin\theta_j \sin\phi_j \\ z_j = z_{j-1} + r_j \cos\theta_j \end{cases} \rightarrow \begin{cases} r_j \in \left(0, \frac{2}{n} \cdot |W_s W_t|\right) \\ \theta_j \in \left(-\frac{\pi}{4}, \frac{\pi}{4}\right) \\ \phi_j \in \left(\alpha - \frac{\pi}{4}, \alpha + \frac{\pi}{4}\right), \end{cases} \tag{3}$$

where, n represents the total count of trajectory points, and $|W_s W_t|$ indicates the direct distance between the starting and target points. We define α as the azimuth angle, which measures the directional deviation between the target and the UAV's starting point on the horizontal plane. This approach encodes the i-th path into a vector composed of three components (radial distance, polar angle, azimuth angle) for each of the n trajectory points, as follows:

$$\Omega_i = (r_{i1}, r_{i2}, ..., r_{in}; \theta_{i1}, \theta_{i2}, ..., \theta_{in}; \phi_{i1}, \phi_{i2}, ..., \phi_{in}). \tag{4}$$

By optimizing the positions of these path points, we can obtain the optimal flight path that satisfies all specified constraints. The specific definitions of the above cost factors are detailed as follows.

3.1.1. Energy Constraint

The overall distance of the UAV's flight route must be minimized to reduce energy consumption and flight time. The distance between adjacent path points is measured using the Euclidean distance metric. Therefore, the path length constraint is represented by the following mathematical equation:

$$\begin{aligned} C_1(P_i) &= \sum_{j=0}^{n} \| \overrightarrow{W_{ij} W_{i,j+1}} \| \\ &= \sum_{j=0}^{n} \sqrt{\left(x_{i,j+1} - x_{i,j}\right)^2 + \left(y_{i,j+1} - y_{i,j}\right)^2 + \left(z_{i,j+1} - z_{ij}\right)^2} \end{aligned} \tag{5}$$

where, when $j = 0$, W_{ij} represents the starting point, W_s, and, when $j = n$, $W_{i,j+1}$ represents the target point, W_t. For 2D path planning problems, the z-coordinate term in the equation can be omitted.

3.1.2. Safety Constraint

To ensure a safe UAV flight, its path must smoothly circumvent obstacles. This paper considers two types of obstacles in 3D environments, cylinders and spheres, to simulate diverse obstacle scenarios in real-world situations. Assume there are K obstacles of this type, each with a radius denoted as R_k. Project the path and a particular obstacle onto the xy-plane, as depicted in Figure 2. If the obstacle is cylindrical, d_k denotes the perpendicular distance from the obstacle's center, C_k, to the projection segment of adjacent trajectory points. C represents the UAV's diameter, and D_k denotes the diameter of the danger zone. The cost associated with avoiding obstacles is described by the following formula:

$$C_3(P_i) = \sum_{j=0}^{n} \sum_{k=1}^{K} T_k\left(\overrightarrow{W_{ij} W_{i,j+1}}\right), \tag{6}$$

$$T_k\left(\overrightarrow{W_{ij} W_{i,j+1}}\right) = \begin{cases} 0, & d_k > R_k + C + D_k \\ (R_k + C + D_k) - d_k, & R_k + C < d_k \leq R_k + C + D_k \\ +\infty, & d_k \leq R_k + C. \end{cases} \tag{7}$$

If the obstacle is a sphere with center coordinates (x_k, y_k, z_k), then d_k is calculated by determining the Euclidean distance from each trajectory point to the sphere's center, C_k, using the following formula:

$$d_k = \sum_{j=1}^{n} \sum_{k=1}^{K} \sqrt{(x_{ij} - x_k)^2 + (y_{ij} - y_k)^2 + (z_{ij} - z_k)^2}. \tag{8}$$

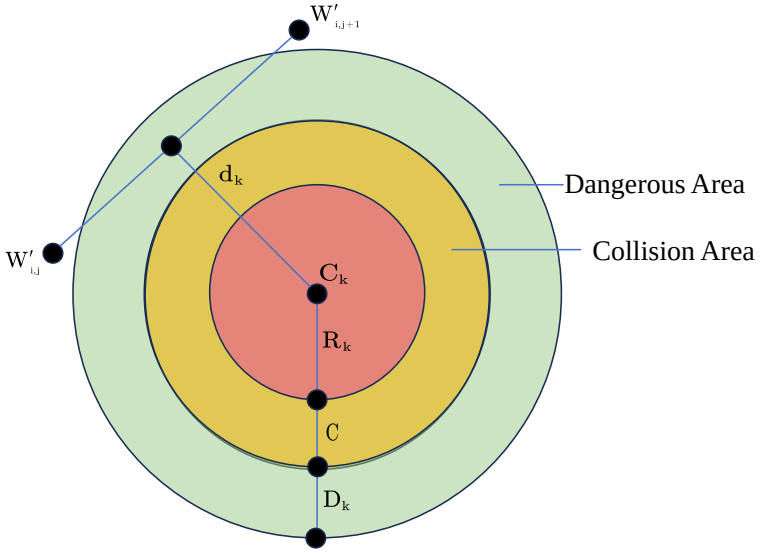

Figure 2. Obstacle hazard area schematic.

3.1.3. Height Constraint

Height constraint is crucial for UAV mission execution and safety. Flying too high increases energy consumption, possibly resulting in mission failure, while flying too low risks colliding with terrain obstacles. To balance these effects, we introduce two constraints: the maximum allowable height, h_{max}, and the minimum allowable height, h_{min}. The height constraint cost is computed by interpolating the current path point, h_{ij}, within the allowable height range as follows:

$$C_2(P_i) = \sum_{j=1}^{n} H_{i,j}, \tag{9}$$

$$H_{ij} = \begin{cases} \left| h_{ij} - \frac{(h_{max}+h_{min})}{2} \right|, & h_{min} \le h_{ij} \le h_{max} \\ +\infty & , \ else. \end{cases} \tag{10}$$

3.1.4. Attitude Angle Constraints

In this section, to ensure a safe and stable UAV flight, we primarily consider dynamic constraints, including bank angle and pitch angle constraints. By calculating the angle between the projections of line segments formed by adjacent path points on the xy-plane, we can determine the yaw angle:

$$\varphi_{ij} = arctan\left(\frac{\| \overrightarrow{W'_{ij}W'_{i,j+1}} \times \overrightarrow{W'_{i,j+1}W'_{i,j+2}} \|}{\overrightarrow{W'_{ij}W'_{i,j+1}} \overrightarrow{W'_{i,j+1}W'_{i,j+2}}} \right), \tag{11}$$

where φ_{ij} denotes the bank angle of the j-th path. Calculating the vertical change in the relative height of path points allows us to determine the pitch angle of the UAV at the j-th path point ψ_{ij} as follows:

$$\psi_{ij} = arctan\left(\frac{z_{i,j+1} - z_{ij}}{\| \overrightarrow{W'_{ij}W'_{i,j+1}} \|}\right). \tag{12}$$

The cost of angle constraints for the unmanned aerial vehicle can then be obtained as:

$$C_4(P_i) = a_1 \sum_{j=1}^{n-2} \varphi_j + a_2 \sum_{j=1}^{n-1} |\psi_{ij} - \psi_{i,j-1}|, \tag{13}$$

where $a_1$and a_2 are the weighting coefficients assigned to the UAV's horizontal bank angle and pitch angle, respectively. Note that in a two-dimensional environment only the bank angle constraint needs to be considered.

3.2. ARO Algorithm

The ARO algorithm [31] draws inspiration from the natural survival strategies of rabbits and primarily comprises two phases: the exploration phase (foraging detour) and the exploitation phase (random hiding). The algorithm dynamically switches between these two stages by adjusting the energy budget. Each stage has a unique updating mechanism, detailed as below.

3.2.1. Population Initialization

Like most algorithms, the ARO algorithm requires initialization of the initial population. Setting the population size as n, the initial position of the i-th rabbit is determined as follows:

$$x_i = low + rand(0,1)(up - low), \quad i = 1, 2, ..., n \tag{14}$$

where x_i represents the position of the i-th rabbit, and low and up, respectively, indicate the lower and upper bounds of the search area.

3.2.2. Detour Foraging (Exploration)

To avoid their burrows being detected by predators, rabbits adopt a detour strategy, foraging randomly in the grasslands near each other's burrows. This strategy can be seen as the exploration behavior of rabbits under a wide field of view, represented by the following mathematical model:

$$\begin{aligned} \vec{x}_c(t+1) = \vec{x}_j(t) + R\left(\vec{x}_i(t) - \vec{x}_j(t)\right) + round(0.5(0.05 + r_1))n_1, \\ i, j = 1, ..., n \text{ and } j \neq i, \end{aligned} \tag{15}$$

$$R = L \cdot c, \tag{16}$$

$$L = \left(e - e^{\left(\frac{t-1}{T}\right)^2}\right) sin(2\pi r_2), \tag{17}$$

$$c(k) = \begin{cases} 1, & k == randperm(dim, l) \\ 0, & else \end{cases} \qquad l = 1, ..., \lceil r_3 \cdot dim \rceil. \tag{18}$$

In Equation (15), $\vec{x}_c(t+1)$ denotes the candidate position of the i-th rabbit at time $t+1$, and R functions as the run operator to simulate the rabbits' running behavior. In Equation (17), T denotes the set maximum number of iterations, and L signifies the step length, indicating the moving speed of rabbits during foraging detours. A longer step length benefits exploration in the algorithm's early phases, whereas it dynamically shortens in later phases to enhance exploitation. In Equation (18), $k \in \{1, \ldots, \text{dim}\}$. And n_1 is a

random variable that follows the standard normal distribution, while r_1, r_2, and r_3 are three random numbers in the range (0, 1).

3.2.3. Random Hiding (Exploitation)

To protect themselves from predators, rabbits use a random hiding strategy to randomly choose burrows for hiding, thereby reducing the risk of capture. The method for generating the j-th burrow of the i-th rabbit is described as follows:

$$\vec{b}_{i,j}(t) = \vec{x}_i(t) + H \cdot g \cdot \vec{x}_i(t),\ j = 1, ..., dim, \tag{19}$$

$$H = \frac{T - t + 1}{T} \cdot r_4, \tag{20}$$

$$g(k) = \begin{cases} 1, & k = j \\ 0, & else, \end{cases} \tag{21}$$

where r_4 is a random number between (0,1), and H is a hiding parameter. As the iteration count, t, increases, the hiding parameter, H, dynamically decreases, facilitating better utilization of the discovered optimal solution during the latter stages.

The generation method of the burrow is given in the above equation. To evade pursuit, rabbits will randomly select a burrow for hiding, and their position update strategy is detailed as follows:

$$\vec{x}_c(t+1) = \vec{x}_i(t) + R\left(r_4 \vec{b}_{i,r}(t) - \vec{x}_i(t)\right), \tag{22}$$

$$g_r(k) = \begin{cases} 1 & k == \lceil r_5 d \rceil \\ 0 & else, \end{cases} \tag{23}$$

$$\vec{b}_{i,r}(t) = \vec{x}_i(t) + H g_r \vec{x}_i(t). \tag{24}$$

In Equation (22), $\vec{b}_{i,r}$ represents the burrow randomly selected from d burrows, and it should be noted that the quantity of burrows is consistent with the dimension of the problem. Therefore, the i-th rabbit can randomly select one of the d available burrows as its candidate position. Also, r_4 and r_5 are random numbers between (0,1). After selecting one of the strategies, either detouring for food or random hiding, the position update formula for the rabbit is outlined as follows:

$$\vec{x}_i(t+1) = \begin{cases} \vec{x}_i(t) & f(\vec{x}_i(t)) \le f(\vec{x}_c(t+1)) \\ \vec{x}_c(t+1) & f(\vec{x}_i(t)) > f(\vec{x}_c(t+1)), \end{cases} \tag{25}$$

this equation describes that if the fitness of the i-th rabbit's candidate position surpasses its current position, the rabbit will adjust its position to the candidate position; otherwise, it will retain its current position.

3.2.4. Energy Shrink

The rabbits are influenced by dynamic energy factors when selecting strategies. Initially, rabbits tend to adopt the detouring foraging strategy, which is beneficial for extensive exploration. As iterations progress, the energy factor dynamically changes, prompting rabbits to gradually shift towards the random hiding strategy, simulating the natural transition from exploration to exploitation.

$$A(t) = 4\left(1 - \frac{t}{T}\right) \ln \frac{1}{r_6}, \tag{26}$$

where T denotes the maximum number of iterations, and r_6 is a random number uniformly distributed between 0 and 1. As the iterations, t, increase, the energy factor, A, gradually diminishes and tends toward zero.

3.3. Douglas–Peucker Algorithm

The Douglas–Peucker algorithm [55] is commonly employed for simplifying curves and reducing a large number of sample points. This algorithm iteratively simplifies complex geometric objects into approximate representations with fewer points, significantly reducing data storage and processing overheads while preserving the fundamental characteristics of the original shape. The fundamental concept of the algorithm is outlined below.

First, set an appropriate distance threshold, D, based on the actual situation. Next, choose the starting point, P1, and the ending point, P10, of the geometric object as the initial simplified line segment, depicted in Figure 3a.

Subsequently, determine the perpendicular distance from all intermediate points to this line segment, as illustrated in Figure 3b. If the perpendicular distance of all points is less than the preset threshold, D, then the line segment is the final simplified result. Otherwise, find the point corresponding to the maximum perpendicular distance as the new splitting point, divide the original geometric object into two segments, and recursively apply the Douglas–Peucker algorithm to these two segments until all sub-segments meet the simplification conditions.

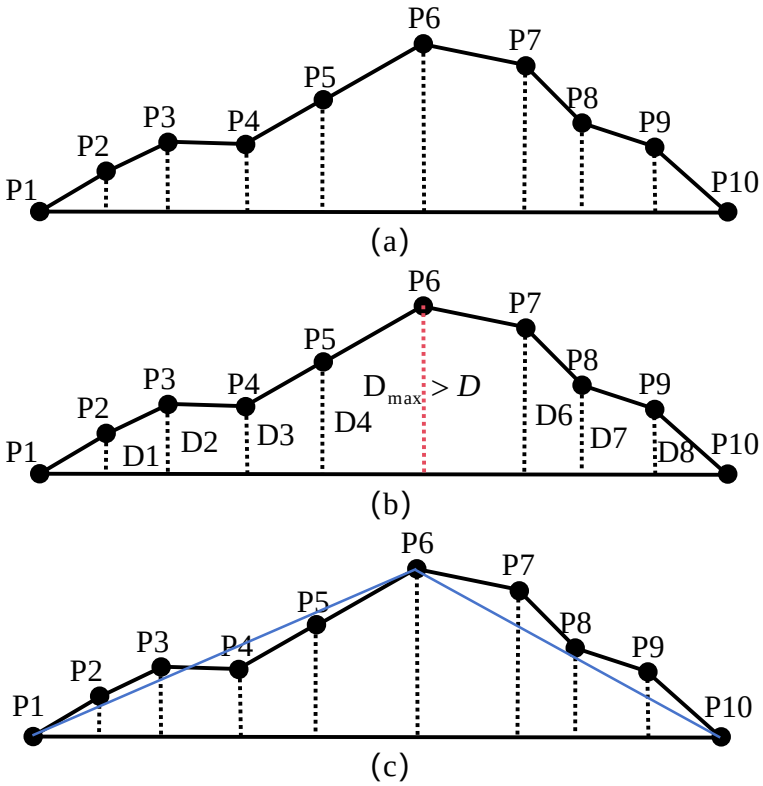

Figure 3. DP algorithm process. (**a**) Initial path representation from points P_1 to P_{10}. (**b**) Calculation of vertical distances D_1 to D_8 between the initial path points P_1 to P_{10} and the baseline, with the red dotted line indicating the maximum distance D_{max} that exceeds the set distance threshold D. (**c**) Simplified path after applying the DP algorithm, with the blue line representing the optimized path.

A geometric shape approximation with fewer key points is finally obtained through continuous iteration of this process, as illustrated in Figure 3c. The blue line segment represents the final simplified sample path.

4. The Proposed Methods

This section introduces the HARO+ algorithm for solving UAV path planning problems in complex environments. HARO+ consists of two parts. Firstly, it improves the basic ARO algorithm, introducing the HARO algorithm. HARO aims to enhance ARO's exploration capability, increase population diversity, and prevent local optima. It includes three improvement strategies: elite-guided exploration, dual exploration switching, and population migration memory. The second part, HARO+'s trajectory optimization, optimizes the initial path from HARO based on varying obstacle density to reduce redundancy and flight cost. The following sections will provide detailed descriptions of these two key components.

4.1. The Proposed HARO

4.1.1. Elite-Guided Exploration Strategy

In the exploration phase of the ARO algorithm, a foraging strategy is adopted, where individuals in the population explore the vicinity of each other's burrows randomly. However, excessively random strategies lack effective individual guidance, leading to slower algorithm convergence and potentially reducing inter-individual communication within the population, affecting overall algorithm performance. To further optimize the foraging strategy, enhance algorithm convergence, and strengthen intra-population information exchange, we adjust the candidate position update strategy for rabbits as follows:

$$\vec{x}_c(t+1) = \vec{x}_j(t) + R(\vec{x}_b(t) - \vec{x}_i(t)) + round(0.5(0.05 + r_1))n_1,\ i, j = 1, ..., n\ and\ j \neq i, \tag{27}$$

where $\vec{x}_b(t)$ represents the present optimal position of the population at the t-th iteration. To expedite the foraging speed of rabbits, we introduce an elite-guided strategy, where elite individuals guide others towards the current optimal position of the population, thereby enhancing overall optimization speed. Essentially, the candidate position update method of the population involves generating new individuals near the optimal individual. In other words, the other members of the population are directed towards the optimal region by the current optimal individual. Additionally, this updating strategy maintains the original algorithm's randomness, safeguarding it against local optima.

4.1.2. Dual Exploration Strategy

The strategy in the exploration phase is crucial for optimizing the entire algorithm. Finding higher-quality solutions in the exploration phase not only benefits subsequent development stages but also provides a variety of options for this phase, ensuring that the algorithm maintains good local development capabilities during subsequent iterations. However, the exploration strategy of the original ARO algorithm is relatively singular, hindering comprehensive exploration across the complete solution space. Inspired by the Crested Porcupine Optimizer [56], we introduce a second exploration strategy in the HARO algorithm. Building on the second defense mechanism of the CPO algorithm, we integrate elite-guided exploration and adjust the original candidate position update formula, proposing the following update scheme:

$$\vec{y}_i(t) = \frac{\vec{x}_i(t) + \vec{x}_j(t) + \vec{x}_b(t)}{3}, i, j = 1, ..., n, \tag{28}$$

$$\vec{x}_c(t+1) = U_r\vec{x}_i(t) + (1 - U_r)\left(\vec{y}_i(t) + rand\left(\vec{x}_{j1}(t) - \vec{x}_{j2}(t)\right)\right), j_1 \neq j_2, \tag{29}$$

$$U_r(k) = \begin{cases} 1, & \tau_1 > \tau_2 \\ & k = 1, ..., dim, \\ 0, & else \end{cases} \tag{30}$$

In Equation (29), U_r is a binary vector that controls whether the original solution is retained or a new exploration strategy is applied during the update process. Each element of the vector, $U_r(k)$, corresponds to the binary value at the k-th dimension and determines the update at that specific dimension. The update rule for $U_r(k)$ is defined in Equation (30), where τ_1 and τ_2 are independently sampled random numbers from the interval (0, 1). These values dictate whether $U_r(k)$ is assigned a value of 1. By adding this exploration strategy, individuals have more methods to explore the solution space, facilitating the exploration of more promising areas.

4.1.3. Population Migration Memory Mechanism

The memory mechanism [57], validated in the Marine Predators algorithm [41], inspired us to propose a population migration memory mechanism. This mechanism can be understood as rabbits migrating with changes in the surrounding food and habitat. As time progresses, each individual in the population will attempt to randomly locate the next candidate position within the nearby area. Using their memory characteristics, they will evaluate whether this candidate's position is better than the current best position. If the candidate's position proves to be superior, the individual will opt to migrate; otherwise, it will remain in its original position. Additionally, we introduce a mutation operation [58], which helps maintain population diversity and improves the algorithm's capacity to avoid local optima. The mathematical expression is as follows:

$$\vec{x}_i(t+1) = \begin{cases} \vec{x}_i(t) + F\left[\vec{x}_{j1}(t) - \vec{x}_{j2}(t)\right] & r \leq 0.2 \\ \vec{x}_i(t) + [0.2(1-r) + r]\left(\vec{x}_{j1}(t) - \vec{x}_{j2}(t)\right) & else, \end{cases} \tag{31}$$

$$F(t) = \frac{1}{2}\left(1 - \frac{t}{T}\right)^{\left(\frac{2t}{T}\right)}, \tag{32}$$

where r represents a random number between 0 and 1. As illustrated in Figure 4, F represents a proportion factor that progressively decreases from 0.5 to 0 as the iterations advance. It decreases slowly in the early iterations, expanding the algorithm's search space, and decreases rapidly in the later iterations, effectively enhancing the algorithm's convergence speed.

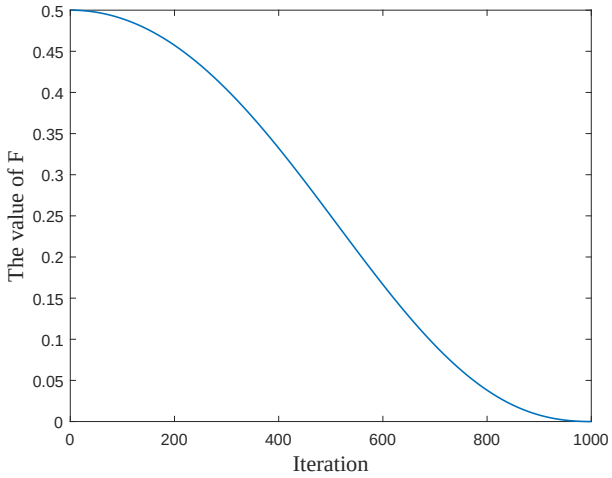

Figure 4. Behavior of H during 1000 iterations.

4.1.4. The Algorithmic Process and Computational Complexity Analysis of HARO

By incorporating the three improvements above into the original ARO algorithm, we introduce the HARO algorithm. Algorithm 1 provides the pseudocode for HARO.

The computational complexity of the HARO algorithm can be determined from its pseudocode, which is divided into the initialization and iteration stages. Its overall computational complexity is:

$$\begin{aligned} O(HARO) = {} & O(initialization) + O(evaluation) \\ & + O(detour\ foraging) \\ & + O(population\ migration) \\ & + O(random\ hiding). \end{aligned} \tag{33}$$

The computational complexity during the initialization stage is $O(n \times d)$, where n represents the population size and d the problem dimension. The iteration stage can be divided into four parts: detour foraging, random hiding, population migration, and fitness function evaluation. The computational complexity of the detour foraging and random hiding stages is $O(\frac{1}{2}Tnd)$, and that of the population migration stage is $O(Tnd)$, where T represents the maximum number of iterations, n denotes the population size, and d indicates the problem dimension.

Algorithm 1 Pseudocode of the HARO algorithm

Input: The maximum iteration count T, the rabbit population size N, and the problem dimensionality D, and some other basic parameters.
Output: The best position X_{best} of the rabbit and its corresponding fitness value F_b

```
Initialize the positions of the rabbits X_i using Equation (14).
while t ≤ T do
    Determine the positions and fitness values of all individuals, and implement the
    memory storage mechanism. Compute the energy factor F as Equation (32).
    for i = 1 to N do
        Calculate the factor A by using Equation (26);
        if A > 1 then
            Select a rabbit at random from other individuals;
            if rand < rand then
                Perform detour foraging by using Equations (15)–(18).
            else
                Perform other exploration strategy by using Equations (28)–(30).
            end if
        else
            Create d burrows and randomly select one for hiding by using Equation (24);
            Execute random hiding in accordance with Equation (22).
        end if
        Update the position of the individual by using Equation (25)
    end for
    Accomplish memory saving, and perform the population migration mechanism by
    using Equations (31)–(32).
end while.
```

4.2. HARO+ Trajectory Optimization

It is essential to highlight that selecting the appropriate threshold is crucial when applying the DP algorithm to UAV trajectory optimization. However, relying solely on the threshold to simplify spatial sample points in the path may lead to neglecting critical points near obstacles, thus posing a risk of collision with obstacles in the simplified path.

As illustrated in Figure 5a, when applying the DP algorithm to simplify the original path in a two-dimensional environment, the threshold strategy cannot account for key points near obstacles, resulting in the planned red path colliding with obstacles. It is also necessary to consider critical points near obstacles in a three-dimensional environment, as shown in Figure 5b. Ensuring safety allows both the maintenance of the original path's characteristics and the reduction of redundant path points, thereby reducing the UAV's energy consumption.

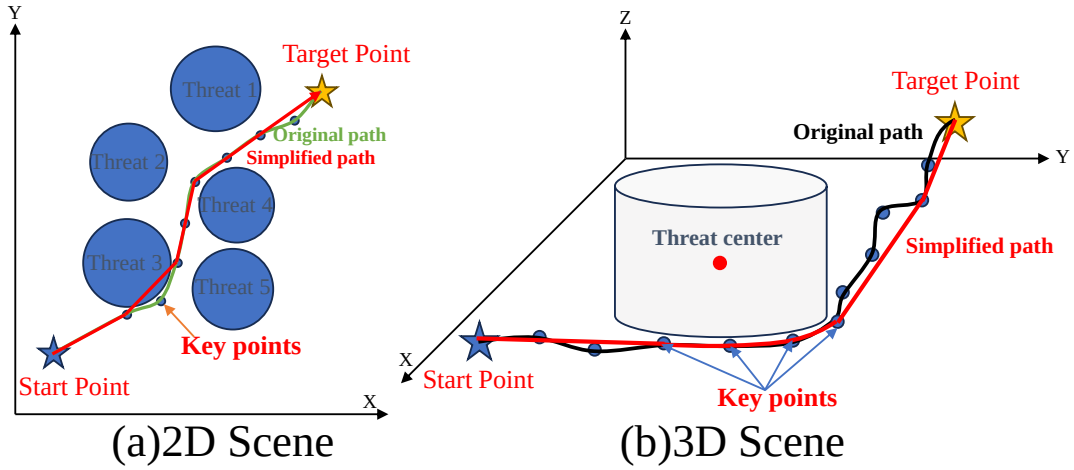

Figure 5. Limitations of the DP algorithm in UAV trajectory optimization ((**a**) 2D environment; (**b**) 3D environment).

Therefore, the method we propose follows these steps:

(1) Considering the UAV safety constraints mentioned earlier, we define path points within the danger zone as key points. Based on the actual obstacle environment, set an appropriate threshold value, D.

(2) Determine the Euclidean distance, d_t, from each point in the flight path, P, to the centers of the obstacles it passes through. If d_t is less than the preset danger zone radius, the path point belongs to a key point and needs to be retained; otherwise, path points outside this area can be ignored. According to the order in which the flight path passes through obstacles, these key points are sequentially stored in set $\mathcal{K}$, and the number of key points is denoted as n,

$$\mathcal{K} = \{k_1, k_2, ..., k_n\}. \tag{34}$$

(3) Utilize the DP algorithm to obtain the simplified set of path points $S_p = \{s_1, s_2, \ldots, s_n\}$. Then, traverse the key point set $\mathcal{K}$, adding the key point to set $\mathcal{S}$ in order if it is not already present; otherwise, leave it unchanged. Set $\mathcal{S}$ represents the final collection of simplified route points containing key points.

Through the aforementioned strategy of retaining key points, we can derive a simplified path that includes these key points. It is worth noting that the approach to computing the distance from a point to a line segment using the DP algorithm varies between two-dimensional and three-dimensional environments. Specifically, we present the computational process for calculating this distance in three-dimensional environments using the DP algorithm, as illustrated in the pseudocode of Algorithm 2.

Algorithm 2 Calculate the distance between a point to a line segment in 3D space using the Douglas–Peucker algorithm

Input: The point p, the starting point a of the line segment, and the ending point b of the line segment.
Output: The distance d_i from the point to the line segment.

Compute the vector of the line segment $v = b - a$
Compute the vector from point p to the starting point a of the line segment $u = p - a$
Calculate the proportion of the point to the line segment projection $t = \frac{u \cdot v}{\|v\|^2}$
if $t \leq 0$ **then**
  // Point p lies in front of the line segment.
  // The shortest distance from p to the starting point a of the line segment is calculated.
  $d_i = \|p - a\|$
else if $t \geq 1$ **then**
  // Point p lies beyond the line segment.
  // The shortest distance from p to the ending point b of the line segment is calculated.
  $d_i = \|p - b\|$
else
  // Point p falls onto the line segment.
  // The coordinates of the projection point p' onto the line segment are calculated.
  Calculate the coordinates of the projection point
    $p' = a + tv$
  // The shortest distance from p to the projection point p' is calculated.
  $d_i = \|p - p'\|$
end if

5. Simulation Experiments and Result Analysis

This section validates the performance improvements of the proposed HARO algorithm through extensive numerical experiments and simulation analyses. Section 5.1 focuses on the numerical results obtained from the CEC2017 benchmark functions, analyzing the improvements in terms of convergence, exploration–exploitation balance, and stability. Section 5.2 applies the HARO algorithm and comparative algorithms to various 2D and 3D UAV path planning scenarios, demonstrating and evaluating its capability to solve complex optimization problems under multiple constraints. Additionally, the effectiveness of the enhanced trajectory optimization strategy, HARO+, is further validated.

5.1. Numerical Experiments and Analysis

To thoroughly assess the performance of the HARO algorithm, this section initially conducts comparative numerical experiments on the classic CEC2017 test function set [59], which includes unimodal, multimodal, hybrid, and composite functions, totaling 29. The comparison is made with various classical, new, and improved optimization algorithms to validate the exceptional performance of HARO in balancing exploration and exploitation capabilities, maintaining population diversity, and optimizing convergence. The comparative algorithms include Particle Swarm Optimization (PSO) [24], Whale Optimization algorithm (WOA) [60], Grey Wolf Optimizer (GWO) [61], African Vultures Optimization algorithm (AVOA) [62], Sparrow Search algorithm (SSA) [63], Dung Beetle Optimizer (DBO) [64], Harris Hawk Optimization algorithm (HHO) [65], Marine Predators algorithm (MPA) [41], artificial rabbits optimization (ARO) [31], and an improved artificial rabbits optimization (IARO) [51].

Furthermore, an analysis of HARO's exploration and exploitation mechanisms is conducted to verify its exceptional ability to balance global exploration and local exploitation. And we employ two non-parametric statistical tests, Wilcoxon's rank-sum test and Friedman's test, to assess the statistical significance of the differences between HARO and other comparison algorithms. These methods are employed to assess the significant performance

disparities of HARO relative to other algorithms from various perspectives, ensuring the credibility and persuasiveness of the results. Finally, through an ablation study of the different improvement strategies introduced in HARO, we confirm the positive impact of these strategies on the performance of the original ARO algorithm.

5.1.1. Operating Environment Setup

To guarantee fair comparative numerical experiments between HARO and other algorithms, we established identical parameters for all participating algorithms: maximum iteration times $T_{max} = 1000$, population size $N = 100$, and dimension $dim = 30$. Each algorithm independently conducted 30 trials on each benchmark function to minimize the influence of randomness. Furthermore, the parameters for each algorithm were configured according to the recommended values in the original literature. However, the number of subpopulations in the DBO algorithm was set according to a certain proportion of subpopulations relative to the total population size, as stated in the original literature. All numerical experiments and simulations were conducted in the "Windows 11 64-bit operating system, Intel i7-12700H 2.30GHz CPU, 16GB memory, Matlab 2021b" environment.

5.1.2. Results and Analysis of CEC2017 Benchmark Functions

To comprehensively evaluate the algorithm's performance, we introduced four evaluation metrics: mean, standard deviation, best value, and ranking. The ranking was obtained by sorting the average values of each algorithm on each function. These indicators quantify and compare the algorithms' optimization ability, convergence, and stability from different perspectives. Table 1 displays the test results of HARO and other comparative algorithms on 30-dimensional CEC2017 benchmark functions. By examining each algorithm's mean, standard deviation, and best value, it is evident that HARO outperforms ARO in all 29 benchmark functions. Specifically, HARO achieves the minimum mean and ranks first in 19 functions; HARO's mean ranks second in the remaining 10 functions. This fully demonstrates that HARO possesses greater competitiveness and superior stability relative to other algorithms.

Table 1. Comparison of results obtained by 11 algorithms on the CEC2017 test suite with 30 dimensions.

Fi	Index	AVOA	DBO	GWO	HHO	MPA	PSO	SSA	WOA	ARO	IARO	HARO
F1	Mean	3.99×10^{3}	2.42×10^{6}	9.86×10^{8}	1.26×10^{7}	4.76×10^{3}	2.40×10^{10}	2.18×10^{3}	7.60×10^{7}	3.84×10^{3}	1.00×10^{2}	1.00×10^{2}
	Std	5.54×10^{3}	5.78×10^{6}	9.65×10^{8}	2.46×10^{6}	3.18×10^{3}	6.16×10^{9}	2.26×10^{3}	6.75×10^{7}	2.86×10^{3}	9.27×10^{-7}	5.21×10^{-4}
	Best	1.16×10^{2}	7.48×10^{2}	8.69×10^{7}	8.84×10^{6}	9.86×10^{2}	1.52×10^{10}	1.23×10^{2}	2.02×10^{7}	2.49×10^{2}	1.00×10^{2}	1.00×10^{2}
	Rank	5	7	10	8	6	11	3	9	4	1	2
F3	Mean	7.26×10^{3}	6.57×10^{4}	3.28×10^{4}	1.17×10^{4}	3.00×10^{2}	5.19×10^{4}	2.61×10^{4}	2.12×10^{5}	2.32×10^{4}	3.00×10^{2}	3.00×10^{2}
	Std	2.67×10^{3}	1.90×10^{4}	1.14×10^{4}	4.82×10^{3}	5.85×10^{-1}	9.16×10^{3}	5.10×10^{3}	5.93×10^{4}	4.98×10^{3}	2.60×10^{-8}	3.57×10^{-3}
	Best	2.49×10^{3}	2.91×10^{4}	9.13×10^{3}	4.06×10^{3}	3.00×10^{2}	3.59×10^{4}	1.89×10^{4}	9.60×10^{4}	1.68×10^{4}	3.00×10^{2}	3.00×10^{2}
	Rank	4	10	8	5	3	9	7	11	6	1	2
F4	Mean	5.08×10^{2}	5.24×10^{2}	5.61×10^{2}	5.33×10^{2}	4.88×10^{2}	4.41×10^{3}	4.95×10^{2}	5.93×10^{2}	5.01×10^{2}	4.15×10^{2}	4.15×10^{2}
	Std	2.52×10^{1}	3.67×10^{1}	4.11×10^{1}	3.22×10^{1}	1.64×10^{0}	1.63×10^{3}	2.93×10^{1}	5.89×10^{1}	2.51×10^{1}	2.44×10^{1}	2.68×10^{1}
	Best	4.68×10^{2}	4.74×10^{2}	4.95×10^{2}	4.81×10^{2}	4.85×10^{2}	1.62×10^{3}	4.01×10^{2}	5.15×10^{2}	4.35×10^{2}	4.00×10^{2}	4.00×10^{2}
	Rank	6	7	9	8	3	11	4	10	5	1	2
F5	Mean	7.00×10^{2}	6.87×10^{2}	5.94×10^{2}	7.31×10^{2}	5.71×10^{2}	8.46×10^{2}	7.39×10^{2}	7.90×10^{2}	5.75×10^{2}	5.80×10^{2}	5.46×10^{2}
	Std	4.41×10^{1}	4.31×10^{1}	1.85×10^{1}	2.06×10^{1}	1.33×10^{1}	3.00×10^{1}	4.71×10^{1}	3.94×10^{1}	2.01×10^{1}	1.62×10^{1}	1.05×10^{1}
	Best	6.35×10^{2}	5.92×10^{2}	5.56×10^{2}	6.85×10^{2}	5.39×10^{2}	7.87×10^{2}	6.52×10^{2}	7.20×10^{2}	5.39×10^{2}	5.43×10^{2}	5.30×10^{2}
	Rank	7	6	5	8	2	11	9	10	3	4	1
F6	Mean	6.42×10^{2}	6.23×10^{2}	6.05×10^{2}	6.60×10^{2}	6.02×10^{2}	6.75×10^{2}	6.45×10^{2}	6.71×10^{2}	6.00×10^{2}	6.00×10^{2}	6.00×10^{2}
	Std	8.67×10^{0}	1.14×10^{1}	2.80×10^{0}	5.58×10^{0}	1.13×10^{0}	3.84×10^{0}	1.07×10^{1}	9.99×10^{0}	2.79×10^{-1}	3.53×10^{-1}	1.44×10^{-3}
	Best	6.23×10^{2}	6.04×10^{2}	6.02×10^{2}	6.43×10^{2}	6.00×10^{2}	6.67×10^{2}	6.26×10^{2}	6.47×10^{2}	6.00×10^{2}	6.00×10^{2}	6.00×10^{2}
	Rank	7	6	5	9	4	11	8	10	2	3	1
F7	Mean	1.11×10^{3}	8.99×10^{2}	8.46×10^{2}	1.23×10^{3}	8.09×10^{2}	1.30×10^{3}	1.24×10^{3}	1.26×10^{3}	8.15×10^{2}	8.20×10^{2}	7.77×10^{2}
	Std	9.28×10^{1}	5.80×10^{1}	4.38×10^{1}	7.65×10^{1}	1.80×10^{1}	6.35×10^{1}	1.06×10^{2}	1.15×10^{2}	1.65×10^{1}	2.21×10^{1}	1.23×10^{1}
	Best	9.32×10^{2}	7.95×10^{2}	7.81×10^{2}	1.06×10^{3}	7.69×10^{2}	1.12×10^{3}	9.62×10^{2}	1.02×10^{3}	7.80×10^{2}	7.83×10^{2}	7.48×10^{2}
	Rank	7	6	5	8	2	11	9	10	3	4	1
F8	Mean	9.63×10^{2}	9.96×10^{2}	8.78×10^{2}	9.67×10^{2}	8.74×10^{2}	1.04×10^{3}	9.83×10^{2}	1.02×10^{3}	8.79×10^{2}	8.74×10^{2}	8.49×10^{2}
	Std	2.88×10^{1}	5.25×10^{1}	1.54×10^{1}	2.39×10^{1}	1.20×10^{1}	2.00×10^{1}	2.21×10^{1}	5.50×10^{1}	1.92×10^{1}	1.35×10^{1}	1.06×10^{1}
	Best	9.12×10^{2}	8.77×10^{2}	8.45×10^{2}	9.13×10^{2}	8.57×10^{2}	1.00×10^{3}	9.33×10^{2}	9.31×10^{2}	8.50×10^{2}	8.48×10^{2}	8.30×10^{2}
	Rank	6	9	4	7	2	11	8	10	5	3	1

Table 1. *Cont.*

Fi	Index	AVOA	DBO	GWO	HHO	MPA	PSO	SSA	WOA	ARO	IARO	HARO
F9	Mean	5.25×10^{3}	6.06×10^{3}	1.48×10^{3}	6.44×10^{3}	9.83×10^{2}	6.47×10^{3}	5.35×10^{3}	7.96×10^{3}	1.11×10^{3}	1.16×10^{3}	9.02×10^{2}
	Std	8.68×10^{2}	2.01×10^{3}	3.23×10^{2}	7.19×10^{2}	5.48×10^{1}	5.29×10^{2}	1.34×10^{2}	2.65×10^{3}	1.95×10^{2}	2.28×10^{2}	2.56×10^{0}
	Best	3.33×10^{3}	2.23×10^{3}	1.04×10^{3}	5.25×10^{3}	9.14×10^{2}	5.62×10^{3}	5.02×10^{3}	4.08×10^{3}	9.10×10^{2}	9.16×10^{2}	9.00×10^{2}
	Rank	6	8	5	9	2	10	7	11	3	4	1
F10	Mean	5.10×10^{3}	5.33×10^{3}	4.01×10^{3}	5.45×10^{3}	3.70×10^{3}	6.73×10^{3}	5.40×10^{3}	6.36×10^{3}	4.03×10^{3}	3.75×10^{3}	3.27×10^{3}
	Std	6.44×10^{2}	6.62×10^{2}	5.16×10^{2}	7.20×10^{2}	4.33×10^{2}	6.59×10^{2}	7.90×10^{2}	8.19×10^{2}	5.47×10^{2}	5.83×10^{2}	4.64×10^{2}
	Best	3.98×10^{3}	4.23×10^{3}	2.54×10^{3}	4.02×10^{3}	3.01×10^{3}	5.06×10^{3}	3.86×10^{3}	4.53×10^{3}	2.29×10^{3}	2.48×10^{3}	2.31×10^{3}
	Rank	6	7	4	9	2	11	8	10	5	3	1
F11	Mean	1.24×10^{3}	1.50×10^{3}	1.48×10^{3}	1.26×10^{3}	1.15×10^{3}	3.57×10^{3}	1.25×10^{3}	3.10×10^{3}	1.18×10^{3}	1.20×10^{3}	1.14×10^{3}
	Std	5.01×10^{1}	1.28×10^{2}	5.25×10^{2}	4.06×10^{1}	2.12×10^{1}	1.18×10^{3}	4.07×10^{1}	1.46×10^{3}	3.77×10^{1}	4.57×10^{1}	1.56×10^{1}
	Best	1.16×10^{3}	1.27×10^{3}	1.24×10^{3}	1.20×10^{3}	1.12×10^{3}	2.12×10^{3}	1.19×10^{3}	1.55×10^{3}	1.13×10^{3}	1.12×10^{3}	1.11×10^{3}
	Rank	5	9	8	7	2	11	6	10	3	4	1
F12	Mean	2.28×10^{6}	2.10×10^{7}	4.27×10^{7}	1.12×10^{7}	1.17×10^{4}	2.75×10^{9}	7.72×10^{5}	7.42×10^{7}	5.00×10^{5}	1.37×10^{4}	9.02×10^{3}
	Std	1.33×10^{6}	3.97×10^{7}	6.84×10^{7}	1.03×10^{7}	5.83×10^{3}	1.75×10^{9}	6.41×10^{5}	5.76×10^{7}	3.73×10^{5}	9.21×10^{3}	5.79×10^{3}
	Best	6.12×10^{5}	3.34×10^{5}	2.29×10^{6}	2.80×10^{6}	3.70×10^{3}	6.07×10^{8}	1.52×10^{5}	1.22×10^{7}	8.90×10^{4}	1.51×10^{3}	2.31×10^{3}
	Rank	6	8	9	7	2	11	5	10	4	3	1
F13	Mean	6.99×10^{4}	1.65×10^{6}	1.91×10^{7}	4.70×10^{5}	1.55×10^{3}	9.51×10^{7}	1.97×10^{4}	2.09×10^{5}	1.44×10^{4}	1.61×10^{3}	1.35×10^{3}
	Std	3.92×10^{4}	3.16×10^{6}	6.91×10^{7}	5.59×10^{5}	4.81×10^{1}	1.69×10^{8}	1.88×10^{4}	1.93×10^{5}	1.08×10^{4}	1.92×10^{2}	3.14×10^{1}
	Best	1.21×10^{4}	1.31×10^{4}	3.62×10^{4}	1.14×10^{5}	1.47×10^{3}	8.83×10^{4}	1.53×10^{3}	6.55×10^{4}	1.46×10^{3}	1.39×10^{3}	1.32×10^{3}
	Rank	6	9	10	8	2	11	5	7	4	3	1
F14	Mean	3.59×10^{4}	7.10×10^{4}	2.79×10^{5}	8.30×10^{4}	1.44×10^{3}	6.37×10^{3}	3.56×10^{4}	1.89×10^{6}	3.51×10^{3}	1.50×10^{3}	1.42×10^{3}
	Std	3.21×10^{4}	1.02×10^{5}	3.81×10^{5}	7.76×10^{4}	6.61×10^{0}	5.36×10^{3}	2.63×10^{4}	1.64×10^{6}	2.55×10^{3}	3.57×10^{1}	1.01×10^{1}
	Best	2.50×10^{3}	2.98×10^{3}	1.97×10^{3}	1.23×10^{4}	1.43×10^{3}	1.70×10^{3}	2.31×10^{3}	3.50×10^{4}	1.49×10^{3}	1.44×10^{3}	1.41×10^{3}
	Rank	7	8	10	9	2	5	6	11	4	3	1
F15	Mean	1.69×10^{4}	7.14×10^{4}	1.25×10^{5}	6.94×10^{4}	1.55×10^{3}	1.30×10^{4}	1.12×10^{4}	1.10×10^{5}	5.31×10^{3}	1.67×10^{3}	1.51×10^{3}
	Std	1.07×10^{4}	6.79×10^{4}	2.47×10^{5}	4.89×10^{4}	1.05×10^{1}	5.05×10^{3}	1.22×10^{4}	9.64×10^{4}	4.44×10^{3}	2.86×10^{2}	5.12×10^{0}
	Best	5.08×10^{3}	9.78×10^{3}	1.13×10^{4}	2.49×10^{4}	1.53×10^{3}	3.00×10^{3}	1.61×10^{3}	2.21×10^{4}	1.59×10^{3}	1.53×10^{3}	1.50×10^{3}
	Rank	7	9	11	8	2	6	5	10	4	3	1
F16	Mean	2.84×10^{3}	2.90×10^{3}	2.39×10^{3}	3.35×10^{3}	2.13×10^{3}	4.43×10^{3}	2.98×10^{3}	3.55×10^{3}	2.45×10^{3}	2.33×10^{3}	2.19×10^{3}
	Std	3.55×10^{2}	3.53×10^{2}	2.60×10^{2}	3.92×10^{2}	1.73×10^{2}	5.78×10^{2}	3.10×10^{2}	4.55×10^{2}	2.48×10^{2}	2.48×10^{2}	1.70×10^{2}
	Best	2.31×10^{3}	2.22×10^{3}	1.80×10^{3}	2.63×10^{3}	1.88×10^{3}	3.62×10^{3}	2.35×10^{3}	2.82×10^{3}	2.03×10^{3}	1.91×10^{3}	1.74×10^{3}
	Rank	6	7	4	9	1	11	8	10	5	3	2
F17	Mean	2.39×10^{3}	2.38×10^{3}	1.95×10^{3}	2.61×10^{3}	1.80×10^{3}	2.99×10^{3}	2.38×10^{3}	2.54×10^{3}	1.95×10^{3}	1.93×10^{3}	1.75×10^{3}
	Std	2.47×10^{2}	2.64×10^{2}	1.13×10^{2}	3.35×10^{2}	4.23×10^{1}	2.87×10^{2}	2.28×10^{2}	2.58×10^{2}	1.57×10^{2}	1.17×10^{2}	5.23×10^{1}
	Best	1.94×10^{3}	1.90×10^{3}	1.77×10^{3}	2.03×10^{3}	1.75×10^{3}	2.33×10^{3}	1.87×10^{3}	1.94×10^{3}	1.74×10^{3}	1.75×10^{3}	1.70×10^{3}
	Rank	8	6	4	10	2	11	7	9	5	3	1
F18	Mean	9.02×10^{5}	1.72×10^{6}	6.32×10^{5}	1.15×10^{6}	1.85×10^{3}	1.99×10^{5}	3.84×10^{5}	3.96×10^{6}	9.69×10^{4}	3.57×10^{3}	1.83×10^{3}
	Std	8.62×10^{5}	4.41×10^{6}	5.65×10^{5}	1.32×10^{6}	8.75×10^{0}	2.96×10^{5}	3.51×10^{5}	3.50×10^{6}	7.48×10^{4}	1.48×10^{3}	6.50×10^{0}
	Best	1.17×10^{5}	2.02×10^{4}	7.70×10^{4}	1.07×10^{5}	1.84×10^{3}	2.50×10^{4}	3.19×10^{4}	1.44×10^{5}	3.26×10^{4}	2.04×10^{3}	1.81×10^{3}
	Rank	8	10	7	9	2	5	6	11	4	3	1
F19	Mean	1.71×10^{4}	3.83×10^{5}	8.14×10^{5}	3.95×10^{5}	1.93×10^{3}	2.75×10^{5}	8.63×10^{3}	5.87×10^{6}	6.52×10^{3}	1.94×10^{3}	1.91×10^{3}
	Std	1.50×10^{4}	7.72×10^{5}	1.29×10^{6}	2.87×10^{5}	4.70×10^{0}	3.70×10^{5}	8.96×10^{3}	4.75×10^{6}	4.52×10^{3}	1.91×10^{1}	3.12×10^{0}
	Best	2.48×10^{3}	2.17×10^{3}	2.27×10^{4}	4.23×10^{4}	1.92×10^{3}	4.48×10^{3}	1.96×10^{3}	2.16×10^{4}	2.06×10^{3}	1.92×10^{3}	1.90×10^{3}
	Rank	6	8	10	9	2	7	5	11	4	3	1
F20	Mean	2.64×10^{3}	2.59×10^{3}	2.36×10^{3}	2.73×10^{3}	2.17×10^{3}	3.04×10^{3}	2.68×10^{3}	2.85×10^{3}	2.30×10^{3}	2.23×10^{3}	2.11×10^{3}
	Std	2.10×10^{2}	1.81×10^{2}	1.30×10^{2}	2.03×10^{2}	6.97×10^{1}	3.14×10^{2}	1.73×10^{2}	2.08×10^{2}	1.73×10^{2}	1.18×10^{2}	8.96×10^{1}
	Best	2.19×10^{3}	2.18×10^{3}	2.13×10^{3}	2.34×10^{3}	2.06×10^{3}	2.30×10^{3}	2.38×10^{3}	2.29×10^{3}	2.04×10^{3}	2.04×10^{3}	2.01×10^{3}
	Rank	7	6	5	9	2	11	8	10	4	3	1
F21	Mean	2.49×10^{3}	2.49×10^{3}	2.37×10^{3}	2.54×10^{3}	2.29×10^{3}	2.68×10^{3}	2.50×10^{3}	2.57×10^{3}	2.37×10^{3}	2.37×10^{3}	2.34×10^{3}
	Std	4.78×10^{1}	3.46×10^{1}	2.59×10^{1}	3.82×10^{1}	7.95×10^{1}	6.97×10^{1}	4.46×10^{1}	4.67×10^{1}	2.21×10^{1}	1.89×10^{1}	1.07×10^{1}
	Best	2.43×10^{3}	2.43×10^{3}	2.33×10^{3}	2.46×10^{3}	2.20×10^{3}	2.56×10^{3}	2.43×10^{3}	2.49×10^{3}	2.33×10^{3}	2.34×10^{3}	2.32×10^{3}
	Rank	6	7	5	9	1	11	8	10	4	3	2
F22	Mean	5.08×10^{3}	5.23×10^{3}	4.31×10^{3}	5.84×10^{3}	2.30×10^{3}	8.63×10^{3}	5.15×10^{3}	6.57×10^{3}	2.30×10^{3}	2.30×10^{3}	2.30×10^{3}
	Std	2.23×10^{3}	2.04×10^{3}	1.83×10^{3}	2.23×10^{3}	1.88×10^{0}	4.83×10^{2}	2.42×10^{3}	2.41×10^{3}	1.21×10^{0}	1.68×10^{0}	9.50×10^{-1}
	Best	2.30×10^{3}	2.30×10^{3}	2.36×10^{3}	2.32×10^{3}	2.30×10^{3}	7.00×10^{3}	2.30×10^{3}	2.33×10^{3}	2.30×10^{3}	2.30×10^{3}	2.30×10^{3}
	Rank	6	8	5	9	4	11	7	10	2	3	1
F23	Mean	2.92×10^{3}	2.87×10^{3}	2.74×10^{3}	3.11×10^{3}	2.69×10^{3}	4.56×10^{3}	2.91×10^{3}	3.06×10^{3}	2.73×10^{3}	2.74×10^{3}	2.70×10^{3}
	Std	7.86×10^{1}	6.24×10^{1}	4.35×10^{1}	1.07×10^{2}	4.94×10^{1}	8.03×10^{2}	8.44×10^{1}	7.09×10^{1}	2.19×10^{1}	2.61×10^{1}	1.46×10^{1}
	Best	2.81×10^{3}	2.78×10^{3}	2.68×10^{3}	2.91×10^{3}	2.46×10^{3}	3.30×10^{3}	2.78×10^{3}	2.92×10^{3}	2.70×10^{3}	2.69×10^{3}	2.67×10^{3}
	Rank	8	6	5	10	1	11	7	9	3	4	2
F24	Mean	3.12×10^{3}	3.04×10^{3}	2.91×10^{3}	3.38×10^{3}	2.86×10^{3}	3.54×10^{3}	3.12×10^{3}	3.20×10^{3}	2.89×10^{3}	2.92×10^{3}	2.86×10^{3}
	Std	1.00×10^{2}	5.34×10^{1}	4.84×10^{1}	1.29×10^{2}	8.71×10^{1}	1.37×10^{2}	1.04×10^{2}	9.84×10^{1}	1.93×10^{1}	4.19×10^{1}	1.46×10^{1}
	Best	2.92×10^{3}	2.92×10^{3}	2.86×10^{3}	3.17×10^{3}	2.50×10^{3}	3.32×10^{3}	2.95×10^{3}	2.99×10^{3}	2.86×10^{3}	2.86×10^{3}	2.82×10^{3}
	Rank	8	6	4	10	2	11	7	9	3	5	1

Table 1. *Cont.*

Fi	Index	AVOA	DBO	GWO	HHO	MPA	PSO	SSA	WOA	ARO	IARO	HARO
F25	Mean	2.90×10^3	2.93×10^3	2.96×10^3	2.92×10^3	2.89×10^3	3.72×10^3	2.89×10^3	3.00×10^3	2.91×10^3	2.89×10^3	2.89×10^3
	Std	2.19×10^1	3.79×10^1	2.82×10^1	1.78×10^1	1.76×10^0	3.18×10^2	1.19×10^1	3.95×10^1	1.97×10^1	9.48×10^0	1.61×10^0
	Best	2.88×10^3	2.89×10^3	2.92×10^3	2.89×10^3	2.88×10^3	3.22×10^3	2.88×10^3	2.92×10^3	2.88×10^3	2.88×10^3	2.88×10^3
	Rank	5	8	9	7	1	11	4	10	6	3	2
F26	Mean	6.24×10^3	5.97×10^3	4.44×10^3	7.00×10^3	2.90×10^3	9.26×10^3	6.35×10^3	7.25×10^3	4.34×10^3	4.66×10^3	3.81×10^3
	Std	1.33×10^3	7.60×10^2	2.84×10^2	1.46×10^3	7.53×10^{-2}	9.25×10^2	1.29×10^3	1.33×10^3	9.32×10^2	9.27×10^2	5.87×10^2
	Best	2.80×10^3	3.77×10^3	4.06×10^3	2.95×10^3	2.90×10^3	7.25×10^3	2.80×10^3	2.96×10^3	2.80×10^3	2.80×10^3	2.80×10^3
	Rank	7	6	4	9	1	11	8	10	3	5	2
F27	Mean	3.26×10^3	3.26×10^3	3.24×10^3	3.38×10^3	3.20×10^3	5.07×10^3	3.27×10^3	3.40×10^3	3.23×10^3	3.25×10^3	3.21×10^3
	Std	2.92×10^1	2.94×10^1	1.80×10^1	1.08×10^2	6.77×10^0	1.62×10^3	3.71×10^1	9.43×10^1	1.36×10^1	2.60×10^1	8.77×10^0
	Best	3.22×10^3	3.21×10^3	3.21×10^3	3.25×10^3	3.19×10^3	3.20×10^3	3.22×10^3	3.26×10^3	3.21×10^3	3.21×10^3	3.20×10^3
	Rank	7	6	4	9	1	11	8	10	3	5	2
F28	Mean	3.22×10^3	3.35×10^3	3.37×10^3	3.27×10^3	3.20×10^3	4.74×10^3	3.22×10^3	3.37×10^3	3.25×10^3	3.12×10^3	3.13×10^3
	Std	2.03×10^1	8.09×10^1	5.79×10^1	2.37×10^1	1.15×10^1	3.85×10^2	3.29×10^1	4.56×10^1	2.57×10^1	4.42×10^1	4.95×10^1
	Best	3.20×10^3	3.22×10^3	3.26×10^3	3.21×10^3	3.18×10^3	4.11×10^3	3.19×10^3	3.31×10^3	3.21×10^3	3.10×10^3	3.10×10^3
	Rank	5	8	9	7	3	11	4	10	6	1	2
F29	Mean	4.14×10^3	3.98×10^3	3.74×10^3	4.45×10^3	3.50×10^3	6.58×10^3	4.16×10^3	5.00×10^3	3.62×10^3	3.60×10^3	3.40×10^3
	Std	2.50×10^2	2.62×10^2	1.17×10^2	4.04×10^2	8.05×10^1	9.61×10^2	2.52×10^2	4.27×10^2	1.57×10^2	1.48×10^2	6.86×10^1
	Best	3.60×10^3	3.55×10^3	3.52×10^3	3.75×10^3	3.31×10^3	4.51×10^3	3.62×10^3	4.22×10^3	3.38×10^3	3.42×10^3	3.34×10^3
	Rank	7	6	5	9	2	11	8	10	4	3	1
F30	Mean	8.20×10^4	6.94×10^5	7.09×10^6	1.88×10^6	6.59×10^3	2.65×10^7	1.28×10^4	1.52×10^7	9.05×10^3	6.73×10^3	5.29×10^3
	Std	5.28×10^4	1.41×10^6	5.84×10^6	1.16×10^6	8.42×10^2	3.62×10^7	5.68×10^3	1.20×10^7	1.98×10^3	1.66×10^3	1.60×10^2
	Best	2.09×10^4	1.86×10^4	1.07×10^6	6.10×10^5	5.62×10^3	1.35×10^6	5.74×10^3	2.43×10^6	6.32×10^3	5.37×10^3	5.06×10^3
	Rank	6	7	9	8	2	11	5	10	4	3	1

Furthermore, to visually demonstrate the excellent performance of the HARO algorithm, Figure 6 shows the convergence curves of each algorithm on the CEC2017 test functions. The red curve in the figure represents HARO. On most benchmark functions, HARO's fitness value rapidly converges as the number of iterations increases, reaching a lower fitness value at the maximum number of iterations and demonstrating a significantly better convergence effect than other comparative algorithms. The above data analysis and visualization results consistently confirm the superior performance of the HARO algorithm. By leveraging the advantages of mixing multiple strategies, HARO performs excellently in global search ability and fully explores local development space, ultimately achieving excellent optimality and convergence speed on a wide range of test functions, demonstrating its enormous potential as a new optimization algorithm.

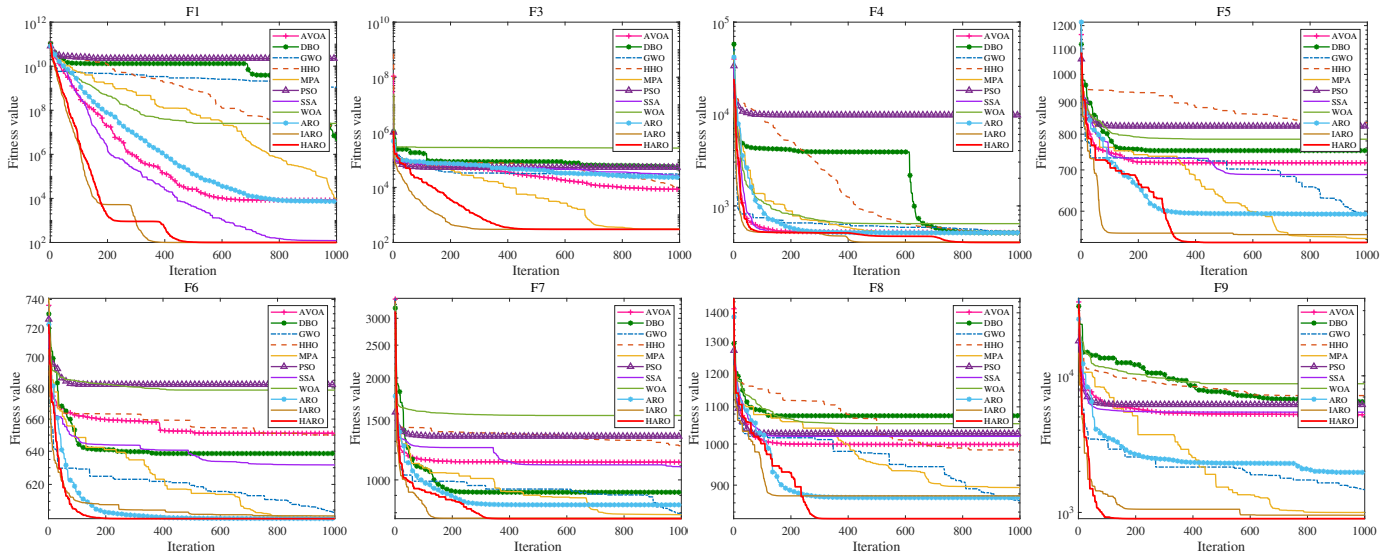

Figure 6. *Cont.*

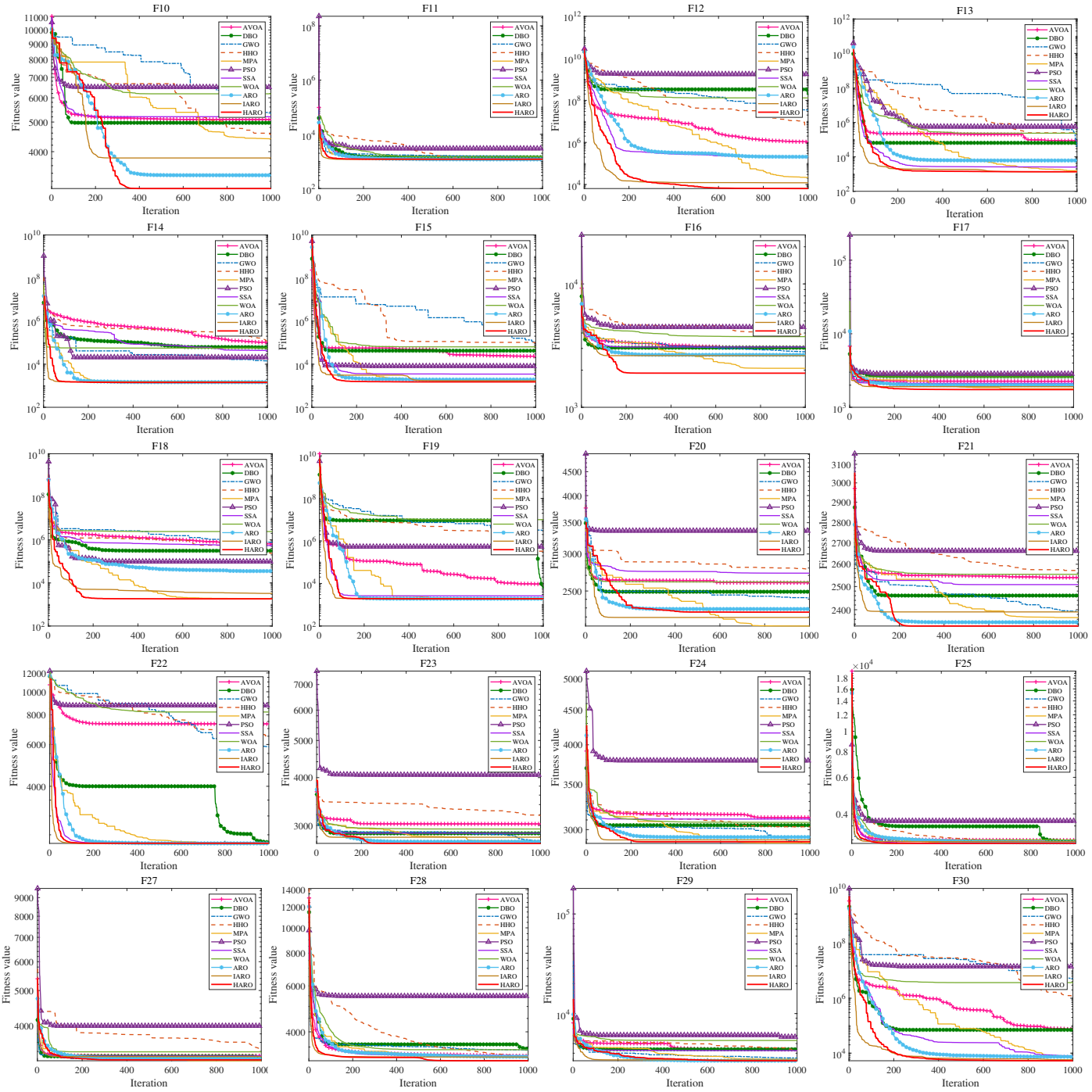

Figure 6. Convergence graphs attained by 11 algorithms on CEC2017 benchmark problems with 30 dim.

5.1.3. Exploration–Exploitation Analysis

The effectiveness of optimization algorithms relies not only on finding the optimal solution but also on sustaining adequate exploration and exploitation abilities throughout the optimization process. Assessing the mean, optimal value, and standard deviation of solutions does not entirely capture the population's search behavior. Therefore, this section employs a method of measuring dimensionality diversity [66] to more clearly observe the HARO algorithm's exploration and exploitation abilities during iterative cycles. The corresponding formula is as follows:

$$Diversity_j = \frac{1}{N}\sum_{i=1}^{N} median\left(x^j\right) - x_i^j, \tag{35}$$

$$Diversity = \frac{1}{D}\sum_{j=1}^{D} Diversity_j, \tag{36}$$

where N represents the population size, $median\left(x^j\right)$ represents the median of all individuals in the population for the j-th dimensional variable, x_i^j represents the value of individual i in dimension j of the population, and D represents the variable dimension. By calculating the average $Diversity_j$ of all individuals in the population and then computing the average diversity of each dimension to obtain $Diversity$, the proportion of exploration and exploitation stages of the population in each iteration can be determined as follows:

$$Exploration\% = \frac{Diversity}{Diversity_{max}} \tag{37}$$

$$Exploitation\% = \frac{\mid Diversity - Diversity_{max} \mid}{Diversity_{max}} \tag{38}$$

We selected a subset of unimodal, multimodal, hybrid, and composite functions from the CEC2017 test function set to analyze exploration and exploitation. Figure 7 illustrates the curves showing the variation in Exploration% and Exploitation% over iterations. We observe that the transition points between the exploration and exploitation stages are inconsistent across different test functions. However, they generally exhibit a clear trend. During the initial iterations, the algorithm emphasizes the exploration phase, conducting a broad global search to help the search agent navigate the solution space quickly and identify potential high-quality solution areas. As iterations advance, the algorithm gradually shifts from the exploration to the exploitation stage, relocating the search focus from global to local regions near suitable solution areas. Within these regions, it conducts local development, continually optimizing and refining the current optimal solution, aiming to discover superior solutions. This transition process demonstrates the algorithm's dynamic equilibrium between global exploration and local exploitation, enabling it to balance breadth and depth search throughout the optimization process. It effectively discovers and utilizes promising solution areas in the solution space, ultimately achieving high-quality optimal solutions. This behavior aligns with the general strategy of efficient optimization algorithms and further validates the rationality and superiority of our algorithm design.

5.1.4. Non-Parametric Statistical Analysis

To objectively assess the performance differences between the proposed HARO algorithm and other algorithms on the 30-dimensional CEC2017 test function set, two widely used non-parametric statistical test methods are employed: the Wilcoxon rank-sum test [67] and the Friedman test [68].

The Wilcoxon rank-sum test assesses whether there is a significant difference between the HARO algorithm and the comparison algorithm based on the p-value at a confidence level of 0.05. If $p \geq 0.05$, the null hypothesis is accepted, indicating no significant difference between the two; otherwise, the null hypothesis is rejected, indicating a significant difference between them. Table 2 displays the p-value differences of each algorithm under the CEC2017 test functions at a confidence level of 0.05. The results indicate that the Wilcoxon test results for the HARO algorithm on most benchmark functions are less than 0.05, demonstrating significant differences compared with other algorithms.

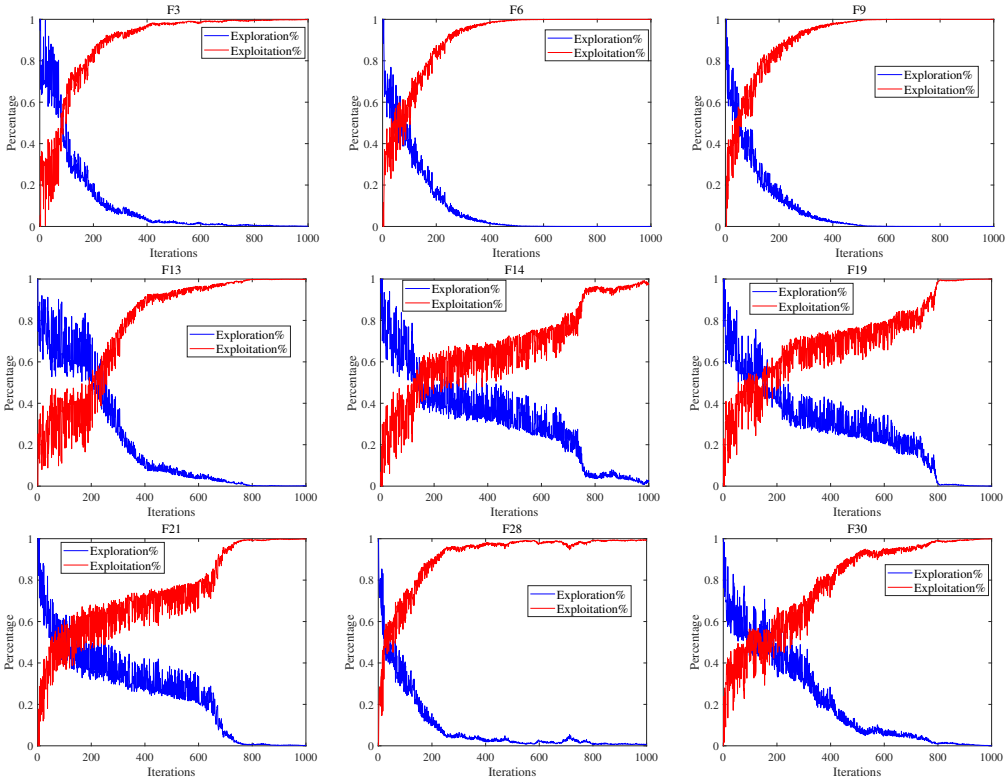

Figure 7. Exploration% and Exploitation% Curves of HARO on CEC2017 benchmark functions.

Table 2. Wilcoxon rank-sum test results of HARO and 10 comparison algorithms on 30-dimensional CEC2017 functions.

Function	AVOA	DBO	GWO	HHO	A, M.P.	PSO	SSA	WOA	ARO	IARO
F1	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}	3.02×10^{-11}	3.02×10^{-11}	4.08×10^{-11}
F3	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}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
F4	7.39×10^{-11}	3.34×10^{-11}	3.02×10^{-11}	3.69×10^{-11}	1.96×10^{-10}	3.02×10^{-11}	1.07×10^{-9}	3.02×10^{-11}	1.46×10^{-10}	7.51×10^{-1}
F5	3.02×10^{-11}	3.02×10^{-11}	6.07×10^{-11}	3.02×10^{-11}	5.46×10^{-9}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	8.35×10^{-8}	3.82×10^{-10}
F6	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}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
F7	3.02×10^{-11}	3.69×10^{-11}	1.09×10^{-10}	3.02×10^{-11}	2.60×10^{-8}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.47×10^{-10}	4.20×10^{-10}
F8	3.02×10^{-11}	3.02×10^{-11}	1.10×10^{-8}	3.02×10^{-11}	2.67×10^{-9}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	4.18×10^{-9}	7.77×10^{-9}
F9	3.00×10^{-11}	3.00×10^{-11}	3.00×10^{-11}	3.00×10^{-11}	3.00×10^{-11}	3.00×10^{-11}	3.00×10^{-11}	3.00×10^{-11}	3.31×10^{-11}	3.00×10^{-11}
F10	3.69×10^{-11}	3.02×10^{-11}	2.49×10^{-6}	3.69×10^{-11}	1.77×10^{-3}	3.02×10^{-11}	3.69×10^{-11}	3.02×10^{-11}	8.20×10^{-7}	1.77×10^{-3}
F11	6.07×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	4.36×10^{-2}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	8.20×10^{-7}	2.39×10^{-8}
F12	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	4.21×10^{-2}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.51×10^{-2}
F13	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}	3.02×10^{-11}	3.02×10^{-11}	6.70×10^{-11}
F14	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	4.44×10^{-7}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
F15	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}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
F16	6.72×10^{-10}	2.61×10^{-10}	6.20×10^{-4}	3.02×10^{-11}	1.81×10^{-1}	3.02×10^{-11}	9.92×10^{-11}	3.02×10^{-11}	2.13×10^{-5}	3.15×10^{-2}
F17	3.34×10^{-11}	3.34×10^{-11}	6.12×10^{-10}	3.02×10^{-11}	1.73×10^{-7}	3.02×10^{-11}	3.69×10^{-11}	3.34×10^{-11}	9.26×10^{-9}	1.69×10^{-9}
F18	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}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
F19	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}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
F20	4.50×10^{-11}	4.98×10^{-11}	2.23×10^{-9}	3.34×10^{-11}	2.76×10^{-3}	3.34×10^{-11}	3.02×10^{-11}	3.34×10^{-11}	1.61×10^{-6}	2.49×10^{-6}
F21	3.02×10^{-11}	3.02×10^{-11}	1.47×10^{-7}	3.02×10^{-11}	4.04×10^{-1}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	1.47×10^{-7}	3.09×10^{-6}
F22	4.60×10^{-10}	7.36×10^{-11}	3.01×10^{-11}	3.01×10^{-11}	2.82×10^{-8}	3.01×10^{-11}	7.35×10^{-10}	3.01×10^{-11}	4.30×10^{-8}	4.59×10^{-1}
F23	3.02×10^{-11}	3.02×10^{-11}	4.31×10^{-8}	3.02×10^{-11}	7.51×10^{-1}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.08×10^{-8}	3.50×10^{-9}
F24	3.02×10^{-11}	3.02×10^{-11}	5.97×10^{-9}	3.02×10^{-11}	8.15×10^{-5}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	1.56×10^{-8}	1.41×10^{-9}
F25	1.11×10^{-4}	5.49×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	1.09×10^{-1}	3.02×10^{-11}	1.44×10^{-3}	3.02×10^{-11}	3.35×10^{-8}	3.59×10^{-5}
F26	3.96×10^{-8}	2.61×10^{-10}	1.73×10^{-7}	1.43×10^{-8}	1.95×10^{-3}	3.02×10^{-11}	3.50×10^{-9}	6.72×10^{-10}	1.32×10^{-4}	2.96×10^{-5}
F27	6.70×10^{-11}	3.47×10^{-10}	1.55×10^{-9}	3.02×10^{-11}	3.56×10^{-4}	1.17×10^{-4}	5.49×10^{-11}	3.02×10^{-11}	7.77×10^{-9}	1.69×10^{-9}
F28	5.97×10^{-9}	3.69×10^{-11}	3.02×10^{-11}	4.08×10^{-11}	1.73×10^{-6}	3.02×10^{-11}	3.08×10^{-8}	3.02×10^{-11}	1.61×10^{-10}	4.64×10^{-5}
F29	3.34×10^{-11}	4.50×10^{-11}	6.70×10^{-11}	3.02×10^{-11}	7.22×10^{-6}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	2.02×10^{-8}	2.39×10^{-8}
F30	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	5.49×10^{-11}	3.02×10^{-11}	3.34×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	9.76×10^{-10}

The Friedman test aims to examine whether there are significant differences in the average rankings of all algorithms across all test functions, where lower rankings indicate better performance. It assumes that if all algorithms have the same performance, their

average rankings across all test functions should be equal. As shown in Figure 8, it is clear that the HARO algorithm has the lowest average ranking, around 1.34, indicating its overall superior performance among the tested benchmark functions. Following HARO are the IARO and ARO algorithms, with average rankings of 3.1034 and 3.9655, respectively, indicating relatively good performance. However, the AVOA algorithm has the highest average ranking, approximately 6.3793, indicating relatively poor optimization performance.

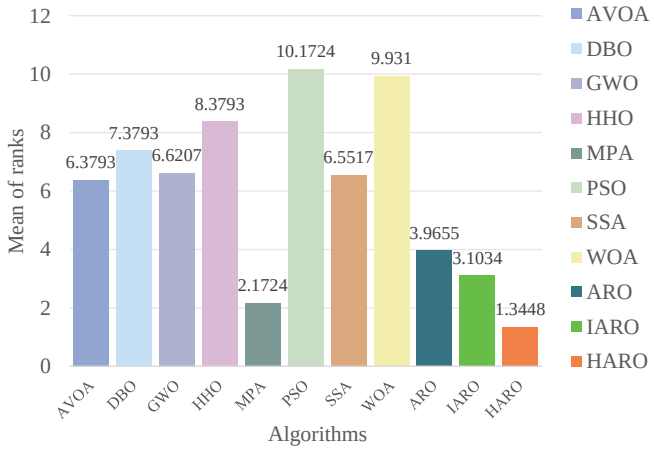

Figure 8. Rankings of 11 algorithms based on Friedman test on 30-D CEC2017 functions.

5.1.5. Ablation Study

To validate the contribution of each improvement strategy to HARO's overall performance, we conducted ablation studies. Based on the original ARO algorithm, three partially improved versions were constructed: EARO with the elite strategy, DEARO with the dual exploration switching strategy, and MARO with the population memory migration strategy. These versions were then compared with the original ARO using the CEC2017 test functions. Table 3 presents the results of the above algorithms, with each algorithm independently tested 30 times in the same test environment.

It is clear that each improvement strategy exhibits superior performance relative to the original algorithm. The EARO algorithm, incorporating the elite strategy, achieves better optimal values than the original ARO in over half of the benchmark functions. This is because elite individuals within the population can guide others towards the optimal direction, accelerating the convergence speed of the algorithm and continuously generating new individuals in the excellent regions of the solution space, thereby constantly improving the overall quality of the population. The DEARO algorithm, with the introduction of a dual exploration switching strategy, shows significant improvements over the original ARO algorithm in functions other than F11 and F30. This indicates that the introduction of this strategy dramatically enhances the overall exploration capability of the algorithm, enabling it to search for optimal solutions over a broader range and avoid becoming stuck in local optima. The MARO algorithm, with the introduction of a population memory migration mechanism, significantly enhances the ability of the algorithm to escape from local optima by effectively improving population diversity. Ablation study results show that MARO significantly improves performance over the original ARO algorithm in all benchmark functions. This improvement stems from the population memory migration mechanism, which maintains population diversity, avoids premature convergence to suboptimal solutions, and enhances the algorithm's global search ability. Therefore, introducing these three strategies effectively enhances ARO's performance and optimization capabilities on different benchmark functions, validating their positive contributions to the overall performance of the HARO algorithm.

Table 3. Comparison of HARO variant strategies on the CEC2017 test suite with 30 dim.

Function	Index	ARO	EARO	DEARO	MARO	Function	Index	ARO	EARO	DEARO	MARO
F1	Mean	3.84×10^{3}	4.71×10^{3}	1.92×10^{3}	1.49×10^{3}	F17	Mean	1.95×10^{3}	1.92×10^{3}	1.89×10^{3}	1.86×10^{3}
	Std	2.86×10^{3}	4.23×10^{3}	1.86×10^{3}	2.30×10^{3}		Std	1.57×10^{2}	1.30×10^{2}	1.21×10^{2}	1.26×10^{2}
	Best	2.49×10^{2}	2.29×10^{2}	1.41×10^{2}	1.00×10^{2}		Best	1.74×10^{3}	1.72×10^{3}	1.73×10^{3}	1.73×10^{3}
	Rank	3	4	2	1		Rank	4	3	2	1
F3	Mean	2.32×10^{4}	1.47×10^{4}	1.79×10^{4}	3.00×10^{2}	F18	Mean	9.69×10^{4}	3.28×10^{4}	3.27×10^{4}	1.83×10^{3}
	Std	4.98×10^{3}	4.04×10^{3}	4.34×10^{3}	4.47×10^{-1}		Std	7.48×10^{4}	2.04×10^{4}	1.61×10^{4}	5.33×10^{0}
	Best	1.68×10^{4}	8.24×10^{3}	1.08×10^{4}	3.00×10^{2}		Best	3.26×10^{4}	4.34×10^{3}	8.73×10^{3}	1.81×10^{3}
	Rank	4	2	3	1		Rank	4	3	2	1
F4	Mean	5.01×10^{2}	4.85×10^{2}	4.98×10^{2}	4.53×10^{2}	F19	Mean	6.52×10^{3}	3.17×10^{3}	2.90×10^{3}	1.91×10^{3}
	Std	2.51×10^{1}	3.43×10^{1}	1.81×10^{1}	4.13×10^{1}		Std	4.52×10^{3}	3.03×10^{3}	1.54×10^{3}	3.71×10^{0}
	Best	4.35×10^{2}	4.00×10^{2}	4.65×10^{2}	4.00×10^{2}		Best	2.06×10^{3}	1.93×10^{3}	1.95×10^{3}	1.91×10^{3}
	Rank	4	2	3	1		Rank	4	3	2	1
F5	Mean	5.75×10^{2}	5.76×10^{2}	5.61×10^{2}	5.59×10^{2}	F20	Mean	2.30×10^{3}	2.28×10^{3}	2.24×10^{3}	2.24×10^{3}
	Std	2.01×10^{1}	2.08×10^{1}	1.80×10^{1}	1.59×10^{1}		Std	1.73×10^{2}	1.26×10^{2}	1.14×10^{2}	9.95×10^{1}
	Best	5.39×10^{2}	5.38×10^{2}	5.40×10^{2}	5.34×10^{2}		Best	2.04×10^{3}	2.07×10^{3}	2.04×10^{3}	2.13×10^{3}
	Rank	3	4	2	1		Rank	4	3	1	2
F6	Mean	6.00×10^{2}	6.00×10^{2}	6.00×10^{2}	6.00×10^{2}	F21	Mean	2.37×10^{3}	2.37×10^{3}	2.35×10^{3}	2.36×10^{3}
	Std	2.79×10^{-1}	4.38×10^{-1}	2.19×10^{-3}	1.22×10^{-3}		Std	2.21×10^{1}	1.78×10^{1}	1.58×10^{1}	1.69×10^{1}
	Best	6.00×10^{2}	6.00×10^{2}	6.00×10^{2}	6.00×10^{2}		Best	2.33×10^{3}	2.34×10^{3}	2.33×10^{3}	2.33×10^{3}
	Rank	3	4	2	1		Rank	4	3	1	2
F7	Mean	8.15×10^{2}	8.25×10^{2}	7.91×10^{2}	7.93×10^{2}	F22	Mean	2.30×10^{3}	2.30×10^{3}	2.30×10^{3}	2.30×10^{3}
	Std	1.65×10^{1}	3.12×10^{1}	1.67×10^{1}	1.51×10^{1}		Std	1.21×10^{0}	1.12×10^{0}	9.90×10^{-1}	9.44×10^{-1}
	Best	7.80×10^{2}	7.78×10^{2}	7.65×10^{2}	7.59×10^{2}		Best	2.30×10^{3}	2.30×10^{3}	2.30×10^{3}	2.30×10^{3}
	Rank	3	4	1	2		Rank	4	3	2	1
F8	Mean	8.79×10^{2}	8.72×10^{2}	8.59×10^{2}	8.59×10^{2}	F23	Mean	2.73×10^{3}	2.72×10^{3}	2.70×10^{3}	2.71×10^{3}
	Std	1.92×10^{1}	1.38×10^{1}	1.28×10^{1}	1.76×10^{1}		Std	2.19×10^{1}	1.70×10^{1}	1.68×10^{1}	2.10×10^{1}
	Best	8.50×10^{2}	8.46×10^{2}	8.36×10^{2}	8.20×10^{2}		Best	2.70×10^{3}	2.69×10^{3}	2.67×10^{3}	2.68×10^{3}
	Rank	4	3	2	1		Rank	4	3	1	2
F9	Mean	1.11×10^{3}	1.18×10^{3}	9.18×10^{2}	9.10×10^{2}	F24	Mean	2.89×10^{3}	2.89×10^{3}	2.88×10^{3}	2.87×10^{3}
	Std	1.95×10^{2}	4.00×10^{2}	3.23×10^{1}	1.25×10^{1}		Std	1.93×10^{1}	1.89×10^{1}	1.61×10^{1}	1.93×10^{1}
	Best	9.10×10^{2}	9.22×10^{2}	9.00×10^{2}	9.00×10^{2}		Best	2.86×10^{3}	2.86×10^{3}	2.85×10^{3}	2.85×10^{3}
	Rank	3	4	2	1		Rank	3	4	2	1
F10	Mean	4.03×10^{3}	3.41×10^{3}	3.69×10^{3}	3.67×10^{3}	F25	Mean	2.91×10^{3}	2.89×10^{3}	2.90×10^{3}	2.89×10^{3}
	Std	5.47×10^{2}	5.11×10^{2}	3.34×10^{2}	5.23×10^{2}		Std	1.97×10^{1}	9.81×10^{0}	1.77×10^{1}	1.87×10^{0}
	Best	2.29×10^{3}	1.62×10^{3}	3.05×10^{3}	2.34×10^{3}		Best	2.88×10^{3}	2.88×10^{3}	2.88×10^{3}	2.88×10^{3}
	Rank	4	1	3	2		Rank	4	2	3	1
F11	Mean	1.18×10^{3}	1.16×10^{3}	1.19×10^{3}	1.14×10^{3}	F26	Mean	4.34×10^{3}	3.95×10^{3}	3.62×10^{3}	3.75×10^{3}
	Std	3.77×10^{1}	3.37×10^{1}	3.59×10^{1}	1.49×10^{1}		Std	9.32×10^{2}	8.09×10^{2}	7.64×10^{2}	7.43×10^{2}
	Best	1.13×10^{3}	1.12×10^{3}	1.14×10^{3}	1.12×10^{3}		Best	2.80×10^{3}	2.80×10^{3}	2.80×10^{3}	2.80×10^{3}
	Rank	3	2	4	1		Rank	4	3	1	2
F12	Mean	5.00×10^{5}	9.24×10^{4}	2.27×10^{5}	8.94×10^{3}	F27	Mean	3.23×10^{3}	3.22×10^{3}	3.23×10^{3}	3.22×10^{3}
	Std	3.73×10^{5}	8.06×10^{4}	1.31×10^{5}	6.29×10^{3}		Std	1.36×10^{1}	1.25×10^{1}	1.08×10^{1}	1.37×10^{1}
	Best	8.90×10^{4}	1.05×10^{4}	2.71×10^{4}	1.62×10^{3}		Best	3.21×10^{3}	3.21×10^{3}	3.21×10^{3}	3.20×10^{3}
	Rank	4	2	3	1		Rank	4	2	3	1
F13	Mean	1.44×10^{4}	1.28×10^{4}	9.80×10^{3}	1.35×10^{3}	F28	Mean	3.25×10^{3}	3.22×10^{3}	3.25×10^{3}	3.16×10^{3}
	Std	1.08×10^{4}	1.17×10^{4}	5.12×10^{3}	2.11×10^{1}		Std	2.57×10^{1}	1.73×10^{1}	2.23×10^{1}	5.24×10^{1}
	Best	1.46×10^{3}	1.39×10^{3}	2.55×10^{3}	1.32×10^{3}		Best	3.21×10^{3}	3.19×10^{3}	3.21×10^{3}	3.10×10^{3}
	Rank	4	3	2	1		Rank	4	2	3	1
F14	Mean	3.51×10^{3}	1.76×10^{3}	1.49×10^{3}	1.43×10^{3}	F29	Mean	3.62×10^{3}	3.55×10^{3}	3.47×10^{3}	3.41×10^{3}
	Std	2.55×10^{3}	1.02×10^{3}	2.36×10^{1}	1.05×10^{1}		Std	1.57×10^{2}	1.27×10^{2}	1.00×10^{2}	7.97×10^{1}
	Best	1.49×10^{3}	1.43×10^{3}	1.46×10^{3}	1.41×10^{3}		Best	3.38×10^{3}	3.35×10^{3}	3.36×10^{3}	3.34×10^{3}
	Rank	4	3	2	1		Rank	4	3	2	1
F15	Mean	5.31×10^{3}	4.69×10^{3}	2.91×10^{3}	1.52×10^{3}	F30	Mean	9.05×10^{3}	9.69×10^{3}	1.01×10^{4}	5.57×10^{3}
	Std	4.44×10^{3}	8.05×10^{3}	1.62×10^{3}	1.36×10^{1}		Std	1.98×10^{3}	3.02×10^{3}	4.42×10^{3}	4.60×10^{2}
	Best	1.59×10^{3}	1.53×10^{3}	1.62×10^{3}	1.51×10^{3}		Best	6.32×10^{3}	6.19×10^{3}	6.60×10^{3}	5.13×10^{3}
	Rank	4	3	2	1		Rank	2	3	4	1
F16	Mean	2.45×10^{3}	2.36×10^{3}	2.25×10^{3}	2.31×10^{3}						
	Std	2.48×10^{2}	2.50×10^{2}	1.87×10^{2}	2.32×10^{2}						
	Best	2.03×10^{3}	1.93×10^{3}	1.79×10^{3}	1.85×10^{3}						
	Rank	4	3	1	2						

5.2. Application of HARO+ in UAV Path Planning

We conducted comparative simulation tests for UAV path planning across various 2D and 3D terrains with different levels of complexity. First, we evaluated the performance of the HARO algorithm in addressing UAV path planning problems in complex environments, comparing it with other algorithms. Finally, we verified the significant contribution of incorporating the trajectory optimization strategy HARO+ in reducing flight costs, as well as the impact of integrating this strategy into other algorithms.

5.2.1. Setting the Simulation Environment

In this simulation, the comparison algorithms include well-performing algorithms for this problem: Spherical Particle Swarm Optimization (SPSO) [30], and widely recognized Particle Swarm Optimization (PSO) [24], Whale Optimization algorithm (WOA) [60], and Sparrow Search algorithm (SSA) [63]. Additionally, newly proposed algorithms in recent years include Northern Goshawk Optimization (NGO) [69], Beluga Whale Optimization (BWO) [70], Dung Beetle Optimizer (DBO) [64], artificial rabbits optimization (ARO) [31], and an improved version of artificial rabbits optimization (IARO) [51].

To ensure fairness in the simulations and avoid the randomness of algorithms, we independently repeated each scenario 30 times. We introduced average, minimum, standard deviation, and rankings based on the minimum average as evaluation metrics, derived from the total cost calculated during the algorithm's iterations. This total cost consists of collision cost, distance cost, altitude cost, and turning cost, and these metrics were used to assess the overall performance of the algorithms. All participating algorithms employed the spherical vector-based encoding method introduced in Section 3. Specifically, UAV trajectories are represented by vectors of radius, inclination angle, and azimuth angle, and optimized using either the HARO algorithm or other comparative algorithms to minimize the total cost function while satisfying all constraints. While some randomness is involved in the search process, the final trajectories are determined through controlled optimization and transformed into Cartesian coordinates to identify the best route. In the two-dimensional environment, we set up complex scenarios with four different obstacle layouts to validate HARO's performance. Some parameter settings are as follows: the population size is 100, the maximum number of iterations is 200, the starting point position is (150, 100), the target point position is (950, 800), and there are 20 intermediate path points. In the three-dimensional scenario, we adopted two terrain map scenarios, one of which used real digital elevation model (DEM) data obtained from sensors [71]. We utilized complex scenarios with five different obstacle layouts in these two terrain maps. The population size is 200, and the maximum number of iterations is 100. The starting point position is (150, 100, 150), and the target point position is (950, 800, 150), with 20 intermediate path points.

5.2.2. HARO for 2D UAV Path Planning

Four complex scenarios were constructed in the two-dimensional path planning simulation, considering obstacles and threat areas distributed across multiple locations. Table 4 present the comparison results between the HARO algorithm and nine other optimization algorithms in these scenarios. The HARO algorithm is observed to successfully find reasonable UAV routes in all scenarios. In contrast, some algorithms, such as IARO, SSA, WOA, and ARO, fail to obtain feasible solutions in certain complex scenarios, resulting in infinite average values. This fully demonstrates the superior stability and robustness of the proposed HARO algorithm.

Further comparisons based on average rankings show that the HARO algorithm ranks first in scenario 2 and second in the remaining scenarios, outperforming the PSO algorithm, NGO algorithm, DBO algorithm, and the original ARO algorithm and its improved version IARO. This outstanding performance is attributed to the introduction of the population memory migration mechanism in the HARO algorithm, which can maintain the superior positions of population individuals during iterations and continuously explore

better solutions. Figure 9 depicts the average convergence curves of various algorithms in four typical two-dimensional scenarios. The HARO algorithm's convergence curve shows a rapid downward trend in each scenario, converging to a low fitness value with few iterations. However, other algorithms, such as IARO and WOA, fail to find suitable path solutions in every scenario when run independently for 30 times. The average convergence curve value of the HARO algorithm is the lowest, verifying its excellent convergence performance and stability in various two-dimensional environments. Figure 10 visually compares the flight paths planned by the HARO algorithm with eight other algorithms (NGO, DBO, IARO, PSO, SSA, WOA, ARO, and BWO) based on their respective optimal solutions in four typical two-dimensional scenarios. In these diagrams, the yellow square marks the initial point, the green pentagon denotes the target point, and the red line represents the HARO algorithm's path. The figure shows that the HARO algorithm effectively avoids obstacles and minimizes flight distance in all scenarios. Especially in scenarios 1 and 2, the HARO algorithm performs exceptionally well, successfully avoiding all obstacles and planning relatively shorter paths. In comparison, the other algorithms perform relatively poorly in these scenarios. For instance, in scenario 1, the BWO and PSO algorithms produce more winding and longer paths, while, in scenario 2, the IARO algorithm shows excessive detouring. Comparatively, the HARO algorithm's path is more intuitive with better obstacle avoidance, demonstrating its efficiency and reliability in path planning. In scenarios 3 and 4, despite more complex obstacle distributions, the HARO algorithm can still flexibly adjust the path to avoid collisions while maintaining a lower total flight cost. Other algorithms (such as WOA and IARO) exhibit longer detours or direct collisions with obstacles in some scenarios, highlighting their limitations in path planning. Overall, the flight paths planned by HARO ensure precise avoidance of all obstacles while maintaining a shorter distance and lower total cost, demonstrating the outstanding performance of this algorithm.

Table 4. Simulation results of eight algorithms for UAV path planning in 2D environments across four scenarios.

Scenario	Algorithms	Mean	Best	Std.	Rank	Scenario	Algorithms	Mean	Best	Std.	Rank
1	NGO	1.15×10^3	1.09×10^3	4.54×10^1	5	2	NGO	1.29×10^3	1.13×10^3	8.75×10^1	3
	DBO	1.09×10^3	1.07×10^3	4.07×10^1	1		DBO	1.27×10^3	1.10×10^3	1.51×10^2	2
	IARO	Inf	1.29×10^3	-	9		IARO	Inf	1.27×10^3	-	8
	PSO	1.26×10^3	1.15×10^3	7.67×10^1	7		PSO	1.36×10^3	1.13×10^3	1.24×10^2	7
	SSA	1.13×10^3	1.07×10^3	7.14×10^1	3		SSA	1.29×10^3	1.28×10^3	1.07×10^1	4
	WOA	1.27×10^3	1.08×10^3	1.28×10^2	8		WOA	Inf	1.33×10^3	-	9
	ARO	1.13×10^3	1.07×10^3	6.61×10^1	4		ARO	1.30×10^3	1.12×10^3	5.35×10^1	5
	BWO	1.23×10^3	1.07×10^3	1.81×10^2	6		BWO	1.36×10^3	1.10×10^3	8.48×10^1	6
	HARO	1.10×10^3	1.08×10^3	6.65×10^0	2		HARO	1.24×10^3	1.10×10^3	1.07×10^2	1
3	NGO	1.42×10^3	1.15×10^3	1.39×10^2	5	4	NGO	1.13×10^3	1.09×10^3	1.69×10^1	5
	DBO	1.33×10^3	1.14×10^3	1.35×10^2	3		DBO	1.11×10^3	1.07×10^3	3.95×10^1	3
	IARO	Inf	Inf	-	9		IARO	Inf	1.27×10^3	-	9
	PSO	1.42×10^3	1.29×10^3	7.42×10^1	4		PSO	1.21×10^3	1.14×10^3	5.75×10^1	8
	SSA	Inf	1.15×10^3	-	7		SSA	1.09×10^3	1.07×10^3	1.91×10^1	1
	WOA	Inf	1.65×10^3	-	8		WOA	1.15×10^3	1.11×10^3	2.88×10^1	6
	ARO	Inf	1.13×10^3	-	6		ARO	1.11×10^3	1.07×10^3	2.61×10^1	4
	BWO	1.22×10^3	1.17×10^3	2.01×10^1	1		BWO	1.15×10^3	1.09×10^3	3.00×10^1	7
	HARO	1.28×10^3	1.16×10^3	6.74×10^1	2		HARO	1.10×10^3	1.07×10^3	2.07×10^1	2

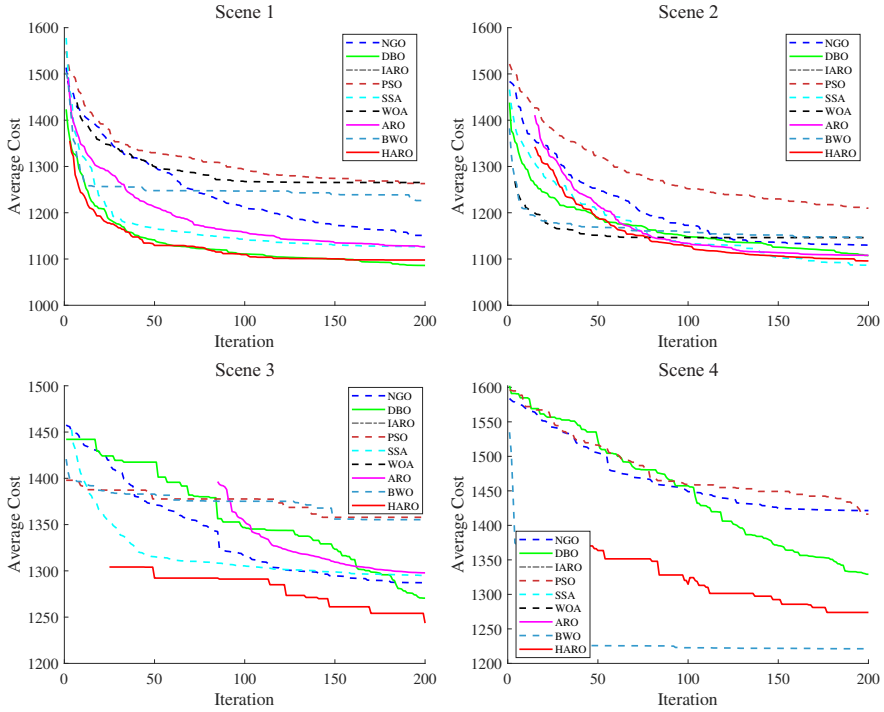

Figure 9. Comparison of convergence curves of multiple algorithms in four 2D terrain scenarios.

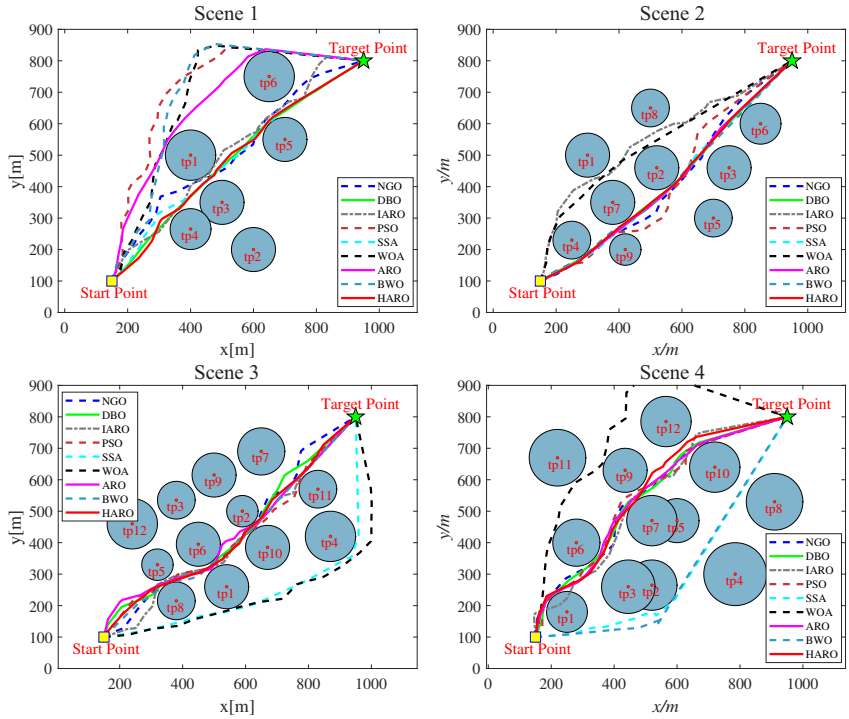

Figure 10. Trajectory plots of multiple algorithms in four 2D terrain scenarios.

5.2.3. HARO for 3D UAV Path Planning

Table 5 display the simulation results for the three-dimensional UAV path planning task under five different scenarios. It can be seen that the HARO algorithm performs well in different scenarios. Based on average rankings, the HARO algorithm ranks first in three scenarios and second or third in the remaining scenarios. This is because the different obstacles in the scenarios lead to changes in the solution space. According to the No Free Lunch Theorem [72], no algorithm can perform the best in all possible optimization

problems. In other words, there is no universal optimization algorithm that can be optimal for all kinds of problems.

Table 5. Comparative simulation results of eight algorithms for UAV path planning in 3D environments across five scenarios.

Scenario	Algorithms	Mean	Best	Std.	Rank	Scenario	Algorithms	Mean	Best	Std.	Rank
1	NGO	6.34×10^3	5.94×10^3	1.99×10^2	4	2	NGO	7.24×10^3	6.62×10^3	3.45×10^2	4
	DBO	5.93×10^3	5.77×10^3	9.81×10^1	2		DBO	7.01×10^3	6.58×10^3	3.24×10^2	3
	IARO	Inf	6.98×10^3	-	9		IARO	Inf	Inf	-	9
	SPSO	7.04×10^3	6.53×10^3	1.78×10^2	6		SPSO	7.46×10^3	6.87×10^3	3.28×10^2	6
	SSA	7.00×10^3	5.94×10^3	6.21×10^2	5		SSA	7.62×10^3	7.23×10^3	1.26×10^2	7
	WOA	Inf	6.04×10^3	-	8		WOA	8.04×10^3	7.23×10^3	2.42×10^2	8
	ARO	6.08×10^3	5.92×10^3	1.24×10^2	3		ARO	6.97×10^3	6.47×10^3	1.46×10^2	2
	BWO	7.13×10^3	6.42×10^3	4.43×10^2	7		BWO	7.35×10^3	6.69×10^3	3.03×10^2	5
	HARO	5.89×10^3	5.68×10^3	1.27×10^2	1		HARO	6.94×10^3	6.47×10^3	2.81×10^2	1
3	NGO	7.86×10^3	6.66×10^3	5.50×10^2	4	4	NGO	7.50×10^3	6.87×10^3	2.52×10^2	4
	DBO	7.14×10^3	6.55×10^3	2.49×10^2	2		DBO	7.22×10^3	6.71×10^3	2.43×10^2	3
	IARO	Inf	Inf	-	9		IARO	Inf	7.64×10^3	-	9
	SPSO	7.93×10^3	7.25×10^3	4.09×10^2	5		SPSO	7.59×10^3	6.95×10^3	2.41×10^2	5
	SSA	Inf	6.45×10^3	-	6		SSA	Inf	6.74×10^3	-	7
	WOA	Inf	7.69×10^3	-	8		WOA	Inf	7.29×10^3	-	8
	ARO	Inf	6.88×10^3	-	7		ARO	7.19×10^3	6.92×10^3	1.33×10^2	2
	BWO	6.99×10^3	6.80×10^3	7.14×10^1	1		BWO	8.65×10^3	7.38×10^3	7.17×10^2	6
	HARO	7.29×10^3	6.75×10^3	2.74×10^2	3		HARO	7.17×10^3	6.87×10^3	1.14×10^2	1
5	NGO	6.47×10^3	6.15×10^3	1.59×10^2	5						
	DBO	5.89×10^3	5.61×10^3	1.32×10^2	1						
	IARO	7.75×10^3	7.14×10^3	3.36×10^2	9						
	SPSO	7.06×10^3	6.78×10^3	1.51×10^2	6						
	SSA	6.35×10^3	5.95×10^3	2.11×10^2	4						
	WOA	7.18×10^3	6.48×10^3	3.81×10^2	7						
	ARO	6.02×10^3	5.91×10^3	5.65×10^1	3						
	BWO	7.47×10^3	6.88×10^3	2.90×10^2	8						
	HARO	5.95×10^3	5.78×10^3	8.68×10^1	2						

Moreover, not all algorithms can guarantee reasonable paths that satisfy multiple constraints in every independently repeated test. For example, the rankings of IARO, SSA, WOA, and ARO vary across different scenarios. Compared with the two-dimensional simulation environment, the three-dimensional environment introduces height constraints, pitch angle constraints, and more complex obstacles, significantly increasing the difficulty for algorithms to search for candidate solutions in the solution space. However, the HARO algorithm maintains relatively stable rankings across all scenarios. This further confirms its strong search capability under the dual exploration strategy, allowing it to find reasonable paths under multiple constraints in complex environments.

Figures 11 and 12 display the optimal trajectory maps generated by various algorithms from three-dimensional and top-down perspectives across five scenarios, each containing differently positioned black cylindrical and red spherical obstacles. In densely obstructed environments, some algorithms, constrained by path rationality requirements and multiple constraints, cannot fully explore the solution space. Consequently, these algorithms struggle to find suitable paths in high-obstacle areas and resort to longer routes. For instance, in scenario 2, IARO and SSA, and, in scenario 3, WOA and NGO, fail to effectively navigate dense obstacles, significantly increasing flight costs. Conversely, the HARO algorithm consistently finds relatively optimal paths across all five terrains, though not necessarily the shortest in every case. Figure 13 illustrates the iteration curves across scenarios, highlighting each algorithm's optimal values. Notably, most algorithms tend toward local optima in complex environments, with HARO achieving a relatively high perfor-

mance, though it experiences slower convergence in certain cases. This slower convergence stems from the larger search space in complex, obstacle-dense environments, complicating path planning. Despite the limited iterations in this experiment, HARO demonstrates strong adaptability.

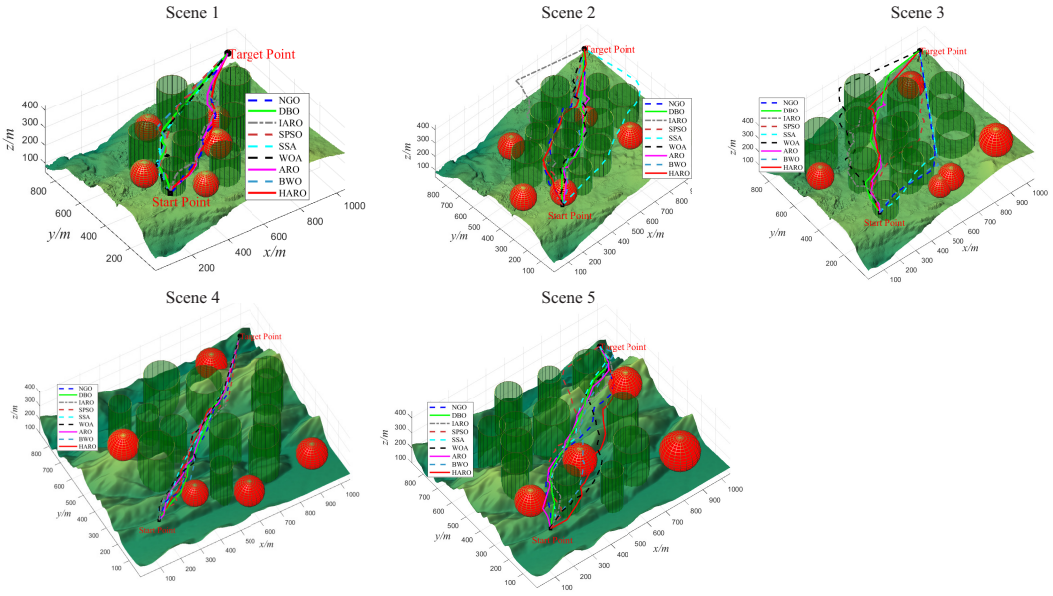

Figure 11. Trajectory plots of multiple algorithms in five 3D terrain scenarios.

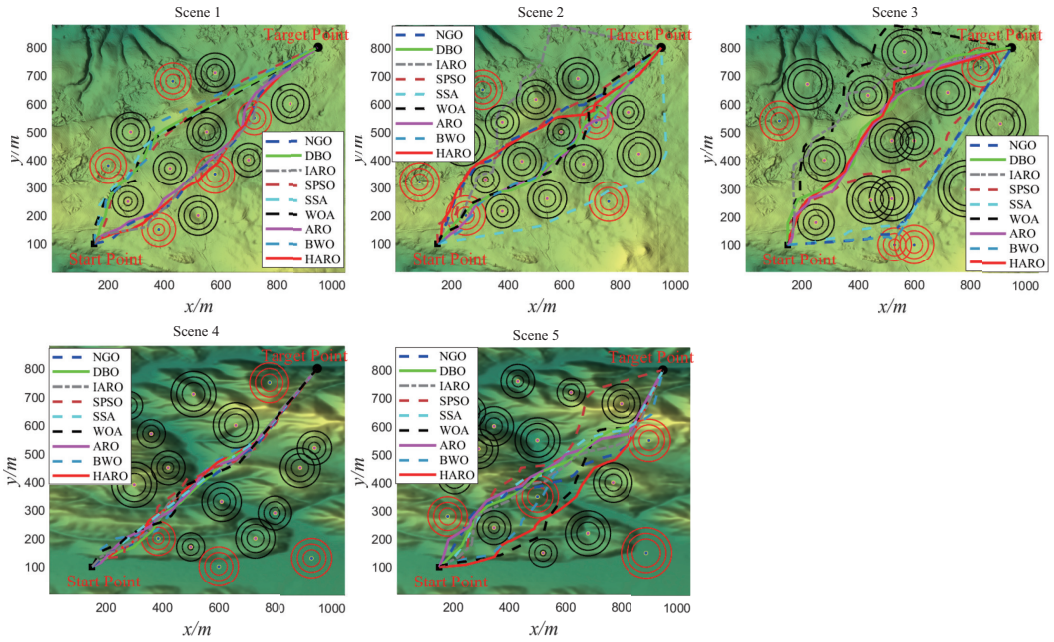

Figure 12. Top-view trajectory plots of multiple algorithms in five 3D terrain scenarios.

In conclusion, the HARO algorithm exhibits robust performance in three-dimensional UAV path planning, consistently identifying high-quality paths that satisfy multiple constraints in challenging environments. This highlights HARO's significant potential for UAV path planning applications. However, like most algorithms, HARO-generated paths still contain redundant waypoints. In the next section, we will use HARO+ to validate the effectiveness of the trajectory optimization strategy in these environments.

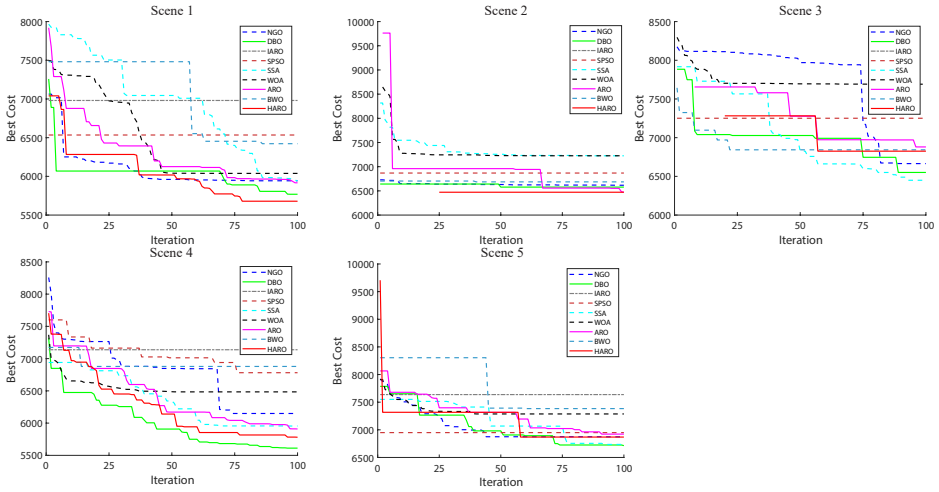

Figure 13. Comparison of convergence curves for multiple algorithms across five 3D terrain scenarios based on optimal values.

5.2.4. HARO+ with Trajectory Optimization in 2D/3D

Tables 6 and 7 present the comparison results of the HARO+ algorithm with a trajectory optimization strategy in both two-dimensional and three-dimensional environments and some other comparative algorithms that also employ this strategy. With a uniform threshold of 2 in the two-dimensional environment and 10 in the three-dimensional environment, all algorithms exhibit significant reductions in flight costs. Among them, in the two-dimensional environment scenario 2, the flight cost of the DBO algorithm is reduced by up to 4.5%. The average fitness cost of the HARO+ algorithm ranks high among all the comparative algorithms. In 30 independent repeated runs, the total number of path points is 660, and the optimized total number of path points in all scenarios is greatly simplified, with a compression rate of approximately 50–80%. Therefore, it can be seen that the algorithm we proposed has general practicality.

Table 6. Comparative results of HARO+ and other algorithms before and after trajectory optimization in the 2D environment. Note: "Path Points" represent the total number of path points obtained by running each algorithm 30 times after applying the trajectory optimization strategy. "Improve" represents the reduction ratio of the total flight cost before and after applying the trajectory optimization strategy. "CR" indicates the reduction ratio of the number of path points before and after applying the trajectory optimization strategy.

Scenario	Algorithms	Mean	Best	Std.	Algorithms	Mean	Best	Std.	Path Points	Improve	CR
1	NGO	1.15×10^3	1.09×10^3	4.54×10^1	NGO+	1.14×10^3	1.09×10^3	4.73×10^1	408	0.59%	61.82%
	DBO	1.09×10^3	1.07×10^3	4.07×10^1	DBO+	1.08×10^3	1.07×10^3	4.03×10^1	316	0.11%	47.88%
	ARO	1.13×10^3	1.07×10^3	6.61×10^1	ARO+	1.13×10^3	1.07×10^3	7.07×10^1	376	0.08%	56.97%
	BWO	1.23×10^3	1.07×10^3	1.81×10^2	BWO+	1.20×10^3	1.07×10^3	1.63×10^2	386	1.76%	58.48%
	HARO	1.10×10^3	1.08×10^3	6.65×10^0	HARO+	1.10×10^3	1.08×10^3	1.85×10^1	363	0.12%	55.00%
2	DBO	1.27×10^3	1.10×10^3	1.51×10^2	DBO+	1.21×10^3	1.10×10^3	9.91×10^1	429	4.50%	65.00%
	PSO	1.36×10^3	1.13×10^3	1.24×10^2	PSO+	1.30×10^3	1.12×10^3	1.16×10^2	444	4.47%	67.27%
	ARO	1.30×10^3	1.12×10^3	5.35×10^1	ARO+	1.29×10^3	1.15×10^3	5.59×10^1	306	0.38%	46.36%
	BWO	1.36×10^3	1.10×10^3	8.48×10^1	BWO+	1.32×10^3	1.09×10^3	1.15×10^2	361	2.48%	54.70%
	HARO	1.24×10^3	1.10×10^3	1.07×10^2	HARO+	1.20×10^3	1.09×10^3	8.62×10^1	454	3.75%	68.79%
3	NGO	1.42×10^3	1.15×10^3	1.39×10^2	NGO+	1.38×10^3	1.14×10^3	1.43×10^2	392	2.88%	59.39%
	DBO	1.33×10^3	1.14×10^3	1.35×10^2	DBO+	1.32×10^3	1.13×10^3	1.19×10^2	414	0.65%	62.73%
	PSO	1.42×10^3	1.29×10^3	7.42×10^1	PSO+	1.40×10^3	1.25×10^3	8.98×10^1	424	1.33%	64.24%
	BWO	1.22×10^3	1.17×10^3	2.01×10^1	BWO+	1.22×10^3	1.16×10^3	1.97×10^1	135	0.11%	20.45%
	HARO	1.28×10^3	1.16×10^3	6.74×10^1	HARO+	1.28×10^3	1.15×10^3	9.03×10^1	428	0.62%	64.85%
4	DBO	1.11×10^3	1.07×10^3	3.95×10^1	DBO	1.10×10^3	1.07×10^3	3.92×10^1	310	0.23%	46.97%
	SSA	1.09×10^3	1.07×10^3	1.91×10^1	SSA	1.08×10^3	1.07×10^3	1.82×10^1	287	0.33%	43.48%
	BWO	1.15×10^3	1.09×10^3	3.00×10^1	BWO+	1.14×10^3	1.08×10^3	3.37×10^1	244	0.17%	36.97%
	HARO	1.10×10^3	1.07×10^3	2.07×10^1	HARO	1.09×10^3	1.07×10^3	2.00×10^1	327	0.29%	49.55%

Table 7. Comparative results of HARO+ and other algorithms before and after trajectory optimization in the 3D environment.

Scenario	Algorithms	Mean	Best	Std.	Algorithms	Mean	Best	Std.	Path Points	Improve	CR
1	NGO	6.34×10^3	5.94×10^3	1.99×10^2	NGO+	6.30×10^3	5.91×10^3	2.08×10^2	436	0.63%	66.06%
	DBO	5.93×10^3	5.77×10^3	9.81×10^1	DBO+	5.89×10^3	5.71×10^3	1.06×10^2	332	0.65%	50.30%
	SPSO	7.04×10^3	6.53×10^3	1.78×10^2	SPSO+	7.01×10^3	6.40×10^3	2.06×10^2	504	0.46%	76.36%
	SSA	7.00×10^3	5.94×10^3	6.21×10^2	SSA+	6.96×10^3	5.86×10^3	6.44×10^2	444	0.58%	67.27%
	ARO	6.08×10^3	5.92×10^3	1.24×10^2	ARO+	6.04×10^3	5.87×10^3	1.18×10^2	418	0.69%	63.33%
	BWO	7.13×10^3	6.42×10^3	4.43×10^2	BWO+	7.05×10^3	6.26×10^3	4.68×10^2	465	1.18%	70.45%
	HARO	5.89×10^3	5.68×10^3	1.27×10^2	HARO+	5.85×10^3	5.64×10^3	1.24×10^2	368	0.75%	55.76%
2	NGO	7.24×10^3	6.62×10^3	3.45×10^2	NGO+	7.18×10^3	6.59×10^3	3.34×10^2	544	0.87%	82.42%
	DBO	7.01×10^3	6.58×10^3	3.24×10^2	DBO+	6.98×10^3	6.55×10^3	3.23×10^2	552	0.37%	83.64%
	SPSO	7.46×10^3	6.87×10^3	3.28×10^2	SPSO+	7.41×10^3	6.86×10^3	3.40×10^2	547	0.59%	82.88%
	SSA	7.62×10^3	7.23×10^3	1.26×10^2	SSA+	7.60×10^3	7.21×10^3	1.30×10^2	392	0.27%	59.39%
	WOA	8.04×10^3	7.23×10^3	2.42×10^2	WOA+	8.03×10^3	7.13×10^3	2.63×10^2	414	0.11%	62.73%
	ARO	6.97×10^3	6.47×10^3	1.46×10^2	ARO+	6.95×10^3	6.42×10^3	1.60×10^2	387	0.38%	58.64%
	BWO	7.35×10^3	6.69×10^3	3.03×10^2	BWO+	7.26×10^3	6.63×10^3	3.24×10^2	514	1.23%	77.88%
	HARO	6.94×10^3	6.47×10^3	2.81×10^2	HARO+	6.87×10^3	6.38×10^3	2.83×10^2	551	0.92%	83.48%
3	NGO	7.86×10^3	6.66×10^3	5.50×10^2	NGO+	7.79×10^3	6.66×10^3	5.56×10^2	558	0.89%	84.55%
	DBO	7.14×10^3	6.55×10^3	2.49×10^2	DBO+	7.08×10^3	6.57×10^3	2.57×10^2	532	0.81%	80.61%
	SPSO	7.93×10^3	7.25×10^3	4.09×10^2	SPSO+	7.88×10^3	7.16×10^3	4.05×10^2	564	0.56%	85.45%
	BWO	6.99×10^3	6.80×10^3	7.14×10^1	BWO+	6.97×10^3	6.76×10^3	7.72×10^1	269	0.22%	40.76%
	HARO	7.29×10^3	6.75×10^3	2.74×10^2	HARO+	7.23×10^3	6.54×10^3	3.05×10^2	543	0.79%	82.27%
4	NGO	7.50×10^3	6.87×10^3	2.52×10^2	NGO+	7.47×10^3	6.64×10^3	2.91×10^2	506	0.42%	76.67%
	DBO	7.22×10^3	6.71×10^3	2.43×10^2	DBO+	7.16×10^3	6.58×10^3	2.47×10^2	505	0.83%	76.52%
	SPSO	7.59×10^3	6.95×10^3	2.41×10^2	SPSO+	7.56×10^3	6.92×10^3	2.69×10^2	519	0.38%	78.64%
	ARO	7.19×10^3	6.92×10^3	1.33×10^2	ARO+	7.15×10^3	6.92×10^3	1.29×10^2	518	0.58%	78.48%
	BWO	8.65×10^3	7.38×10^3	7.17×10^2	BWO+	8.59×10^3	7.23×10^3	7.37×10^2	546	0.71%	82.73%
	HARO	7.17×10^3	6.87×10^3	1.14×10^2	HARO+	7.08×10^3	6.80×10^3	1.28×10^2	502	1.21%	76.06%
5	NGO	6.47×10^3	6.15×10^3	1.59×10^2	NGO+	6.41×10^3	5.98×10^3	1.79×10^2	437	0.91%	66.21%
	DBO	5.89×10^3	5.61×10^3	1.32×10^2	DBO+	5.86×10^3	5.58×10^3	1.42×10^2	378	0.46%	57.27%
	IARO	7.75×10^3	7.14×10^3	3.36×10^2	IARO+	7.64×10^3	6.94×10^3	3.31×10^2	503	1.44%	76.21%
	SPSO	7.06×10^3	6.78×10^3	1.51×10^2	SPSO+	7.01×10^3	6.64×10^3	1.46×10^2	504	0.60%	76.36%
	SSA	6.35×10^3	5.95×10^3	2.11×10^2	SSA+	6.32×10^3	5.94×10^3	2.12×10^2	421	0.56%	63.79%
	WOA	7.18×10^3	6.48×10^3	3.81×10^2	WOA+	7.13×10^3	6.47×10^3	3.99×10^2	498	0.72%	75.45%
	ARO	6.02×10^3	5.91×10^3	5.65×10^1	ARO+	6.01×10^3	5.87×10^3	7.82×10^1	374	0.23%	56.67%
	BWO	7.47×10^3	6.88×10^3	2.90×10^2	BWO+	7.34×10^3	6.68×10^3	3.64×10^2	484	1.71%	73.33%
	HARO	5.95×10^3	5.78×10^3	8.68×10^1	HARO+	5.93×10^3	5.75×10^3	9.11×10^1	406	0.23%	61.52%

Figure 14 illustrates the comparison of flight trajectories before and after path simplification for HARO+ and two other algorithms in the two-dimensional environment. The figure demonstrates that the trajectories obtained by the HARO+ algorithm can effectively traverse obstacles while preserving key points near obstacles. The optimized paths not only retain the overall trajectory characteristics but also feature enlarged portions in the figure, indicating the simplified paths in some regions and resulting in smoother trajectories. Due to the complexity of the three-dimensional environment, which is not conducive to a clear and intuitive display of the before and after comparison, this paper only presents the comparison results of flight costs in the three-dimensional environment. In scenario 3, after incorporating the trajectory optimization strategy proposed in this paper, the WOA algorithm significantly simplifies the path points near obstacle 4. This indicates that the strategy can reduce redundant points in the flight path while ensuring flight safety and avoiding collisions with obstacles. It also ensures that key points near densely populated obstacle areas are well-preserved, while path points in sparsely populated obstacle areas are reduced, achieving smoother trajectories.

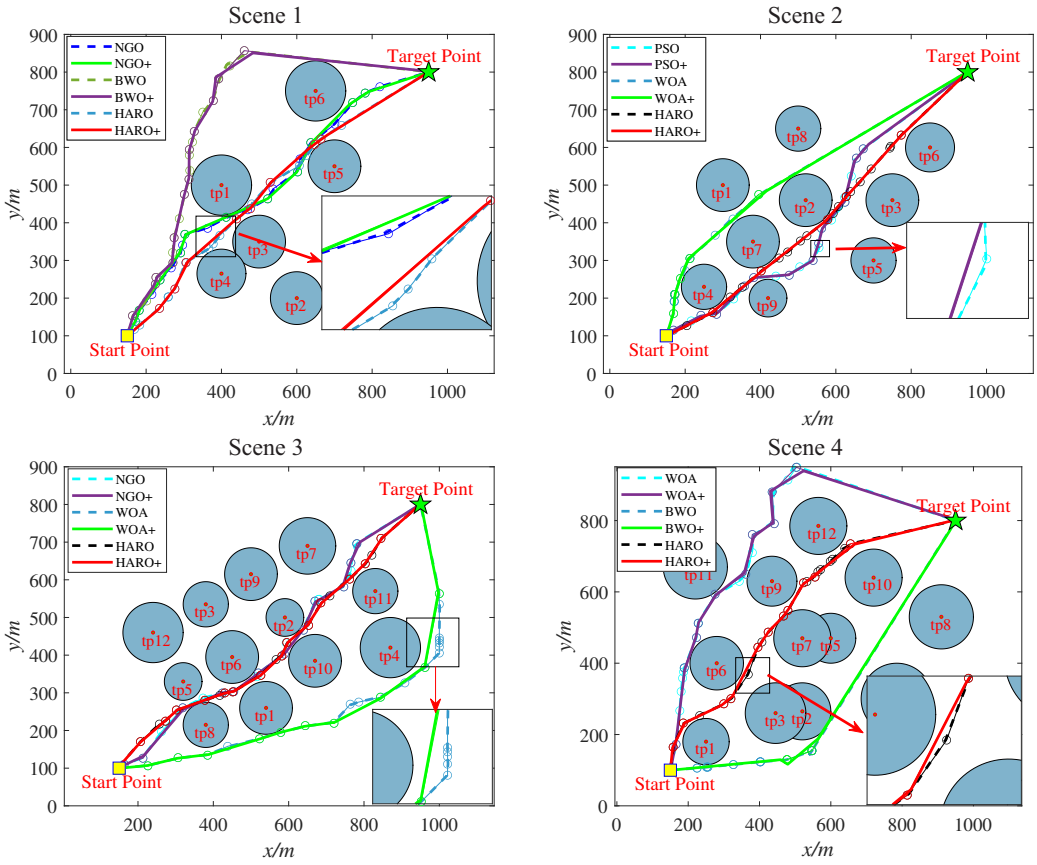

Figure 14. Comparison of paths before and after trajectory optimization in the 2D environment.

Overall, the trajectory optimization strategy achieves significant effects in both 2D and 3D environments. The performance of the HARO+ algorithm is particularly prominent in reducing flight costs, simplifying the number of path points, and ensuring flight safety and path smoothness. This indicates that the trajectory optimization strategy's application potential in UAV path planning is enormous, not only improving the efficiency of path planning but also enhancing its practicality and reliability.

6. Conclusions and Future Work

This paper proposes a hybrid multi-strategy artificial rabbit optimization algorithm called HARO+ to solve the multi-constraint UAV path planning problem involving numerous path points in complex environments. In the algorithm design, we propose a dual-exploration switching strategy to ensure more robust search capabilities and flexibility during the exploration phase, balancing exploration and exploitation. We propose a population migration memory mechanism to maintain population diversity and avoid falling into local optima during iteration. We validate the excellent search performance of HARO through comparative tests, complexity analysis, exploration and exploitation analysis, ablation studies, and Wilcoxon and Friedman statistical tests on 29 CEC2017 test functions, compared with 12 other classical or novel algorithms. HARO effectively preserves high-quality individuals throughout the iteration process, maintaining a balance between exploration and exploitation. In various complex 2D and 3D UAV flight scenarios, HARO provides more stable and reasonable high-quality paths than other methods, demonstrating significant advantages in high-dimensional problems.

In terms of trajectory optimization, we propose a key point retention trajectory optimization strategy called HARO+ to cope with obstacles of different densities. We retain key points in areas with many obstacles to ensure flight safety and reduce path points in sparse areas to generate safe, smooth, and low-cost UAV flight paths. This method can effectively reduce the number of path points and flight costs while ensuring flight safety. Test results

show that this universal strategy efficiently compresses redundant path points, significantly reducing fitness costs. However, there is still room for improvement when applied to more complex problems. In the future, refining constraints to address multi-objective optimization problems will be essential. Additionally, smoothing three-dimensional UAV paths remains a noteworthy challenge.

Author Contributions: Conceptualization, B.L., D.L. and G.X.; methodology, B.L. and Y.C.; software, B.L.; validation, B.L. and D.L.; formal analysis, B.L.; supervision, Y.C. and D.L.; visualization, B.L. and D.L.; investigation, B.L. and D.L.; resources, D.L. and K.L.; data curation, B.L. and Y.C.; writing—original draft, B.L. and D.L.; writing—review & editing, Y.C., D.L., K.L. and B.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used to support the findings of this study are included in the article.

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

References

1. Roberge, V.; Tarbouchi, M.; Labonté, G. Fast Genetic Algorithm Path Planner for Fixed-Wing Military UAV Using GPU. *IEEE Trans. Aerosp. Electron. Syst.* **2018**, *54*, 2105–2117.
2. Zhao, R.; Wang, Y.; Xiao, G.; Liu, C.; Hu, P.; Li, H. A method of path planning for unmanned aerial vehicle based on the hybrid of selfish herd optimizer and particle swarm optimizer. *Appl. Intell.* **2022**, *52*, 16775–16798.
3. 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.
4. Liu, Y. An optimization-driven dynamic vehicle routing algorithm for on-demand meal delivery using drones. *Comput. Oper. Res.* **2019**, *111*, 1–20.
5. Huang, H.; Hu, C.; Zhu, J.; Wu, M.; Malekian, R. Stochastic Task Scheduling in UAV-Based Intelligent On-Demand Meal Delivery System. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 13040–13054.
6. Bakirci, M. Smart city air quality management through leveraging drones for precision monitoring. *Sustain. Cities Soc.* **2024**, *106*, 105390.
7. Tahir, A. Formation Control of Swarms of Unmanned Aerial Vehicles. Ph.D. Thesis, University of Turku, Turku, Finland, 2023.
8. 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.
9. Sun, C.; Tang, J.; Zhang, X. FT-MSTC*: An Efficient Fault Tolerance Algorithm for Multi-robot Coverage Path Planning. In Proceedings of the 2021 IEEE International Conference on Real-time Computing and Robotics (RCAR), Xining, China, 15–19 July 2021; pp. 107–112. https://doi.org/10.1109/RCAR52367.2021.9517650.
10. Deng, Y.; Chen, Y.; Zhang, Y.; Mahadevan, S. Fuzzy Dijkstra algorithm for shortest path problem under uncertain environment. *Appl. Soft Comput.* **2012**, *12*, 1231–1237.
11. Cai, Y.; Xi, Q.; Xing, X.; Gui, H.; Liu, Q. Path planning for UAV tracking target based on improved A-star algorithm. In Proceedings of the 2019 1st International Conference on Industrial Artificial Intelligence (IAI), Shenyang, China, 23–27 July 2019; pp. 1–6.
12. Li, W.; Wang, L.; Zou, A.; Cai, J.; He, H.; Tan, T. Path Planning for UAV Based on Improved PRM. *Energies* **2022**, *15*, 7267.
13. Kelner, J.M.; Burzynski, W.; Stecz, W. Modeling UAV swarm flight trajectories using Rapidly-exploring Random Tree algorithm. *J. King Saud-Univ.-Comput. Inf. Sci.* **2024**, *36*, 101909.
14. Qu, C.; Gai, W.; Zhong, M.; Zhang, J. A novel reinforcement learning based grey wolf optimizer algorithm for unmanned aerial vehicles (UAVs) path planning. *Appl. Soft Comput.* **2020**, *89*, 106099.
15. Su, Y.; Dai, Y.; Liu, Y. A hybrid hyper-heuristic whale optimization algorithm for reusable launch vehicle reentry trajectory optimization. *Aerosp. Sci. Technol.* **2021**, *119*, 107200.
16. Chakraborty, S.; Sharma, S.; Saha, A.K.; Saha, A. A novel improved whale optimization algorithm to solve numerical optimization and real-world applications. *Artif. Intell. Rev.* **2022**, *55*, 1–112.
17. Mohapatra, P.; Nath Das, K.; Roy, S. A modified competitive swarm optimizer for large scale optimization problems. *Appl. Soft Comput.* **2017**, *59*, 340–362.
18. Nadimi-Shahraki, M.H.; Zamani, H.; Mirjalili, S. Enhanced whale optimization algorithm for medical feature selection: A COVID-19 case study. *Comput. Biol. Med.* **2022**, *148*, 105858.

19. Zhu, F.; Li, G.; Tang, H.; Li, Y.; Lv, X.; Wang, X. Dung beetle optimization algorithm based on quantum computing and multi-strategy fusion for solving engineering problems. *Expert Systems with Applications* **2024**, *236*, 121219.
20. Holland, J.H. Genetic Algorithms. *Sci. Am.* **1992**, *267*, 66–73.
21. Li, M.; Xu, G.; Zeng, L.; Lai, Q. Hybrid whale optimization algorithm based on symbiosis strategy for global optimization. *Appl. Intell.* **2023**, *53*, 16663–16705.
22. Li, M.; Xu, G.; Fu, Y.; Zhang, T.; Du, L. Improved whale optimization algorithm based on variable spiral position update strategy and adaptive inertia weight. *J. Intell. Fuzzy Syst.* **2022**, *42*, 1501–1517.
23. Storn, R.; Price, K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359.
24. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95 - International Conference on Neural Networks, Perth, WA, Australia, 27 November 27–1 December 1995; Volume 4, pp. 1942–1948.
25. Dorigo, M.; Birattari, M.; Stutzle, T. Ant colony optimization. *IEEE Comput. Intell. Mag.* **2006**, *1*, 28–39.
26. 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.
27. 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.
28. Phung, M.D.; Ha, Q.P. Motion-encoded particle swarm optimization for moving target search using UAVs. *Appl. Soft Comput.* **2020**, *97*, 106705.
29. Xu, X.; Xie, C.; Luo, Z.; Zhang, C.; Zhang, T. A multi-objective evolutionary algorithm based on dimension exploration and discrepancy evolution for UAV path planning problem. *Inf. Sci.* **2024**, *657*, 119977.
30. Phung, M.D.; Ha, Q.P. Safety-enhanced UAV path planning with spherical vector-based particle swarm optimization. *Appl. Soft Comput.* **2021**, *107*, 107376.
31. Wang, L.; Cao, Q.; Zhang, Z.; Mirjalili, S.; Zhao, W. Artificial rabbits optimization: A new bio-inspired meta-heuristic algorithm for solving engineering optimization problems. *Eng. Appl. Artif. Intell.* **2022**, *114*, 105082.
32. Alsaiari, A.O.; Moustafa, E.B.; Alhumade, H.; Abulkhair, H.; Elsheikh, A. A coupled artificial neural network with artificial rabbits optimizer for predicting water productivity of different designs of solar stills. *Adv. Eng. Softw.* **2023**, *175*, 103315.
33. Gülmez, B. Stock price prediction with optimized deep LSTM network with artificial rabbits optimization algorithm. *Expert Syst. Appl.* **2023**, *227*, 120346.
34. Yang, B.; Li, Y.; Huang, J.; Li, M.; Zheng, R.; Duan, J.; Fan, T.; Zou, H.; Liu, T.; Wang, J.; et al. Modular reconfiguration of hybrid PV-TEG systems via artificial rabbit algorithm: Modelling, design and HIL validation. *Appl. Energy* **2023**, *351*, 121868.
35. Dangi, D.; Telang Chandel, S.; Kumar Dixit, D.; Sharma, S.; Bhagat, A. An efficient model for sentiment analysis using artificial rabbits optimized vector functional link network. *Expert Syst. Appl.* **2023**, *225*, 119849.
36. Cao, Q.; Wang, L.; Zhao, W.; Yuan, Z.; Liu, A.; Gao, Y.; Ye, R. Vibration State Identification of Hydraulic Units Based on Improved Artificial Rabbits Optimization Algorithm. *Biomimetics* **2023**, *8*, 243.
37. Zhang, C.; Zhou, W.; Qin, W.; Tang, W. A novel UAV path planning approach: Heuristic crossing search and rescue optimization algorithm. *Expert Syst. Appl.* **2023**, *215*, 119243.
38. Yang, C.; Liu, Y.; Jiang, X.; Zhang, Z.; Wei, L.; Lai, T.; Chen, R. Non-Rigid Point Set Registration via Adaptive Weighted Objective Function. *IEEE Access* **2018**, *6*, 75947–75960. https://doi.org/10.1109/ACCESS.2018.2883689.
39. Tahir, A.; Haghbayan, H.; Böling, J.M.; Plosila, J. Energy-Efficient Post-Failure Reconfiguration of Swarms of Unmanned Aerial Vehicles. *IEEE Access* **2023**, *11*, 24768–24779. https://doi.org/10.1109/ACCESS.2022.3181244.
40. Zhao, L.; Shi, G. A method for simplifying ship trajectory based on improved Douglas–Peucker algorithm. *Ocean Eng.* **2018**, *166*, 37–46.
41. Faramarzi, A.; Heidarinejad, M.; Mirjalili, S.; Gandomi, A.H. Marine Predators Algorithm: A nature-inspired metaheuristic. *Expert Syst. Appl.* **2020**, *152*, 113377.
42. Huang, C.; Zhou, X.; Ran, X.; Wang, J.; Chen, H.; Deng, W. Adaptive cylinder vector particle swarm optimization with differential evolution for UAV path planning. *Eng. Appl. Artif. Intell.* **2023**, *121*, 105942.
43. Wang, W.; Ye, C.; Tian, J. SGGTSO: A Spherical Vector-Based Optimization Algorithm for 3D UAV Path Planning. *Drones* **2023**, *7*, 452.
44. Pan, Z.; Zhang, C.; Xia, Y.; Xiong, H.; Shao, X. An Improved Artificial Potential Field Method for Path Planning and Formation Control of the Multi-UAV Systems. *IEEE Trans. Circuits Syst. II Express Briefs* **2022**, *69*, 1129–1133.
45. Lin, K.; Li, Y.; Chen, S.; Li, D.; Wu, X. Motion Planner with Fixed-Horizon Constrained Reinforcement Learning for Complex Autonomous Driving Scenarios. *IEEE Trans. Intell. Veh.* **2024**, *9*, 1577–1588.
46. Lin, K.; Li, D.; Li, Y.; Chen, S.; Wu, X. FHCPL: An Intelligent Fixed-Horizon Constrained Policy Learning System for Risk-Sensitive Industrial Scenario. *IEEE Trans. Ind. Inform.* **2024**, *20*, 5794–5804.
47. Bai, X.; Xie, Z.; Xu, X.; Xiao, Y. An adaptive threshold fast DBSCAN algorithm with preserved trajectory feature points for vessel trajectory clustering. *Ocean Eng.* **2023**, *280*, 114930.
48. Tang, C.; Wang, H.; Zhao, J.; Tang, Y.; Yan, H.; Xiao, Y. A method for compressing AIS trajectory data based on the adaptive-threshold Douglas-Peucker algorithm. *Ocean Eng.* **2021**, *232*, 109041.

49. Awadallah, M.A.; Braik, M.S.; Al-Betar, M.A.; Abu Doush, I. An enhanced binary artificial rabbits optimization for feature selection in medical diagnosis. *Neural Comput. Appl.* **2023**, *35*, 20013–20068.
50. Luo, X.; Zou, H.; Hu, Y.; Gui, P.; Xu, Y.; Zhang, D.; Hu, W.; Hu, M. Synergistic registration of CT-MRI brain images and retinal images: A novel approach leveraging reinforcement learning and modified artificial rabbit optimization. *Neurocomputing* **2024**, *585*, 127506.
51. Hu, G.; Jing, W.; Houssein, E.H. Elite-based feedback boosted artificial rabbits-inspired optimizer with mutation and adaptive group: a case study of degree reduction for ball NURBS curves. *Soft Comput.* **2023**, *27*, 16919–16957.
52. Jiang, Y.; Wu, Q.; Zhu, S.; Zhang, L. Orca predation algorithm: A novel bio-inspired algorithm for global optimization problems. *Expert Syst. Appl.* **2022**, *188*, 116026.
53. Braik, M.; Hammouri, A.; Atwan, J.; Al-Betar, M.A.; Awadallah, M.A. White Shark Optimizer: A novel bio-inspired meta-heuristic algorithm for global optimization problems. *Knowl.-Based Syst.* **2022**, *243*, 108457.
54. Hu, G.; Huang, F.; Seyyedabbasi, A.; Wei, G. Enhanced multi-strategy bottlenose dolphin optimizer for UAVs path planning. *Appl. Math. Model.* **2024**, *130*, 243–271.
55. Douglas, D.H.; Peucker, T.K. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartogr. Int. J. Geogr. Inf. Geovisualization* **1973**, *10*, 112–122.
56. Abdel-Basset, M.; Mohamed, R.; Abouhawwash, M. Crested Porcupine Optimizer: A new nature-inspired metaheuristic. *Knowl.-Based Syst.* **2024**, *284*, 111257.
57. Parouha, R.P.; Das, K.N. A memory based differential evolution algorithm for unconstrained optimization. *Appl. Soft Comput.* **2016**, *38*, 501–517.
58. Opara, K.; Arabas, J. Comparison of mutation strategies in Differential Evolution—A probabilistic perspective. *Swarm Evol. Comput.* **2018**, *39*, 53–69.
59. Wu, G.; Mallipeddi, R.; Suganthan, P. *Problem Definitions and Evaluation Criteria for the CEC 2017 Competition and Special Session on Constrained Single Objective Real-Parameter Optimization*; Technology Report; Nanyang Technological University: Singapore, 2016; pp. 1–18.
60. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67.
61. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61.
62. Abdollahzadeh, B.; Gharehchopogh, F.S.; Mirjalili, S. African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems. *Comput. Ind. Eng.* **2021**, *158*, 107408.
63. Xue, J.; Shen, B. A novel swarm intelligence optimization approach: sparrow search algorithm. *Syst. Sci. Control. Eng.* **2020**, *8*, 22–34.
64. Xue, J.; Shen, B. Dung beetle optimizer: A new meta-heuristic algorithm for global optimization. *J. Supercomput.* **2023**, *79*, 7305–7336.
65. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H. Harris hawks optimization: Algorithm and applications. *Future Gener. Comput. Syst.* **2019**, *97*, 849–872.
66. Morales-Castañeda, B.; Zaldívar, D.; Cuevas, E.; Fausto, F.; Rodríguez, A. A better balance in metaheuristic algorithms: Does it exist? *Swarm Evol. Comput.* **2020**, *54*, 100671.
67. 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.
68. Demšar, J. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* **2006**, *7*, 1–30.
69. Dehghani, M.; Hubálovskỳ, Š.; Trojovskỳ, P. Northern goshawk optimization: A new swarm-based algorithm for solving optimization problems. *IEEE Access* **2021**, *9*, 162059–162080.
70. Zhong, C.; Li, G.; Meng, Z. Beluga whale optimization: A novel nature-inspired metaheuristic algorithm. *Knowl.-Based Syst.* **2022**, *251*, 109215.
71. Australia, G. *Digital Elevation Model (DEM) of Australia Derived from LiDAR 5 Metre Grid*; Commonwealth of Australia and Geoscience Australia: Canberra, Australia, 2015.
72. Wolpert, D.; Macready, W. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82.