Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

An Improved Beaver Behavior Optimizer for Three-Dimensional UAV Path Planning

Huanlong Zhang

zzuli407@163.com

Dian Jiao
Yingcong Wang
Qingjun Jia

Research Article

Keywords:

Posted Date: November 26th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11728123/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

An Improved Beaver Behavior Optimizer for Three-Dimensional UAV Path Planning

Huanlong Zhang[1*], Dian Jiao[1], Yingcong Wang[1], Qingjun Jia[1]

[1]College of Electric and Information Engineering, Zhengzhou University of Light Industry, Zhengzhou, 450000, Henan, China .

*Corresponding author(s). E-mail(s): zzuli407@163.com;
Contributing authors: 332501060065@zzuli.edu.cn;
ying_cong_wang@163.com; 2407142668@qq.com;

Abstract

This paper proposes an improved Beaver Behavior Optimizer (UBBO) for three-dimensional unmanned aerial vehicle (UAV) path planning to address the limitations of conventional swarm intelligence algorithms in search efficiency, convergence behavior, and path optimization. First, an adaptive phase factor based on population fitness distribution is introduced to dynamically regulate the transition between exploration and exploitation, while Lévy-flight perturbations are incorporated to enhance population diversity and improve the ability to escape local optima. Second, a target-oriented population update mechanism is developed by integrating destination guidance into the explorer position-update strategy, thereby reducing redundant searches and improving search directionality. Third, a UAV-specific path processing module is designed to reconstruct three-dimensional waypoint sequences, preserve predefined starting and terminal positions, and restrict intermediate flight altitudes within prescribed limits. Furthermore, a weighted-sum path-planning objective function is established by considering path length, trajectory smoothness, altitude constraints, and obstacle avoidance. The proposed UBBO algorithm is evaluated using selected CEC 2017 benchmark functions and simulated three-dimensional UAV path-planning scenarios, with comparisons against several representative optimization algorithms. The experimental results demonstrate competitive optimization accuracy and convergence performance on selected benchmark functions, while the generated UAV paths exhibit relatively favorable length and smoothness characteristics in the investigated environments. These findings indicate that the proposed method provides a promising optimization framework for three-dimensional UAV path planning.

Keywords: Beaver Behavior Optimizer; Unmanned Aerial Vehicle; Path Planning; Lévy Flight; CEC 2017 Benchmark Functions

1 Introduction

Unmanned aerial vehicles (UAVs) offer flexible deployment, maneuverability, and reduced human exposure to hazardous operating conditions, making them useful for power-line inspection, agricultural operations, emergency response, and logistics. Path planning is an essential component of autonomous UAV operation because it influences mission efficiency and flight safety. A suitable route can reduce travel distance and avoid obstacles and restricted regions while supporting reliable mission execution[5, 17, 21, 40].

In practical flight scenarios, UAV path planning must satisfy several requirements, including obstacle avoidance, flight-altitude limits, and trajectory smoothness. When a path is parameterized by multiple three-dimensional waypoints, the resulting task can be formulated as a high-dimensional nonlinear constrained optimization problem. Traditional methods, such as A* search, Dijkstra's algorithm, and artificial potential fields, have been widely used for path planning. However, their performance may be limited in complex three-dimensional environments: graph-search methods can incur substantial computational and memory costs as the search space grows, whereas artificial potential fields may become trapped in local minima and often require careful parameter tuning. These limitations have motivated the use of swarm-intelligence optimization algorithms. Such algorithms conduct global search by simulating the cooperative behaviors of biological swarms in nature, require no gradient information of objective functions, and exhibit strong adaptability to complicated optimization tasks. A variety of classical swarm intelligence algorithms have been developed to date and widely adopted for path planning applications, such as particle swarm optimization (PSO) inspired by bird flocking foraging, grey wolf optimizer (GWO) based on the hierarchical regime of grey wolves, whale optimization algorithm (WOA) mimicking whale predation behaviors, and Harris hawks optimization (HHO) drawing on the encirclement hunting strategy of Harris hawks. Newly proposed algorithms in recent years, including pelican optimization algorithm (POA), sparrow search algorithm (SSA) and secretary bird optimization algorithm, have also gained extensive applications in the field of path planning[1, 12, 14, 16, 18, 19, 22, 23, 27–29, 32, 33, 38, 39, 43, 47, 51, 56, 58].

The Beaver Behavior Optimizer (BBO) is a novel swarm intelligence optimization algorithm. Its core idea simulates the role division of beavers during dam construction: architects are responsible for design and construction, while explorers undertake material searching and transportation, and the two types of individuals complete the optimization process through cooperation. With few adjustable parameters and a clear framework, BBO exhibits favorable convergence performance in benchmark functions. Nevertheless, three limitations emerge when directly applying BBO to UAV path planning. First, the algorithm adopts a fixed sine factor for stage transition, which fails to dynamically balance exploration and exploitation according to real-time population states, resulting in insufficient early-stage exploration and slow late-stage convergence.

Second, lacking an effective global perturbation mechanism, BBO is prone to trapping in local optima due to the rapid decline of population diversity when solving multimodal and high-dimensional problems. Third, as a general optimization framework, the original BBO fails to fully exploit the target-oriented characteristics of path planning, leading to redundant searches. To tackle similar defects in swarm intelligence algorithms, numerous scholars have proposed various improvement strategies. Some studies introduce chaotic maps into population initialization to distribute initial individuals more evenly and accelerate the initial convergence rate. Other researchers adopt opposition-based learning and lens imaging learning to enrich population diversity and retard premature convergence. Additional works integrate random walk mechanisms such as Lévy flight and Brownian motion into position update formulas to strengthen the capability of escaping local optima. Furthermore, improvements tailored to specific application scenarios have attracted increasing attention, such as introducing target guidance terms and designing dedicated constraint handling modules for path planning, so that the algorithms can better meet practical engineering requirements[9, 13, 24, 26, 30, 31, 34, 35, 37, 41, 42, 45, 48–50, 53, 55, 57].

Although previous studies have improved swarm intelligence methods through adaptive search, random perturbation, and problem-specific guidance, further investigation of BBO for three-dimensional UAV path planning remains worthwhile. This paper develops an improved Beaver Behavior Optimizer (UBBO) by combining adaptive search control, target-oriented updating, and UAV-oriented path processing. First, a phase factor incorporating population fitness information is used to adjust exploration and exploitation, and Lévy-flight perturbations are introduced to diversify the search. Second, destination information is incorporated into the explorer update rule to provide additional directional guidance, while the architect–explorer update procedure is refined. Third, a path-processing module reconstructs three-dimensional waypoint sequences, retains the specified endpoints, and clips intermediate waypoint altitudes to a prescribed range. The proposed method is evaluated through selected CEC 2017 benchmark tests and simulated UAV path-planning experiments. These simulations assess optimization behavior and geometric path quality; practical flight deployment would require further feasibility and safety validation.

A closely related study by Wang et al. recently introduced IBBO-UAV for three-dimensional UAV path planning, combining adaptive exploration–exploitation control with obstacle-aware search and spline-based trajectory evaluation[54]. The present UBBO study shares the broad application domain and the use of an improved BBO framework, but focuses on a different combination of mechanisms: a fitness-distribution-based phase factor, Lévy-flight perturbation, terminal-point guidance for explorers, and waypoint-based altitude processing. Although both studies address three-dimensional UAV path planning using improved BBO frameworks, their technical formulations differ. The present work uses waypoint sequences and a cylindrical obstacle-threat cost, whereas the IBBO-UAV study uses spline-based trajectory generation and continuous obstacle-risk accumulation.

The main contributions of this study are summarized as follows:

(1) An adaptive global search enhancement strategy is proposed. The adaptive adjustment factor is constructed by calculating the relative relationships among the

optimal, average and worst fitness values of the population to replace the fixed sine stage factor in the original algorithm, enabling the search intensity to vary dynamically with the population convergence state. Meanwhile, the Lévy flight mechanism is introduced into the position update during the exploration phase. Relying on its random walk characteristic of alternating long and short steps, the global exploration capability and the ability to escape local optima of the algorithm are enhanced.

(2) A target-oriented population update mechanism is designed. To exploit the goal-directed nature of path-planning problems, a terminal-guidance term is added to the explorer update formula to direct individuals toward the destination. The explorer update strategy is reconstructed while retaining fitness-based architect selection, where explorers either learn from elite individuals with a certain probability or conduct searches combining cosine attenuation perturbation and target guidance, balancing search diversity and convergence efficiency simultaneously.

(3) A UAV-oriented path processing module is incorporated. Intermediate waypoint altitudes are clipped to a prescribed range, and solution vectors are reshaped into three-dimensional waypoint sequences with fixed start and end points. This module facilitates consistent evaluation of candidate paths under the geometric and altitude constraints considered in the simulations; dynamic flight feasibility remains beyond the scope of the present work.

The remainder of this paper is organized as follows. Section 2 introduces the fundamental principle and mathematical model of the original Beaver Behavior Optimizer. Section 3 elaborates on the three improvement strategies and overall implementation procedure of the proposed UBBO algorithm. Section 4 constructs the three-dimensional path planning model for UAVs. Section 5 verifies the algorithm performance via simulation experiments and analyzes the corresponding results. Section 6 concludes this paper and discusses directions for future research.

2 Basic Principles of the Beaver Behavior Optimizer

2.1 Biological Inspiration and Algorithm Idea

Beavers are remarkable "engineers" in nature. Through group cooperation, they can build structurally complex dams to reconstruct habitats and evade natural enemies. During dam construction, beaver colonies exhibit a distinct division of labor: a subset of experienced individuals undertakes the overall design and core construction of dams, while the remaining members search the surrounding areas for construction materials, such as branches, stones, and mud, and transport them to the construction sites. The two groups perform complementary roles and cooperate to build and maintain the dams[41].

The Beaver Behavior Optimizer (BBO) mathematically models this biological cooperative behavior. Within the algorithm framework, each beaver corresponds to a candidate solution in the solution space, the "material attributes" carried by beavers represent the dimensional components of solution vectors, and the construction quality of dams matches the fitness value of solutions. By simulating two core procedures of beavers, namely material collection (exploration phase) and dam reinforcement

(exploitation phase), the algorithm searches the solution space and gradually converges to the global optimal solution[41].

2.2 Population Initialization

At the initialization stage of the algorithm, an initial population is randomly generated within the search space. Let N denote the population size and D represent the dimension of the optimization problem; the population can be formulated as an $N \times D$ matrix:

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,D} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N,1} & x_{N,2} & \cdots & x_{N,D} \end{bmatrix} \tag{1}$$

Herein, the position of the i-th beaver in the j-th dimension is randomly generated by the following formula:

$$x_{i,j} = \mathrm{lb}_j + (\mathrm{ub}_j - \mathrm{lb}_j) \times \mathrm{rand} \tag{2}$$

where ub_j and lb_j denote the upper and lower bounds of the j-th dimensional search space, respectively, and rand represents a random number uniformly distributed over $[0, 1]$.

2.3 Dam-Phase Adjustment Factor

To achieve a smooth transition between exploration and exploitation, BBO introduces a Dam-Phase Factor to dynamically adjust the relative weights of the two phases. This factor increases monotonically with the number of iterations, enabling the algorithm to prioritize global exploration at the early stage and gradually shift toward local exploitation as iterations proceed [41].

$$D = \sin\left(\frac{\pi \cdot t}{2 \cdot T_{\max}}\right) \tag{3}$$

where t denotes the current iteration and $T_{\max}$ denotes the maximum number of iterations. The value of D ranges from 0 to 1: when $t = 0$, $D = 0$ and the algorithm fully enters the exploration phase; when $t = T_{\max}$, $D = 1$, and the algorithm switches entirely to the exploitation phase.

In each iteration, a random number r_1 is generated. If $r_1 \leq D$, the exploitation phase is executed; otherwise, the exploration phase is activated. As iterations progress, D increases gradually, raising the probability of entering the exploitation phase and realizing a natural transition from exploration to exploitation.

2.4 Exploration Phase: Material Collection

During the exploration phase, the beaver colony focuses on searching for construction materials over extensive areas, which corresponds to the algorithm conducting

broad searches across the solution space to identify potential high-quality regions. The population is divided into two roles in this phase: architects and explorers.

2.4.1 Architect Behavior

Architects are the top 25% of individuals ranked by fitness and represent the high-quality solution regions identified so far. During the exploration phase, information exchange occurs among architects. Each architect randomly selects another architect as the learning target with a probability of 50%, and optimizes its own solution quality by referencing the counterpart's "material attributes"[41].

The position update formula of architects is given as follows:

$$x_{i,j}(t+1) = x_{i,j}(t) + \mathbb{I}(r_2 < 0.5) \cdot r_3 \cdot \left(x_{k,j}(t) - x_{i,j}(t)\right) \tag{4}$$

where $x_{i,j}(t)$ denotes the position of the i-th architect in the j-th dimension at the t-th iteration, k represents the index of another architect randomly selected from the architect group, and $\mathbb{I}(\bullet)$ refers to the indicator function that equals 1 if the condition inside the parentheses holds and 0 otherwise. Both r_2 and r_3 are random numbers uniformly distributed within $[0, 1]$.

Through mutual learning among architects, high-quality information within the elite population is shared, facilitating refined searches around promising solution regions.

2.4.2 Explorer Behavior

Explorers make up the majority of the population and are responsible for searching a wider area for new material sources, corresponding to global exploration in the solution space. Their position-update rule contains two components: probabilistic learning from architects to exploit elite information, and random perturbations around the current position to maintain population diversity[41].

The position update formula for explorers is expressed as:

$$\begin{aligned} x_{i,j}(t+1) = x_{i,j}(t) &+ \mathbb{I}(r_4 < 0.5) \cdot r_5 \cdot \left(x_{arc,j}(t) - x_{i,j}(t)\right) \\ &+ \frac{r_6 \cdot \cos\left(\dfrac{\pi t}{2T_{\max}}\right)}{10} \cdot \left(ub_j - lb_j\right) \end{aligned} \tag{5}$$

where $x_{arc,j}(t)$ denotes the j-dimensional position of a randomly selected architect; r_4 and r_5 are random numbers uniformly distributed over $[0, 1]$; and r_6 is a random number that obeys the standard normal distribution. The first term on the right hand side of the formula corresponds to the learning component from architects, while the second term represents the cosine decay random perturbation. The perturbation amplitude gradually decreases with the increase in iterations, which matches the overall algorithm trend shifting from exploration to exploitation.

2.5 Exploitation Phase: Dam Maintenance and Reinforcement

After the beaver colony collects sufficient materials, it proceeds to the dam maintenance and reinforcement phase, which corresponds to the exploitation stage of the algorithm. During this stage, all individuals take part in the refined construction of the dam, and the division between architects and explorers is eliminated. Each beaver learns from the best individual in the population or from other randomly selected members, thereby refining its solution through information from multiple sources[41].

The position update formula for the exploitation stage is given as:

$$x_{i,j}(t+1) = x_{i,j}(t) + r_7 \cdot \big(x_{k,j}(t) - x_{i,j}(t)\big) + r_8 \cdot \big(x_{\text{best},j}(t) - x_{i,j}(t)\big) \tag{6}$$

where $x_{\text{best},j}(t)$ stands for the j-dimensional position of the current optimal individual in the population; k denotes the index of a randomly selected individual from the population; r_7 and r_8 are random numbers uniformly distributed in the range $[0, 1]$. The first term on the right hand side of the formula represents the learning from a random individual, and the second term indicates the learning from the global best individual. The combination of the two terms enables the algorithm to carry out fine-grained searching around promising regions and gradually approach the optimal solution.

2.6 Overall Algorithm Flow

Combined with the above-mentioned mechanisms, the flow chart of the original BBO algorithm is shown in Figure 1, and its implementation procedures are summarized as follows:

Step 1: Set parameters including the population size N, the maximum number of iterations $T_{\max}$, and the upper and lower bounds of the search space. Randomly initialize the initial population within the solution space.

Step 2: Calculate the fitness value of each individual in the population, and record the current global optimal solution together with its fitness.

Step 3: Compute the dam-phase factor D for the current iteration.

Step 4: Generate a random number r_1 and compare it with D to determine whether to enter the exploration or exploitation stage.

Step 5: For the exploration stage, sort the population by fitness, select the top 25% as architects and the rest as explorers. Update the positions of the two groups of individuals according to their respective formulas.

Step 6: For the exploitation stage, update the positions of all individuals using the position update formula for the exploitation phase.

Step 7: Conduct boundary inspection and out-of-boundary treatment for updated individuals to guarantee that all individuals stay within the search space range.

Step 8: Recalculate the fitness of all individuals and renew the global optimal solution.

Step 9: Judge whether the maximum iteration number is reached. If not, return to Step 3 for subsequent iterations; otherwise, output the global optimal solution and its optimal fitness, and terminate the algorithm.

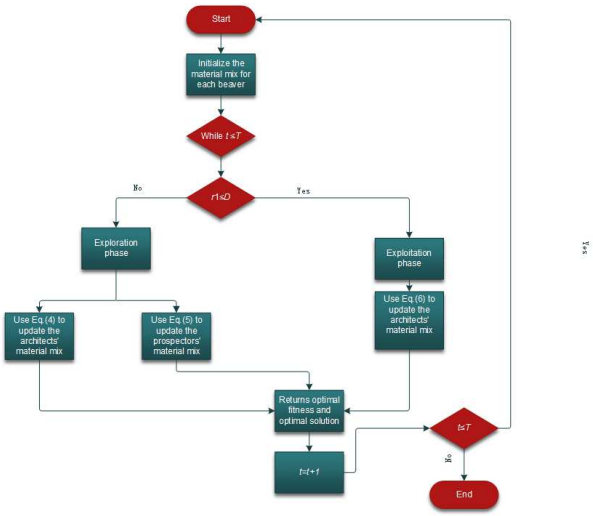

Fig. 1: Flowchart of the original BBO algorithm

3 Improved Beaver Behavior Optimizer (UBBO)

This section presents the proposed UBBO method for three-dimensional UAV path planning. Three modifications to the original BBO are considered. First, an adaptive phase factor based on population fitness information and Lévy-flight perturbations are introduced to adjust the search process and maintain diversity. Second, destination information is incorporated into the explorer update rule, while the architect–explorer search procedure is refined. Third, a UAV-oriented path-processing module is used to reconstruct waypoint sequences, retain fixed endpoints, and limit intermediate waypoint altitudes. These components are intended to improve search efficiency and provide consistently formatted paths for evaluation in the simulated environment.

3.1 Adaptive Global Search Enhancement Strategy

3.1.1 Problem Analysis

The original BBO uses a sinusoidal phase factor to switch between exploration and exploitation. Because this factor depends only on the iteration count, it cannot directly reflect the current fitness distribution of the population. Consequently, the timing of exploration and exploitation may not always match the search state, which motivates an adaptive adjustment in the proposed method.

3.1.2 Adaptive Stage Factor

To tackle the above-mentioned problem, this paper introduces the distribution information of population fitness and constructs an adaptive phase factor, so that the intensities of exploration and exploitation can be dynamically adjusted according to the real-time population state.

First, the relative-concentration index φ of population fitness is defined as:

$$\varphi = \frac{f_{\text{best}} - f_{\text{avg}}}{f_{\text{best}} - f_{\text{worst}} + \varepsilon} \tag{7}$$

where f_{best}, f_{avg} and f_{worst} represent the best, average and worst fitness values of the current-generation population, respectively; ε denotes a tiny constant set to 10^{-20} to prevent the denominator from being zero.

φ lies between 0 and 1 for a minimization problem with $f_{\text{best}} \leq f_{\text{avg}} \leq f_{\text{worst}}$. A large value indicates that the average fitness is relatively close to the worst fitness within the current fitness range, whereas a small value indicates that the average fitness is relatively close to the best fitness. Thus, φ describes the relative position of the population-average fitness within the best-to-worst fitness interval; it should not be interpreted as a direct geometric measure of population dispersion.

On this basis, the original fixed dam-phase factor is improved to the adaptive phase factor E:

$$E = \sin\left(\frac{\pi \cdot t}{2 \cdot T_{\max}}\right) \cdot (1 - \varphi) \tag{8}$$

It can be observed that the adaptive factor E is jointly determined by two components. The sinusoidal term that increases with iteration numbers retains the overall exploration-to-exploitation trend of the original algorithm, while the φ term related to population concentration realizes the dynamic adjustment of search intensity. When the population-average fitness is relatively close to the worst fitness, φ is large and E remains small, favoring exploration. When the average fitness is relatively close to the best fitness, φ is small and E increases, raising the probability of exploitation. This mechanism adjusts the phase-selection probability according to the relative position of the average fitness within the current fitness range and may help mitigate an inappropriate transition between exploration and exploitation. The schematic diagram for the regulation logic of the adaptive phase factor is illustrated in Figure 2.

3.1.3 Lévy Flight Global Perturbation

To further enhance the algorithm's capability to escape from local optima, a Lévy-flight random-walk mechanism is introduced into the position update during the exploration stage. As a type of random walk subject to the Lévy distribution, Lévy flight alternates between short-range meticulous searches and occasional long-distance jumps. These step-size characteristics are consistent with the search patterns observed in many natural foraging processes [35, 42, 53, 57].

In optimization algorithms, the long-jump property of Lévy flight enables individuals to break away from the constraints of the current search zone and explore far-off unknown regions within the solution space, which may help the algorithm escape from local optima. Meanwhile, short-step searches support fine-scale investigation around promising regions.

The calculation formula for the step size of Lévy flight is given as:

$$\text{Levy}(u) = 0.01 \cdot \frac{u}{|v|^{1/\beta}} \tag{9}$$

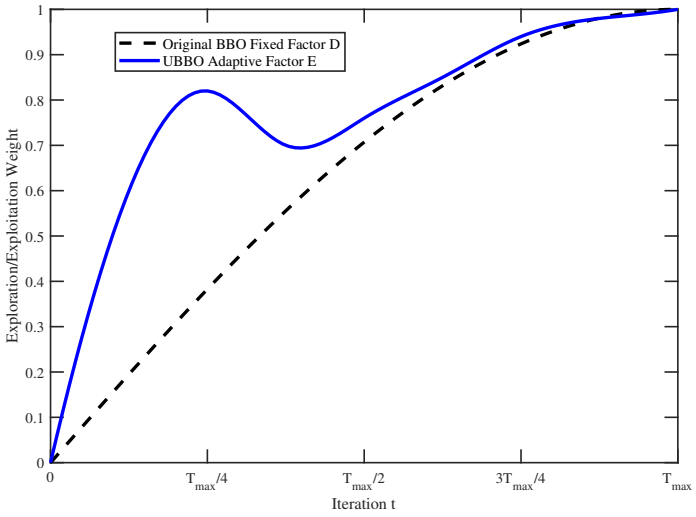

Fig. 2: Schematic diagram of the regulation logic for the adaptive phase factor

where β stands for the Lévy index, which is generally set to 1.5. Both u and v obey the normal distribution, namely $u \sim N(0, \sigma^2)$ and $v \sim N(0, 1)$. The computational formula for parameter σ is expressed as:

$$\sigma = \left[\frac{\Gamma(1+\beta) \cdot \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \cdot \beta \cdot 2^{\frac{\beta-1}{2}}} \right]^{1/\beta} \tag{10}$$

where Γ denotes the gamma function.

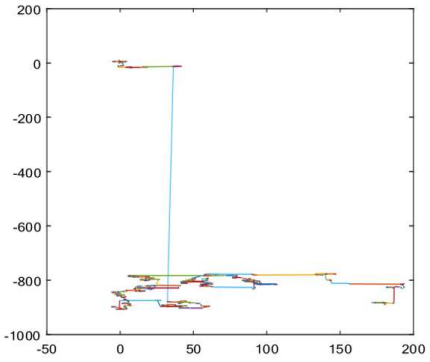

Fig. 3: Lévy-flight trajectory

Lévy-flight perturbation is incorporated into the second explorer-update mode in the exploration stage. When this mode is selected, the explorer adds a randomly oriented Lévy step to its position update, thereby broadening the search scope and potentially strengthening global exploration. During specific implementation, the amplitude of Lévy perturbation is proportional to the range of the search space, which ensures that perturbations are carried out within a reasonable scale[35, 42, 53, 57].

3.2 Target-Oriented Population Update Mechanism

3.2.1 Problem Analysis

The original BBO algorithm serves as a general-purpose optimization framework. Its position-update process entirely relies on information interaction inside the population without introducing any problem-specific prior knowledge. Nevertheless, the UAV path-planning problem possesses explicit goal-oriented characteristics, where the destination of the flight path is known and all waypoints ought to extend toward the target point. Deprived of the guidance from target information, the original algorithm generates substantial redundancy during the search. Individuals may conduct useless exploration in areas far away from the destination, which impairs the convergence speed and final path quality.

Besides, the original algorithm divides architects and explorers at a fixed proportion of 25%, and both groups adopt relatively simplistic update strategies. Architects merely carry out basic mutual-learning within the elite population, leading to insufficient depth of information interaction. Although the update rules for explorers contain learning and perturbation operations, the perturbation lacks clear directionality and learners are selected in a completely random manner. Consequently, the guiding value of elite individuals cannot be fully exploited.

3.2.2 Target-Point Guidance Mechanism

A terminal-guidance term is incorporated into explorer updates to provide an additional search direction. Because a candidate solution represents a sequence of intermediate waypoints rather than the physical position of a UAV, this term does not by itself ensure collision avoidance or path feasibility. Its effect is considered together with the objective function and the other search mechanisms.

Specifically, a terminal-guidance term is added to the position update of explorers:

$$x_{i,j}(t+1) = x_{i,j}(t) + 0.5 \cdot \left(G_{\mathrm{map}(j)} - x_{i,j}(t)\right) \tag{11}$$

where G represents the terminal-point coordinate vector consisting of three-dimensional x, y and z components. For the interleaved waypoint encoding $\mathbf{x} = [x_1, y_1, z_1, x_2, y_2, z_2, \ldots]$, $\mathrm{map}(j) = 1 + \mathrm{mod}(j-1, 3)$ selects the corresponding component of the terminal vector $G = (G_x, G_y, G_z)$. Here, $x_{i,j}$ denotes the j-th component of individual i.

The guidance term is determined by the difference between the destination coordinate and the corresponding coordinate component of the candidate solution. It can

therefore bias intermediate waypoint coordinates toward the destination. The coefficient of 0.5 sets the strength of this bias. By incorporating terminal-point information, the mechanism is intended to reduce redundant search and improve convergence efficiency; however, its overall effect depends on the path encoding, obstacle configuration, and other optimization terms.

It should be noted that the target-guidance term provides directional information during the optimization process rather than directly determining the flight path. Its effectiveness is evaluated together with the obstacle-avoidance and smoothness terms in the comprehensive objective function.

3.2.3 Improved Architect–Explorer Update Strategy

This paper optimizes the architect–explorer mechanism of the original algorithm by focusing on the explorer position-update rules, while retaining the fitness-based architect selection used in the original BBO.

Fitness-Based Architect Selection (Retained from BBO)

In each generation, individuals are sorted in ascending order of fitness, and the top 25% are selected as architects, following the selection rule described for the original BBO in Section 2. This selection is retained as part of the implementation and is not claimed as an independent improvement. The changes examined here focus on the explorer update mechanism described below.

Reconstruction of the Dual-Update Strategy for Explorers

This paper redesigns the position-update logic for explorers and defines two update modes, each activated with a 50% probability.

Mode 1: Learning from architects. Each explorer randomly selects one architect as the learning reference and moves toward the elite individual via differential learning to acquire high-quality solution information:

$$x_{i,j}(t+1) = x_{i,j}(t) + r \cdot \left(x_{\mathrm{arc},j}(t) - x_{i,j}(t)\right) \tag{12}$$

where $x_{\mathrm{arc},j}$ denotes the j-dimensional position of the randomly selected architect, and r is a random number within the range $[0, 1]$.

Mode 2: Cosine-decay perturbation combined with target guidance. On the basis of the current position, explorers superimpose random perturbation with cosine attenuation and introduce the target-point guidance term, which maintains search diversity and guarantees the direction of convergence simultaneously:

$$\begin{aligned} x_{i,j}(t+1) = x_{i,j}(t) + \frac{r \cdot \cos\left(\frac{\pi t}{2T_{\max}}\right)}{10} \cdot (ub_j - lb_j) \\ + 0.5 \cdot \left(G_{\mathrm{map}(j)} - x_{i,j}(t)\right) + \mathrm{Levy}(u) \end{aligned} \tag{13}$$

where r is a standard normally distributed random number, and $\mathrm{Levy}(u)$ denotes the j-th component of the Lévy perturbation vector. The cosine term enables the

perturbation amplitude to decrease gradually with iterations, matching the overall algorithm transition from exploration to exploitation. Meanwhile, the target-guidance term provides a directional bias toward the destination, while the random and Lévy perturbations retain stochastic variation.

The fitness-based selection of architects is retained from the original BBO, while the two explorer-update modes introduce alternative search directions. Elite learning provides information from promising individuals, whereas the second mode combines cosine-decay perturbation, destination guidance, and Lévy perturbation. Their combined effect is evaluated experimentally and should not be assumed to improve every problem instance.

3.3 UAV-Specific Constraint Processing Module

3.3.1 Problem Analysis

The original BBO algorithm outputs a general-purpose high-dimensional vector, which cannot be directly converted into a feasible UAV flight path. In practical engineering, UAV flights are subject to multiple constraints. For instance, the flight altitude should stay within a safe range, the flight route needs to start from the preset starting point and terminate at the destination, and the sequence of waypoints must comply with the geometric logic of three-dimensional space. Directly adopting the algorithm-generated vector as the flight path may trigger several defects including altitude overflow, disconnected endpoints and disordered dimension sequence, and the obtained trajectory cannot be applied to actual flight missions[7, 10, 36].

3.3.2 Constraint Handling Mechanism

To standardize the solution vectors produced by UBBO for the path-planning simulations, a UAV-oriented processing procedure is introduced. It reconstructs waypoint coordinates, inserts the prescribed start and end points, and restricts intermediate waypoint altitudes to the predefined range. The procedure consists of the following three steps.

Path Dimension Reconstruction

The one-dimensional decision vector generated by the algorithm is reshaped into a three-dimensional waypoint matrix. Let $\mathbf{x} = [x_1, x_2, \ldots, x_D]$, where $D = 3M$ and each consecutive triplet encodes the x-, y-, and z-coordinates of one intermediate waypoint. The resulting matrix contains $M = D/3$ waypoints:

$$\mathbf{P} = \begin{bmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ \vdots & \vdots & \vdots \\ x_{D-2} & x_{D-1} & x_D \end{bmatrix} \tag{14}$$

Each row of $\mathbf{P}$ gives the three-dimensional coordinates of one intermediate waypoint.

Start-and-End-Point Concatenation

The start-point S and destination G are concatenated to the head and tail of the waypoint matrix separately to construct the complete flight trajectory:

$$\text{Trajectory} = [\, S;\ \mathbf{P};\ G\,] \tag{15}$$

This step ensures that the specified start and end points are retained in the assembled waypoint sequence. It does not, by itself, establish collision-free or dynamically feasible flight.

Altitude Constraint Clipping

The flight altitude of intermediate waypoints (all waypoints excluding the start and end points) is clipped within the preset interval to restrict the altitude to the safe height range:

$$z_k = \max\left(z_{\min},\ \min(z_k,\ z_{\max})\right), \quad k = 2, 3, \ldots, L-1 \tag{16}$$

where L denotes the total number of waypoints, $z_{\min}$ and $z_{\max}$ represent the lower and upper bounds of the safe-flight altitude. According to the practical flight demands of the unmanned aerial vehicle in this research, $z_{\min} = 20\,\text{m}$ and $z_{\max} = 40\,\text{m}$.

After these operations, the intermediate waypoints are converted back into a one-dimensional solution vector for subsequent optimization iterations. This module provides a consistent path representation, retains the specified endpoints, and enforces the prescribed altitude range for intermediate waypoints. These operations facilitate candidate-path evaluation in the simulated environment but do not, by themselves, establish collision-free or dynamically feasible flight.

The altitude-clipping operation is applied before fitness evaluation to ensure that intermediate waypoints remain within the prescribed altitude range. The altitude component F_3 in Section 4.3 nevertheless remains active because it penalizes deviations from the midpoint of the permitted altitude interval, even when a waypoint satisfies the altitude bounds.

3.4 Overall Flow of UBBO Algorithm

Combined with the three aforementioned improvement strategies, the flow chart of the UBBO algorithm is shown in Figure 4. The overall implementation procedures are described as follows:

Step 1: Configure parameters including the population size N, maximum iteration number $T_{\max}$, upper and lower bounds of the search space, start point S, destination point G and safe altitude range.

Step 2: Randomly initialize the population within the search space, calculate the fitness value of each individual, and record the current global optimal solution and its corresponding optimal fitness.

Step 3: Work out the best, average and worst fitness values of the current population, and further compute the population concentration φ and adaptive phase factor E.

Step 4: Generate a random number and compare it with the adaptive factor E to determine whether to enter the exploration or exploitation phase.

Step 5: For the exploration phase, sort all individuals in ascending order of fitness and select the top 25% as architects while the rest are defined as explorers. The architects update their positions according to the corresponding formula. Each explorer updates its position with a 50% probability by learning from architects, or adopts the cosine-perturbation-and-target-guidance update rule integrated with Lévy-flight disturbance at the remaining 50% probability.

Step 6: For the exploitation phase, all individuals perform position updates with the exploitation-phase formula.

Step 7: Carry out boundary inspection and out-of-bound clipping for updated individuals.

Step 8: Invoke the dedicated UAV constraint-handling function to complete altitude restriction, endpoint concatenation and format standardization for trajectories.

Step 9: Recalculate the fitness of all individuals and renew the global optimal solution.

Step 10: Record convergence-curve data of the current iteration.

Step 11: Judge whether the maximum iteration number is satisfied. If not, return to Step 3 for subsequent iterations; otherwise, output the global optimal solution, optimal fitness and convergence curve, and terminate the algorithm.

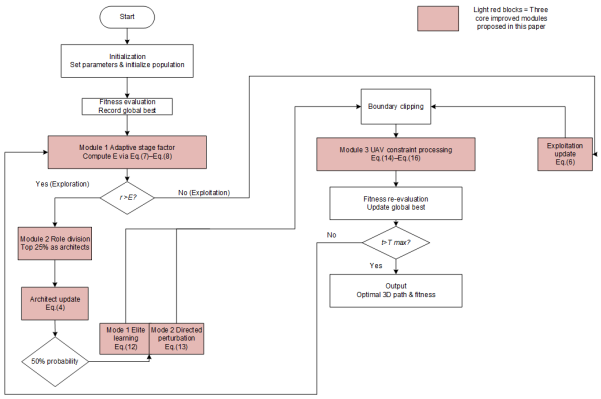

Fig. 4: Flowchart of the UBBO algorithm

4 UAV Path Planning

The goal of UAV path planning is to determine a route from a prescribed start point to a destination while accounting for path length, obstacle clearance, altitude limits, and geometric smoothness. In this study, these criteria are combined through a weighted-sum objective function to evaluate candidate waypoint sequences. The resulting scalar objective provides a trade-off among the considered criteria, following established UAV

path-planning and trajectory-smoothing formulations[2, 15]; it does not, by itself, guarantee dynamic or flight-control feasibility.

4.1 Path-Length Cost

Let the i-th candidate path be represented by an ordered sequence of n three-dimensional waypoints, including the prescribed start and end points. The j-th waypoint is denoted by $\mathbf{P}_{ij} = (X_{ij}, Y_{ij}, Z_{ij})$. The Euclidean length of the segment connecting two consecutive waypoints is $\|\mathbf{P}_{i,j+1} - \mathbf{P}_{ij}\|$. The total path-length cost is therefore:

$$F_1(X_i) = \sum_{j=1}^{n-1} \|\mathbf{P}_{i,j+1} - \mathbf{P}_{ij}\| \tag{17}$$

4.2 Obstacle-Threat Cost

Let $\mathcal{K}$ denote the set of cylindrical obstacles and let $K = |\mathcal{K}|$ be the number of obstacles. The horizontal projection of obstacle k has center C_k and radius R_k. The UAV is modeled using an effective horizontal radius r_{UAV}, and S denotes an additional safety margin. For each path segment, let d_{ijk} denote the minimum horizontal distance between the projection of that segment and the projected center C_k of obstacle k. The obstacle-threat cost is defined by penalizing segments that enter the safety region around an obstacle, consistent with risk-aware UAV path-planning formulations[44]:

$$F_2(X_i) = \sum_{j=1}^{n-1} \sum_{k=1}^{K} T_{ijk} \tag{18}$$

where

$$T_{ijk} = \begin{cases} 0, & d_{ijk} > S + r_{\mathrm{UAV}} + R_k, \\ (S + r_{\mathrm{UAV}} + R_k) - d_{ijk}, & r_{\mathrm{UAV}} + R_k < d_{ijk} \le S + r_{\mathrm{UAV}} + R_k, \\ \infty, & d_{ijk} \le r_{\mathrm{UAV}} + R_k. \end{cases} \tag{19}$$

Here, d_{ijk} must be computed consistently with the cylindrical obstacle geometry and the path-segment representation used in the implementation. In particular, the use of horizontal distance is appropriate only when the obstacle model treats the cylinder as vertically extending through the relevant flight-altitude range.

4.3 Flight-Altitude Constraint

During operation, the altitude of each intermediate waypoint is restricted to the interval $[h_{\min}, h_{\max}]$. In keeping with the waypoint-processing procedure in Section 3.3, the fixed start and end points are excluded from this intermediate-waypoint altitude penalty. The penalty for an intermediate waypoint is defined relative to the midpoint

of the permitted altitude interval:

$$H_{ij} = \begin{cases} \left| h_{ij} - \dfrac{h_{\max} + h_{\min}}{2} \right|, & h_{\min} \le h_{ij} \le h_{\max}, \\ \infty, & \text{otherwise.} \end{cases} \tag{20}$$

Here, h_{ij} is the altitude of waypoint j relative to the local ground level. Values outside the prescribed interval receive an infinite penalty. The total altitude cost is summed over the intermediate waypoints:

$$F_3(X_i) = \sum_{j=2}^{n-1} H_{ij} \tag{21}$$

4.4 Path-Smoothness Cost

The smoothness cost evaluates changes in the horizontal turning direction and in the climb angle along the waypoint sequence. Let $\mathbf{v}'_{ij}$ denote the horizontal projection of the segment from waypoint j to waypoint $j+1$:

$$\mathbf{v}'_{ij} = \begin{bmatrix} X_{i,j+1} - X_{ij} \\ Y_{i,j+1} - Y_{ij} \\ 0 \end{bmatrix}. \tag{22}$$

The horizontal turning angle between two consecutive projected segments is calculated as:

$$\phi_{ij} = \operatorname{atan2}\left(\left\| \mathbf{v}'_{ij} \times \mathbf{v}'_{i,j+1} \right\|, \mathbf{v}'_{ij} \cdot \mathbf{v}'_{i,j+1} \right), \quad j = 1, \ldots, n-2. \tag{23}$$

The climb angle Ψ_{ij} is the angle between the three-dimensional path segment and its horizontal projection:

$$\Psi_{ij} = \operatorname{atan2}\left(Z_{i,j+1} - Z_{ij},\ \left\| \mathbf{v}'_{ij} \right\| \right), \quad j = 1, \ldots, n-1. \tag{24}$$

The smoothness cost is then defined as:

$$F_4(X_i) = a_1 \sum_{j=1}^{n-2} \phi_{ij} + a_2 \sum_{j=2}^{n-1} \left| \Psi_{ij} - \Psi_{i,j-1} \right| \tag{25}$$

where a_1 and a_2 are the weighting coefficients for the horizontal turning angle and the change in climb angle, respectively.

4.5 Composite Objective Function

The four cost components are combined into a scalar objective function:

$$F(X_i) = \sum_{k=1}^{4} b_k F_k(X_i) \tag{26}$$

where $b_k \geq 0$ is the weighting coefficient assigned to the k-th cost component.

5 Experimental Results and Analysis

To evaluate the proposed UBBO algorithm, comparative experiments are conducted in two settings. First, the CEC 2017 benchmark function test suite is adopted for numerical-optimization experiments, so as to verify the convergence speed, optimization accuracy and robustness of the algorithm from the perspective of general-purpose optimization capability. Second, the algorithm is applied to the three-dimensional unmanned-aerial-vehicle path-planning scenario. Comparisons with multiple intelligent optimization algorithms are conducted to assess its performance in the simulated path-planning problem.

5.1 Experimental Setup

5.1.1 Experimental Environment

All simulation experiments were conducted in the same hardware and software environment to ensure a fair comparison. The experimental platform is configured as follows: the operating system is Windows 10, the processor is an Intel Core i7-10700K CPU running at 3.80 GHz, the RAM is 16 GB, and all algorithms are coded in MATLAB R2021b.

5.1.2 Comparison Algorithms and Parameter Settings

For the CEC 2017 benchmark tests, UBBO is compared with six representative algorithms: the original Beaver Behavior Optimizer (BBO), particle swarm optimization (PSO), whale optimization algorithm (WOA), Harris hawks optimization (HHO), pelican optimization algorithm (POA), and northern goshawk optimization (NGO). A separate set of comparison algorithms is used for the UAV path-planning experiment, as specified in Section 5.3[11, 16, 23, 38, 39, 41, 51].

For a fair comparison, the population size was set to 30 and the maximum number of iterations to 500 for all algorithms. Each algorithm was run independently 30 times. The minimum value (Min), mean value (Avg), and standard deviation (Std) are adopted as evaluation metrics. The minimum value reflects the search accuracy, the average value represents the overall optimization capacity, and the standard deviation illustrates algorithm stability and robustness. The remaining parameters of each algorithm were configured according to the recommended values in the original literature.

5.2 CEC 2017 Benchmark Tests

5.2.1 Description of the Benchmark Functions

The CEC 2017 benchmark suite originally defines 30 single-objective test functions. Function F2 is commonly excluded from comparative studies because of reported numerical instability, leaving 29 functions for evaluation. The remaining functions

are grouped into unimodal functions (F1 and F3), simple multimodal functions (F4–F10), hybrid functions (F11–F20), and composition functions (F21–F30). These groups evaluate different aspects of an optimizer, including local exploitation, escape from local optima, and performance on more complex composite landscapes[4].

In this study, the numerical results cover 27 selected CEC 2017 benchmark functions: F1, F3–F19, and F22–F30. Functions F2, F20, and F21 are not included in the supplied result tables. The convergence plots are discussed separately in Section 5.2.4.

5.2.2 Test Results and Analysis

The statistical results of the seven compared algorithms on the selected CEC 2017 benchmark functions are presented in Tables 1–3. Each table reports the minimum value (Min), standard deviation (Std), and average value (Avg) for the corresponding test functions. The reported statistics were calculated from 30 independent runs for each algorithm.

The tables provide a function-by-function comparison of the seven algorithms. Min represents the best objective value obtained in the repeated runs, Avg represents the arithmetic mean of the obtained values, and Std describes the variation among runs. The relative performance differs across benchmark functions; therefore, the results are presented separately for each function rather than assuming that one algorithm is uniformly superior on all test problems.

5.2.3 Statistical Significance Analysis

To assess whether the performance differences between UBBO and the comparison algorithms are statistically significant, the Wilcoxon rank-sum test was applied to the results of the independent runs for the 27 selected CEC 2017 functions. The significance level was set to $\alpha = 0.05$. Because multiple hypotheses are tested, the Holm step-down procedure was applied separately to the 27 p-values associated with each comparison algorithm to control the family-wise error rate within each comparison family [3, 20]. Specifically, for the ordered raw p-values $p_{(1)} \leq \cdots \leq p_{(m)}$ (where $m = 27$), the adjusted values are calculated as

$$p_{(i)}^{\mathrm{Holm}} = \min\left\{1, \max_{1\leq j\leq i}\left[(m-j+1)p_{(j)}\right]\right\}, \quad i = 1, \ldots, m. \tag{27}$$

The adjusted values are then mapped back to their corresponding benchmark functions. A comparison is considered statistically significant only when its Holm-adjusted p-value is below 0.05. The adjusted p-values are reported in Table 4, and the asterisk denotes statistical significance after Holm correction.

After Holm correction, statistically significant differences are observed on 22, 27, 4, 27, and 27 functions for WOA, POA, NGO, HHO, and BBO, respectively. In particular, the number of significant WOA comparisons decreases from 23 to 22, while the number for NGO decreases from 11 to 4. This illustrates why multiple-comparison correction is important when interpreting a large set of pairwise tests. A statistically significant result indicates a difference between the two result distributions but does

Table 1: Test results of the compared algorithms on CEC 2017 benchmark functions (F1–F10)

Function	Metric	WOA	PSO	POA	NGO	HHO	BBO	UBBO
F1	Min	6894808.048	100.6291979	21495.84192	100.0048125	287162.4481	369.0994859	853709168.2
F1	Std	114477431.4	352614991.1	430605364.8	1513.1505	1116661.628	13726429.38	3699574962
F1	Avg	73944504.42	93670206.45	201927377.3	1596.037633	1298024.74	4089765.787	6322302094
F3	Min	1040.542151	300	314.0631383	300.0023593	309.5342297	300.0151617	4958.256008
F3	Std	7411.512338	1.54323E-07	1268.082615	6.51542523	371.7688236	702.777107	2694.59229
F3	Avg	9291.595665	300.0000001	1281.128485	303.7938481	689.1923579	822.1882538	9107.86234
F4	Min	404.835944	400.0051551	401.1495917	400.0431646	400.5372991	401.9112214	584.7883637
F4	Std	66.22344127	3.871578034	44.5060945	1.375844798	33.49310775	43.68023562	468.5458017
F4	Avg	460.9846261	404.8665519	428.072608	402.5335408	423.6236131	431.3781808	1503.763288
F5	Min	524.4972472	505.9697543	511.0432623	507.2161901	521.0749319	517.9092277	568.464073
F5	Std	20.52288689	17.5883105	21.1382779	7.753959291	19.42887491	18.0127822	13.91884107
F5	Avg	553.8331902	528.5533505	548.6244946	521.6226298	558.0849858	544.6009872	592.1278605
F6	Min	609.3849995	600.0000158	602.3063067	600.0000366	618.9213338	600.613657	623.1038431
F6	Std	14.8474034	4.071016805	12.36315729	0.234106796	12.73038682	7.767112735	8.162057753
F6	Avg	640.3894897	602.1860294	619.148261	600.0919752	641.2518311	609.5853688	637.9418305
F7	Min	746.1419073	715.3248336	716.4830837	720.7027595	742.8876017	723.6007461	778.5065615
F7	Std	33.49573478	5.994771158	21.18957565	6.525417024	21.03863457	16.20756458	14.08573589
F7	Avg	788.8758672	725.318302	764.2527352	731.8262177	788.5452474	747.6698248	802.0960007
F8	Min	818.0575225	805.9697543	815.7632357	804.3599931	812.264262	812.9344577	836.3923042
F8	Std	12.06131224	7.881136734	4.997661809	3.27022318	9.77888418	11.77887075	6.639447165
F8	Avg	841.1675152	819.525296	823.315711	811.4176865	831.1772833	834.2110687	854.2163472
F9	Min	973.9862609	900	924.2823944	900	1268.690891	900.8790578	987.3470054
F9	Std	446.9868083	21.06966345	120.1544067	6.075511875	196.899002	66.1821854	208.8379895
F9	Avg	1416.255244	905.1949595	1124.57519	901.7502434	1549.413017	957.970385	1403.820818
F10	Min	1639.299532	1457.431299	1340.899145	1360.585514	1281.067407	1380.680203	2351.972064
F10	Std	343.4348347	245.4170129	171.4795651	178.1408899	314.2348903	389.4914654	181.3491617
F10	Avg	2254.144748	1913.118552	1718.161994	1736.154891	2052.471025	2034.13156	2713.708955

Table 2: Test results of the compared algorithms on CEC 2017 benchmark functions (F11–F19)

Function	Metric	WOA	PSO	POA	NGO	HHO	BBO	UBBO
F11	Min	1130.345481	1103.032907	1106.220907	1102.880813	1106.01185	1117.924905	1224.860929
F11	Std	70.78592317	31.89744919	40.89396031	2.927213046	72.81913977	103.4754571	753.6125282
F11	Avg	1221.946739	1130.464374	1153.97276	1106.547313	1197.087893	1233.129166	1712.76743
F12	Min	12819.57769	1950.15842	7376.67406	3143.42254	111489.0503	2308.8954	6962740.597
F12	Std	5271559.043	1650594.852	736115.401	7932.118847	4710587.697	5278842.319	70420716.31
F12	Avg	4408445.84	321736.334	401252.3659	12445.04318	4850458.476	3415030.201	67617889.6
F13	Min	2926.484276	1545.550945	1450.042071	1582.194771	2797.28092	2266.272342	19355.34163
F13	Std	17740.25399	6976.433872	1672.250068	2101.082761	7964.856628	12520.23262	544361.0688
F13	Avg	17310.02864	10511.99737	2854.318744	4059.563012	14189.91474	12908.36308	267725.1391
F14	Min	1506.102667	1435.219494	1428.636626	1453.315494	1459.012698	1495.958068	1827.860908
F14	Std	1688.593987	1642.962788	20.48641743	34.95848018	685.6289926	1514.217759	1079.238685
F14	Avg	2766.097907	2728.738243	1455.70804	1507.935593	1939.001931	2134.725407	4044.540485
F15	Min	1723.1111	1560.272244	1550.02904	1573.423125	1895.953888	1586.506454	3586.141153
F15	Std	9312.083039	6831.851787	89.12599711	139.3277827	3448.716386	7631.919711	7300.987928
F15	Avg	12859.19594	6184.877318	1648.2167	1747.804393	8115.810578	5027.326595	13084.22127
F16	Min	1652.97704	1600.995052	1604.402966	1603.381458	1636.291065	1603.359647	1814.886154
F16	Std	177.129443	142.7906929	111.0154875	63.8060813	145.4032906	130.1904198	78.36234809
F16	Avg	1931.594028	1802.367989	1781.652983	1668.429874	1894.250657	1796.43037	1987.078311
F17	Min	1733.620159	1719.15796	1729.122481	1729.832249	1733.448591	1742.041956	1787.300919
F17	Std	57.21932471	47.10618847	16.41747998	8.826110726	73.77264329	18.9257529	18.32622911
F17	Avg	1807.933702	1790.020639	1755.915574	1743.485687	1808.656304	1773.16411	1819.289225
F18	Min	4441.025471	2367.099315	1882.941045	1941.439715	2102.4139	2834.182896	53074.13145
F18	Std	11095.77139	11212.91487	5016.89834	1239.763517	11485.93932	16601.57223	4352500.118
F18	Avg	18926.26554	13693.57657	3627.824299	3043.694788	16203.97681	21421.14533	2834347.408
F19	Min	2338.140765	1925.116021	1903.798548	1946.32645	2410.004918	1957.995401	2535.787276
F19	Std	293729.5181	6374.47373	41.90839264	150.3522661	16487.95047	20276.99304	81211.82488
F19	Avg	98220.55116	5563.869265	1950.927721	2074.222742	19412.88908	12262.28595	84687.49841

Table 3: Test results of the compared algorithms on CEC 2017 benchmark functions (F22–F30)

Function	Metric	WOA	PSO	POA	NGO	HHO	BBO	UBBO
F22	Min	2311.621916	2300.55161	2303.268989	2237.915807	2247.261813	2244.193423	2266.206316
F22	Std	504.2377541	246.5755106	37.51031365	11.48709479	243.6904953	21.56518118	62.946838
F22	Avg	2490.127922	2385.947822	2328.182315	2298.709568	2355.348665	2311.599237	2385.069159
F23	Min	2614.703062	2610.922932	2618.894019	2605.174549	2618.039331	2617.741742	2531.767667
F23	Std	29.77431004	17.4867144	16.74801694	4.918143359	31.80853184	16.47571565	32.01821383
F23	Avg	2654.775108	2635.775183	2653.231675	2614.073333	2674.274028	2647.443472	2672.857749
F24	Min	2567.341429	2500	2509.078022	2536.848728	2502.976079	2531.751163	2584.93695
F24	Std	58.61597879	83.03116388	101.9262333	52.48584861	71.29980005	85.07312554	71.9336912
F24	Avg	2776.830546	2736.770578	2731.703952	2720.7144	2831.030627	2734.216854	2671.884038
F25	Min	2908.856445	2897.886175	2643.47058	2897.74287	2899.95271	2897.940444	3100.61773
F25	Std	37.38563711	25.19954864	66.97309877	22.54897102	25.97756681	22.08288675	203.3719889
F25	Avg	2968.380121	2929.217792	2928.037347	2924.428878	2931.877036	2944.146299	3470.704644
F26	Min	2835.440009	2800	2657.088936	2600.00383	2820.41515	2900	2987.818089
F26	Std	586.1311553	384.4573333	240.4537948	108.373461	551.6850903	194.1402854	156.9998726
F26	Avg	3527.223477	3103.137365	3066.512921	2908.866935	3549.737412	3136.141589	3229.378506
F27	Min	3096.248515	3095.274349	3089.975622	3089.005529	3101.480768	3093.987238	3110.362932
F27	Std	34.94936676	35.37964732	20.71797162	2.388521174	56.29519442	17.26150971	27.13505018
F27	Avg	3130.264845	3127.207124	3105.880181	3091.086251	3188.361179	3106.559737	3148.562185
F28	Min	3136.620357	2800	2884.064243	3100.000034	3102.77204	3100.000061	3227.761866
F28	Std	170.6225444	169.3999806	128.1870615	108.5215311	178.4777057	146.8822074	249.726607
F28	Avg	3466.448695	3235.900075	3267.029634	3199.085462	3396.639578	3363.72127	3698.564531
F29	Min	3185.844967	3163.133116	3149.702188	3156.460546	3207.575298	3169.440385	3236.310258
F29	Std	140.3170807	49.20656332	58.00989873	20.31923332	116.3122656	70.7316816	64.03041092
F29	Avg	3370.66891	3239.050238	3226.290026	3192.860242	3376.59524	3282.516053	3359.379575
F30	Min	10532.85519	4087.234517	4594.769486	4361.331518	33370.11907	6495.979404	366323.4605
F30	Std	2502685.526	900519.6641	863650.8699	181316.6143	1385564.683	596061.4272	2986765.131
F30	Avg	1991138.864	496941.3863	439796.6226	75432.50872	1122922.011	611515.6271	3898919.523

not, by itself, establish which algorithm performs better; the direction of the difference should be interpreted together with the descriptive statistics in Tables 1–3. PSO is not included because rank-sum test results for this comparison were not available. Note that the Holm procedure is applied separately to each comparator across the 27 functions; it does not treat all 135 tests as one single family.

Table 4: Holm-adjusted Wilcoxon rank-sum test p-values for UBBO against the comparison algorithms on the selected CEC 2017 functions

Function	WOA vs. UBBO	POA vs. UBBO	NGO vs. UBBO	HHO vs. UBBO	BBO vs. UBBO
F1	0.6858	$8.154e-10^{*}$	$9.014e-10^{*}$	$8.154e-10^{*}$	$7.550e-10^{*}$
F3	$1.347e-08^{*}$	$8.154e-10^{*}$	$1.060e-09^{*}$	$8.154e-10^{*}$	$7.550e-10^{*}$
F4	0.0016^{*}	$8.154e-10^{*}$	0.2882	$8.770e-09^{*}$	$7.550e-10^{*}$
F5	0.9650	$1.237e-08^{*}$	1.0000	$1.819e-08^{*}$	$5.056e-09^{*}$
F6	$8.150e-05^{*}$	$8.154e-10^{*}$	1.0000	$8.154e-10^{*}$	$2.442e-10^{*}$
F7	0.1075	$8.154e-10^{*}$	1.0000	$8.154e-10^{*}$	$7.550e-10^{*}$
F8	$3.403e-05^{*}$	$9.756e-09^{*}$	0.2178	$1.296e-07^{*}$	$1.586e-07^{*}$
F9	$1.550e-08^{*}$	$8.154e-10^{*}$	1.0000	$8.154e-10^{*}$	$3.272e-11^{*}$
F10	$9.145e-08^{*}$	$1.237e-08^{*}$	1.0000	$9.549e-06^{*}$	$2.984e-06^{*}$
F11	$3.829e-07^{*}$	$8.154e-10^{*}$	$1.967e-05^{*}$	$8.154e-10^{*}$	$7.550e-10^{*}$
F12	$3.835e-05^{*}$	$5.211e-09^{*}$	1.0000	0.0019^{*}	$1.936e-09^{*}$
F13	$6.892e-09^{*}$	$6.526e-07^{*}$	1.0000	0.0133^{*}	$7.550e-10^{*}$
F14	$1.995e-09^{*}$	$1.237e-08^{*}$	0.3353	0.0133^{*}	$7.550e-10^{*}$
F15	$1.995e-09^{*}$	$1.639e-08^{*}$	1.0000	$5.441e-06^{*}$	$7.550e-10^{*}$
F16	$2.403e-05^{*}$	$9.756e-09^{*}$	1.0000	$1.917e-05^{*}$	$2.609e-09^{*}$
F17	$1.110e-06^{*}$	$8.154e-10^{*}$	0.2269	$1.166e-07^{*}$	$1.627e-08^{*}$
F18	$1.062e-08^{*}$	$7.394e-09^{*}$	1.0000	0.0019^{*}	$7.550e-10^{*}$
F19	$2.480e-09^{*}$	$7.244e-09^{*}$	1.0000	$9.697e-06^{*}$	$7.550e-10^{*}$
F22	0.9650	$7.244e-09^{*}$	0.4184	$1.003e-08^{*}$	$5.056e-09^{*}$
F23	0.6858	$5.054e-09^{*}$	0.5973	$3.564e-08^{*}$	$1.567e-09^{*}$
F24	0.0499^{*}	$8.154e-10^{*}$	0.2358	$1.552e-09^{*}$	$7.550e-10^{*}$
F25	$4.916e-05^{*}$	$2.622e-08^{*}$	$6.430e-06^{*}$	$2.620e-07^{*}$	$7.787e-08^{*}$
F26	$7.280e-06^{*}$	$7.101e-09^{*}$	1.0000	$1.173e-06^{*}$	$7.428e-08^{*}$
F27	$9.166e-07^{*}$	$8.154e-10^{*}$	0.9508	$7.212e-07^{*}$	$1.828e-08^{*}$
F28	$1.134e-06^{*}$	$1.237e-08^{*}$	1.0000	$1.292e-05^{*}$	$2.984e-06^{*}$
F29	$1.253e-06^{*}$	$3.327e-09^{*}$	1.0000	$2.187e-09^{*}$	$9.374e-10^{*}$
F30	$3.896e-06^{*}$	$1.237e-08^{*}$	1.0000	$1.359e-06^{*}$	$1.586e-07^{*}$

Note: The reported values are Holm-adjusted p-values. The asterisk (*) indicates significance at $\alpha = 0.05$ after correction. The Holm correction was applied separately to the 27 function-wise tests for each comparator.

5.2.4 Convergence Curve Analysis

Figures 5–7 present the convergence curves for the functions included in the plotted benchmark set. The horizontal axis represents the iteration number, and the vertical axis shows the best fitness value obtained so far, using the scale indicated in each subplot. Tables 1–3 report numerical results for the functions listed in their captions. Function F2 is excluded in accordance with the commonly adopted CEC 2017 test protocol. The convergence plots shown here omit F12, F18, and F24; all three functions are included in the numerical tables.

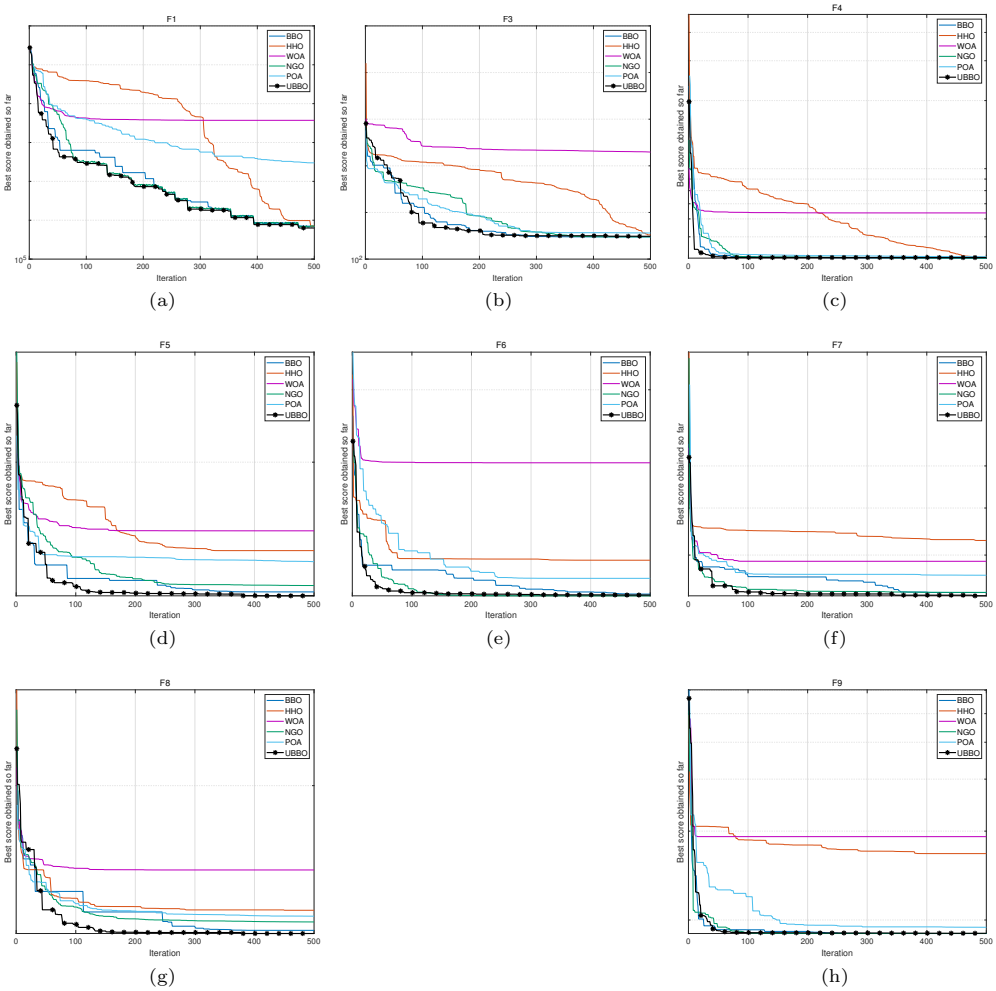

Fig. 5: Convergence curves of CEC 2017 test functions F1 and F3–F9

Observations drawn from the convergence curves are summarized as follows:

(1) Convergence speed: In the plotted tests, UBBO generally reduces the fitness value rapidly during the initial iterations. The target-oriented update is intended to provide additional directional information, although its individual contribution is not isolated in these experiments.

(2) Convergence accuracy: UBBO reaches low final fitness values in several plotted cases, indicating competitive search accuracy under the tested settings. The observed behavior reflects the combined optimization process, and cannot be attributed to any single modification without further ablation analysis.

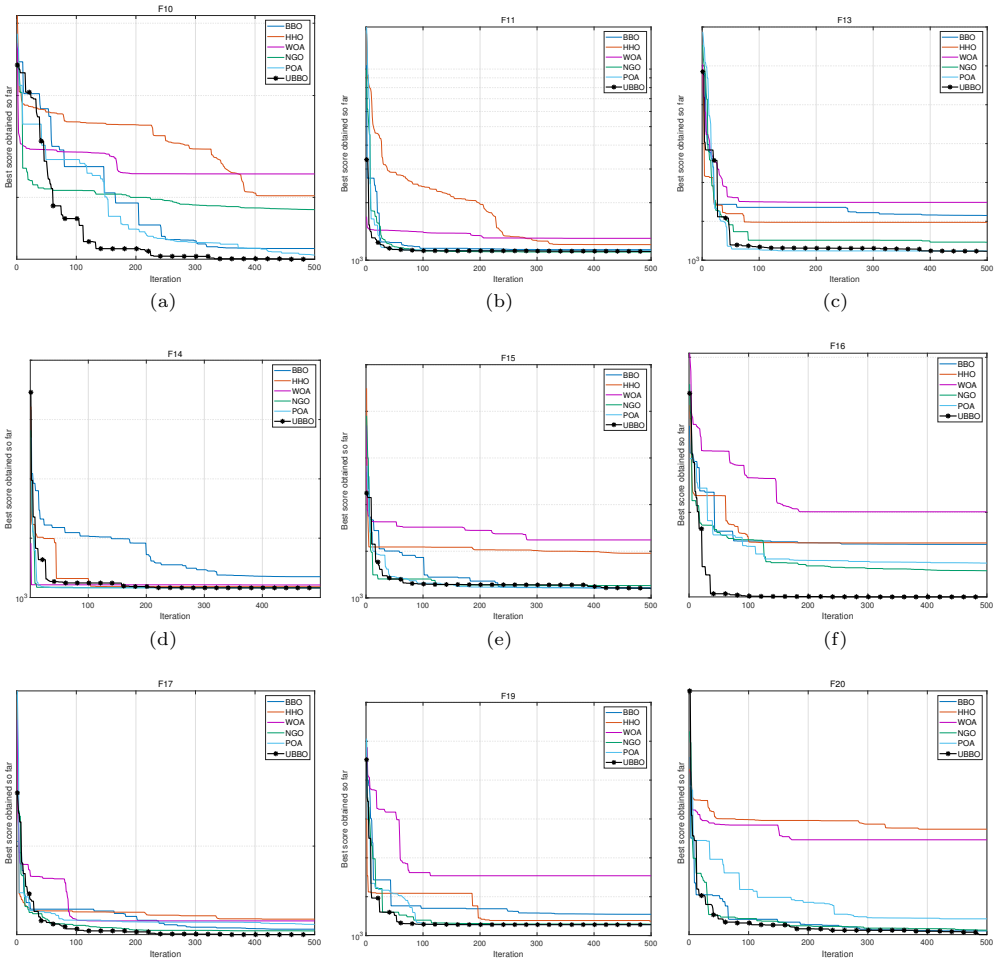

Fig. 6: Convergence curves of test functions F10, F11, F13–F17, F19, F20

(3) Convergence behavior: The plotted UBBO curves show relatively consistent progress in several cases. These results suggest that the combined update mechanisms may help balance broad search and local refinement in the selected tests.

5.3 UAV Path Planning Experiment

The UAV path-planning experiments are conducted in a three-dimensional terrain scenario containing six cylindrical obstacles. Each candidate path is represented by ten intermediate waypoints in addition to the prescribed start and end points. The start point is $(200, 100, 150)$ and the destination is $(800, 800, 150)$. The population size is set to 30, and the maximum number of iterations is set to 200. The compared algorithms

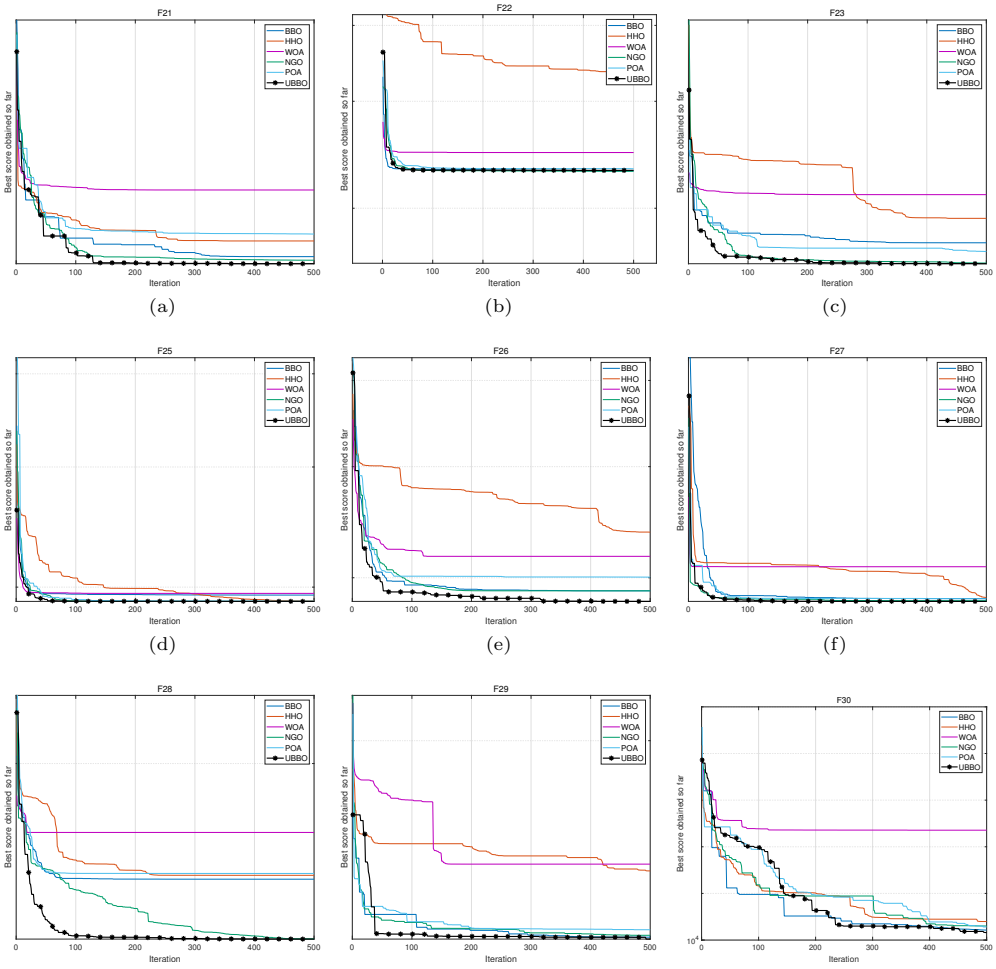

Fig. 7: Convergence curves of test functions F21–F23, F25–F30

are UBBO, BBO, WOA, PSO, POA, NGO, and HHO. The flight-path comparison and the corresponding cost statistics are presented in Figure 8 and Table 5, respectively.

Representative trajectories generated by the compared algorithms are shown in Figure 8. The figure provides a visual comparison of the spatial routes in the considered environment.

Table 5 reports the minimum, maximum, average, and standard deviation of the flight-cost values. These statistics summarize the distribution of the objective-function values obtained by the algorithms in the reported runs. A lower cost indicates a better solution under the adopted objective function, whereas a smaller standard deviation indicates less variation among repeated runs.

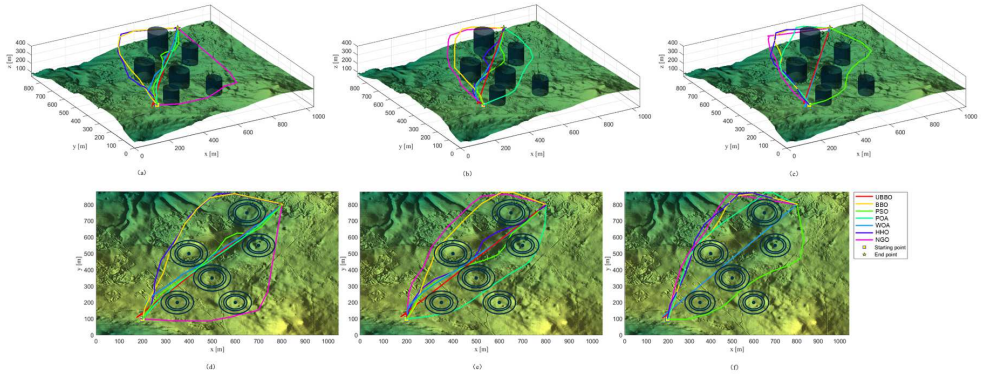

Fig. 8: Comparison of UAV paths generated by different algorithms in the three-dimensional terrain scenario

Table 5: Statistical results of flight cost obtained by the compared algorithms

Algorithm	Minimum	Maximum	Average	Std
UBBO	9.36E+03	9.59E+03	9.47E+03	1.12E+02
BBO	9.59E+03	1.17E+04	1.07E+04	1.06E+03
WOA	1.12E+04	1.33E+04	1.23E+04	1.05E+03
PSO	9.49E+03	1.18E+04	1.07E+04	1.17E+03
POA	9.56E+03	1.55E+04	1.25E+04	2.99E+03
NGO	1.14E+04	1.45E+04	1.30E+04	1.55E+03
HHO	9.54E+03	1.50E+04	1.23E+04	2.73E+03

According to Table 5, UBBO obtains the lowest minimum cost (9.36×10^3) and the lowest average cost (9.47×10^3) among the compared methods. Its standard deviation is also the smallest (1.12×10^2), indicating that its results vary less across the reported runs. The maximum cost obtained by UBBO is 9.59×10^3, which is lower than the maximum values reported for the other algorithms. These results suggest that UBBO performs favorably in terms of the reported flight-cost statistics in this scenario.

5.3.1 Convergence Performance Comparison

The convergence curves of the compared algorithms for the path-planning problem are shown in Figure 9.

The convergence curves illustrate the evolution of the objective-function values during optimization. Together with the statistical results in Table 5, they provide complementary information about the optimization process and the final solution quality in the selected simulation scenario.

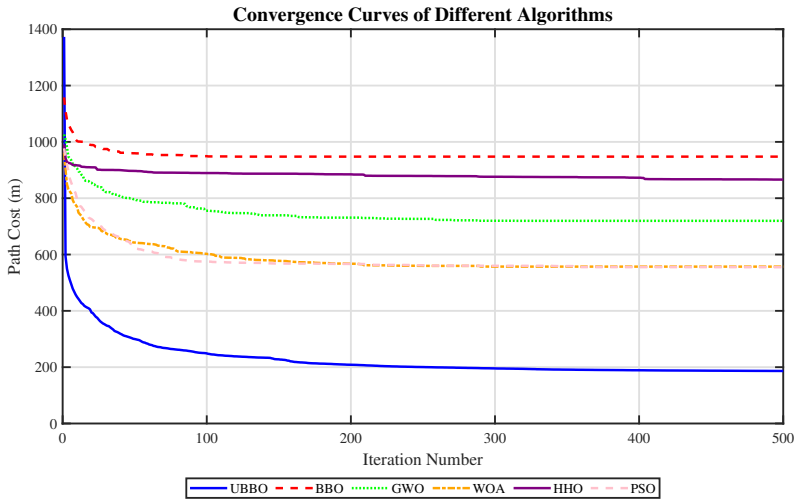

Fig. 9: Convergence curves for UAV path planning

5.4 Summary of Experimental Results

This section evaluated UBBO using selected CEC 2017 benchmark functions and a simulated three-dimensional UAV path-planning scenario. The benchmark results indicate competitive performance on several test functions, although the relative ranking varies across functions and algorithms.

In the UAV path-planning simulation, the trajectory plots provide a visual comparison, while the flight-cost statistics provide a quantitative comparison of the objective-function values. UBBO obtains the lowest reported minimum and average costs, as well as the smallest standard deviation, among the compared algorithms in the selected scenario. These results are specific to the adopted simulation settings and objective function.

Overall, the results provide an initial evaluation of the combined UBBO modifications in numerical optimization and the selected UAV path-planning application.

6 Conclusion and Future Work

To address the three-dimensional UAV path-planning problem, this paper proposes an improved Beaver Behavior Optimizer (UBBO). The proposed method incorporates three modifications into the original BBO framework. First, an adaptive phase factor based on population fitness information and Lévy-flight perturbations are introduced to regulate the search process and enhance population diversity. Second, a target-oriented population update mechanism is developed by incorporating terminal-point information into the explorer update strategy while retaining the original fitness-based architect selection. Third, a UAV-oriented path-processing module is introduced to reconstruct three-dimensional waypoint sequences, preserve predefined endpoints, and

restrict intermediate waypoint altitudes. These modifications aim to improve optimization performance and support the generation of geometrically consistent paths under the considered simulation settings.

The proposed method was assessed on selected CEC 2017 benchmark functions and in a simulated UAV path-planning scenario. It achieved competitive numerical optimization results and obtained favorable flight-cost statistics in the reported UAV path-planning scenario. These findings suggest that UBBO is a promising approach for the considered static path-planning problem, while further tests are required to establish physical flight feasibility.

Future work will focus on the following directions:

(1) *Path planning in dynamic environments.* The present study considers static obstacles, whereas real-world flight environments may contain moving obstacles and other time-varying elements. Future work could extend UBBO to dynamic path-planning tasks and investigate real-time obstacle avoidance and path-replanning schemes[8, 25, 52].

(2) *Cooperative path planning for multiple UAVs.* Demand for cooperative multi-UAV missions continues to grow with advances in UAV-swarm technology. UBBO could be combined with multi-agent coordination mechanisms to address multi-UAV path-planning problems while balancing individual trajectory quality and overall mission efficiency[6].

(3) *Hybrid optimization with deep learning.* Deep-learning methods can support feature extraction, environmental perception, and adaptive parameter tuning. Future work could therefore investigate hybrid frameworks that combine deep-learning modules with UBBO to improve decision-making in complex environments[46].

(4) *Validation on physical UAV platforms.* All experiments in this study were conducted in simulation. Future work should implement UBBO on physical UAV platforms and evaluate its effectiveness and reliability through field tests under realistic flight conditions[3].

Statements and Declarations

Funding: This work was supported by the National Natural Science Foundation of China (Grant Nos. 62272423 and 62673450), the Joint Fund Key Project of the Science and Technology Research and Development Plan of Henan Province, China (Grant No. 235200810022), and the Key Science and Technology Program of Henan Province, China (Grant No. 252102220062).

Competing interests: The authors declare that they have no competing interests.

Ethics approval and consent to participate: Not applicable.

Consent for publication: Not applicable.

Data availability: The datasets generated and analyzed during this study are available from the corresponding author upon reasonable request.

Materials availability: Not applicable.

Code availability: The source code developed for this study is not publicly available.

Author contributions: Huanlong Zhang contributed to the research methodology and manuscript review. Dian Jiao was primarily responsible for writing the original

manuscript and manuscript preparation. Yingcong Wang contributed to manuscript review and revision. Qingjun Jia contributed to manuscript review and revision. All authors read and approved the final manuscript.

References

[1] Abhishek B, et al (2020) Hybrid PSO-HSA and PSO-GA algorithm for 3D path planning in autonomous UAVs. SN Applied Sciences 2. https://doi.org/10.1007/s42452-020-03498-0

[2] Anderson EP, Beard RW, McLain TW (2005) Real-time dynamic trajectory smoothing for unmanned air vehicles. IEEE Transactions on Control Systems Technology 13(3):471–477. https://doi.org/10.1109/TCST.2004.839555

[3] Arafat MY, Alam MM, Moh S (2023) Vision-based navigation techniques for unmanned aerial vehicles: Review and challenges. Drones 7(2):89. https://doi.org/10.3390/drones7020089

[4] Awad NH, Ali MZ, Liang JJ, et al (2016) Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective real-parameter numerical optimization. Tech. rep., Nanyang Technological University, Singapore

[5] Cabreira TM, Brisolara LB, Ferreira Jr. PR (2019) Survey on coverage path planning with unmanned aerial vehicles. Drones 3(1):4. https://doi.org/10.3390/drones3010004

[6] Chen J, Zhang R, Zhao H, et al (2023) Path planning of multiple unmanned aerial vehicles covering multiple regions based on minimum consumption ratio. Aerospace 10(2):93. https://doi.org/10.3390/aerospace10020093

[7] Coello Coello CA (2002) Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. Computer Methods in Applied Mechanics and Engineering 191(11–12):1245–1287. https://doi.org/10.1016/S0045-7825(01)00323-1

[8] Dadkhah N, Mettler B (2012) Survey of motion planning literature in the presence of uncertainty: Considerations for UAV guidance. Journal of Intelligent & Robotic Systems 65:233–246. https://doi.org/10.1007/s10846-011-9642-9

[9] Das S, Suganthan PN (2011) Differential evolution: A survey of the state-of-the-art. IEEE Transactions on Evolutionary Computation 15(1):4–31. https://doi.org/10.1109/TEVC.2010.2059031

[10] Deb K (2000) An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering 186(2–4):311–338. https://doi.org/10.1016/S0045-7825(99)00389-8

[11] Dehghani M, Hubalovsky S, Trojovský P (2021) Northern goshawk optimization: A new swarm-based algorithm for solving optimization problems. IEEE Access 9:162059–162080. https://doi.org/10.1109/ACCESS.2021.3133286

[12] Dijkstra EW (1959) A note on two problems in connexion with graphs. Numerische Mathematik 1:269–271. https://doi.org/10.1007/BF01386390

[13] Dorigo M, Maniezzo V, Colorni A (1996) Ant system: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics 26(1):29–41. https://doi.org/10.1109/3477.484436

[14] Duan H, Yu Y, Zhang X, et al (2010) Three-dimension path planning for UCAV using hybrid meta-heuristic ACO-DE algorithm. Simulation Modelling Practice and Theory 18(8):1104–1115. https://doi.org/10.1016/j.simpat.2009.10.006

[15] Dubins LE (1957) On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. American Journal of Mathematics 79(3):497–516. https://doi.org/10.2307/2372560

[16] Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proc. 6th International Symposium on Micro Machine and Human Science, Nagoya, Japan, pp 39–43, https://doi.org/10.1109/MHS.1995.494215

[17] Floreano D, Wood RJ (2015) Science, technology and the future of small autonomous drones. Nature 521:460–466. https://doi.org/10.1038/nature14542

[18] Flores-Caballero G, Rodriguez-Molina A, Aldape-Perez M, et al (2020) Optimized path-planning in continuous spaces for unmanned aerial vehicles using meta-heuristics. IEEE Access 8:176774–176788. https://doi.org/10.1109/ACCESS.2020.3026666

[19] Fu Y, Liu D, Chen J, et al (2024) Secretary bird optimization algorithm: A new metaheuristic for solving global optimization problems. Artificial Intelligence Review 57:123. https://doi.org/10.1007/s10462-024-10729-y

[20] Garcia S, Fernandez A, Luengo J, et al (2010) Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. Information Sciences 180(10):2044–2064. https://doi.org/10.1016/j.ins.2009.12.010

[21] Goerzen C, Kong Z, Mettler B (2010) A survey of motion planning algorithms from the perspective of autonomous UAV guidance. Journal of Intelligent & Robotic Systems 57:65–100. https://doi.org/10.1007/s10846-009-9383-1

[22] Hart PE, Nilsson NJ, Raphael B (1968) A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics 4(2):100–107. https://doi.org/10.1109/TSSC.1968.300136

[23] Heidari AA, Mirjalili S, Faris H, et al (2019) Harris hawks optimization: Algorithm and applications. Future Generation Computer Systems 97:849–872. https://doi.org/10.1016/j.future.2019.02.028

[24] Holland JH (1992) Genetic algorithms. Scientific American 267(1):66–72. https://doi.org/10.1038/scientificamerican0792-66

[25] Hu L, Ma J, Zang S (2026) A 3D dynamic window approach with receding horizon optimization for real-time UAV path planning. Complex & Intelligent Systems 12:170. https://doi.org/10.1007/s40747-026-02299-z

[26] Huang R, He W (2026) A crisscross-strategy-boosted beaver behavior optimizer for global optimization and oil reservoir production. Scientific Reports 16:14363. https://doi.org/10.1038/s41598-026-43024-7

[27] Hwang YK, Ahuja N (1992) A potential field approach to path planning. IEEE Transactions on Robotics and Automation 8(1):23–32. https://doi.org/10.1109/70.127236

[28] Karaman S, Frazzoli E (2011) Sampling-based algorithms for optimal motion planning. The International Journal of Robotics Research 30(7):846–894. https://doi.org/10.1177/0278364911406761

[29] Khatib O (1986) Real-time obstacle avoidance for manipulators and mobile robots. The International Journal of Robotics Research 5(1):90–98. https://doi.org/10.1177/027836498600500106

[30] Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680. https://doi.org/10.1126/science.220.4598.671

[31] Kohli M, Arora S (2018) Chaotic grey wolf optimization algorithm for constrained optimization problems. Journal of Computational Design and Engineering 5(4):458–472. https://doi.org/10.1016/j.jcde.2017.02.005

[32] Li S, Zhang R, Ding Y, et al (2022) Multi-UAV path planning algorithm based on BINN-HHO. Sensors 22(24):9786. https://doi.org/10.3390/s22249786

[33] Liu Q, Zhang Y, Li M, et al (2021) Multi-UAV path planning based on fusion of sparrow search algorithm and improved bioinspired neural network. IEEE Access 9:124670–124681. https://doi.org/10.1109/ACCESS.2021.3109879

[34] Long W, Jiao J, Xu M, et al (2022) Lens-imaging learning Harris hawks optimizer for global optimization and its application to feature selection. Expert Systems with Applications 202:117255. https://doi.org/10.1016/j.eswa.2022.117255

[35] Mantegna RN (1994) Fast, accurate algorithm for numerical simulation of Lévy stable stochastic processes. Physical Review E 49(5):4677–4683. https://doi.org/

10.1103/PhysRevE.49.4677

[36] Mezura-Montes E, Coello Coello CA (2011) Constraint-handling in nature-inspired numerical optimization: Past, present and future. Swarm and Evolutionary Computation 1(4):173–194. https://doi.org/10.1016/j.swevo.2011.10.001

[37] Mirjalili S (2016) SCA: A sine cosine algorithm for solving optimization problems. Knowledge-Based Systems 96:120–133. https://doi.org/10.1016/j.knosys.2015.12.022

[38] Mirjalili S, Lewis A (2016) The whale optimization algorithm. Advances in Engineering Software 95:51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008

[39] Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Advances in Engineering Software 69:46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007

[40] Otto A, Agatz N, Campbell J, et al (2018) Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey. Networks 72(4):411–458. https://doi.org/10.1002/net.21818

[41] Ouyang K, Wei D, Sha X, et al (2026) Beaver behavior optimizer: A novel metaheuristic algorithm for solar PV parameter identification and engineering problems. Journal of Advanced Research 84:525–555. https://doi.org/10.1016/j.jare.2025.09.001

[42] Pavlyukevich I (2007) Lévy flights, non-local search and simulated annealing. Journal of Computational Physics 226(2):1830–1844. https://doi.org/10.1016/j.jcp.2007.06.008

[43] Pehlivanoglu YV, Pehlivanoglu P (2021) An enhanced genetic algorithm for path planning of autonomous UAV in target coverage problems. Applied Soft Computing 112:107796. https://doi.org/10.1016/j.asoc.2021.107796

[44] Primatesta S, Guglieri G, Rizzo A (2019) A risk-aware path planning strategy for UAVs in urban environments. Journal of Intelligent & Robotic Systems 95:629–643. https://doi.org/10.1007/s10846-018-0924-3

[45] Rahnamayan S, Tizhoosh HR, Salama MMA (2008) Opposition-based differential evolution. IEEE Transactions on Evolutionary Computation 12(1):64–79. https://doi.org/10.1109/TEVC.2007.894200

[46] Ramezani M, Amiri Atashgah MA, Rezaee A (2024) A fault-tolerant multi-agent reinforcement learning framework for unmanned aerial vehicles—unmanned ground vehicle coverage path planning. Drones 8(10):537. https://doi.org/10.3390/drones8100537

[47] Roberge V, Tarbouchi M, Labonté G (2013) Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning. IEEE Transactions on Industrial Informatics 9(1):132–141. https://doi.org/10.1109/TII.2012.2198665

[48] Shi Y, Eberhart RC (1998) A modified particle swarm optimizer. In: Proc. IEEE International Conference on Evolutionary Computation, pp 69–73, https://doi.org/10.1109/ICEC.1998.699146

[49] Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11:341–359. https://doi.org/10.1023/A:1008202821328

[50] Tizhoosh HR (2005) Opposition-based learning: A new scheme for machine intelligence. In: Proc. International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, pp 695–701, https://doi.org/10.1109/CIMCA.2005.1631345

[51] Trojovský P, Dehghani M (2022) Pelican optimization algorithm: A novel nature-inspired algorithm for engineering applications. Sensors 22(3):855. https://doi.org/10.3390/s22030855

[52] Venkatasivarambabu P, Puppala RS, Bikku T, et al (2026) Multi-UAV path planning in dynamic environments using a hybrid PSO-Whale Optimization approach. International Journal of Computational Intelligence Systems 19:231. https://doi.org/10.1007/s44196-026-01329-5

[53] Viswanathan GM, Buldyrev SV, Havlin S, et al (1999) Optimizing the success of random searches. Nature 401:911–914. https://doi.org/10.1038/44831

[54] Wang J, Zhu L, Zhao H, et al (2026) IBBO-UAV: An improved beaver behavior optimization for three-dimensional UAV path planning. In: Proceedings of the 38th Chinese Control and Decision Conference (CCDC 2026), pp 3975–3980, https://doi.org/10.1109/CCDC69976.2026.11560459

[55] Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1(1):67–82. https://doi.org/10.1109/4235.585893

[56] Xue J, Shen B (2020) A novel swarm intelligence optimization approach: Sparrow search algorithm. Systems Science & Control Engineering 8(1):22–34. https://doi.org/10.1080/21642583.2019.1708830

[57] Yang XS, Deb S (2009) Cuckoo search via Lévy flights. In: Proc. 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), pp 210–214, https://doi.org/10.1109/NABIC.2009.5393690

[58] Zhang R, Li S, Ding Y, et al (2022) UAV path planning algorithm based on improved Harris Hawks Optimization. Sensors 22(14):5232. https://doi.org/10.3390/s22145232