Applied Mathematical Modelling 150 (2026) 116383

Contents lists available at ScienceDirect

Applied Mathematical Modelling

journal homepage: www.elsevier.com/locate/apm

# A novel framework for 4D UAV swarm path planning

Gang Hu [a,b,*], Peidong He [a], Mahmoud Abdel Salam [c], Guo Wei [d]

[a] Department of Applied Mathematics, Xi'an University of Technology, Xi'an 710054, PR China
[b] School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, PR China
[c] Faculty of Computers and Information Science, Mansoura University, Mansoura, 35516, Egypt
[d] University of North Carolina at Pembroke, Pembroke, NC 28372, USA

ARTICLE INFO

*Keywords:*
Large-scale unmanned aerial vehicle swarm
Agricultural monitor
Military strikes
Autonomic flight
Environment modeling
Plan separation and consolidation method
Parallelization
Path search algorithm

ABSTRACT

UAVs (Unmanned Aerial Vehicles) have gained popularity in agricultural monitoring and military strikes due to their efficiency, flexibility, and affordability. However, achieving autonomic flight for UAVs still faces challenges. We make innovative adjustments in three aspects: environment modeling, path planning process, and path generator. Combine the advantages of the Global Path Search Algorithm and Local Path Search Algorithm, then design a large-scale UAV swarm path planning algorithm based on Plan Separation and Consolidation Method. We use many hybrid models to re-model the various environmental elements, which solves the weakness of the "combination explosion" and the inability to balance the various practical needs when extending the 2D grid method to 3D space. The Plan Separation and Consolidation Method separates the path planning problem of large-scale UAV swarms both in the time dimension and the dimension of the individuals of UAVs, and then consolidates the planning problems through temporary static obstacle regions, which utterly eliminates the "curse of dimensionality" in the planning process of large-scale UAV swarms, and provides a new opportunity for the autonomous control and parallelization of UAV swarms. We design a new local path evaluator and an enhanced Differentiated Creative Search algorithm to generate approximately optimal local paths quickly and accurately. We validate the effectiveness of the Plan Separation and Consolidation Method process and path generation method through a large number of repetitive experiments. Supplementary materials and related code can be downloaded at https://ogi.teracloud.jp/share/1202e180fd6a0c09.

## 1. Introduction

### 1.1. Problem background

UAV is a kind of unmanned aircraft widely used in many specific missions due to its advantages, such as being remotely operable, high mobility, portability, and flexibility. UAV technology has been rapidly developed [1]. In modern usage scenarios, UAVs are often used to perform missions, reference Table 1 for details. Due to the advantages of low cost and high-scale efficiency, in recent years, UAVs have also been emphasized in the military field to perform missions such as battlefield reconnaissance, intelligence collection, terrain survey, and airborne early warning [2,3] etc. Since UAV flight processes are not yet fully automated, operators need to be involved in decision-making to ensure that UAVs can successfully complete their missions [1]. This manual involvement diminishes

* Corresponding author.
*E-mail address:* hugang@xaut.edu.cn (G. Hu).

https://doi.org/10.1016/j.apm.2025.116383
Received 18 June 2025; Received in revised form 12 August 2025; Accepted 19 August 2025
Available online 22 August 2025

the cost-effectiveness and scalability advantages of UAVs.

The autonomous operation of UAVs primarily relies on the flying system and the path planning system, which provides the necessary power for operation and the technical support needed for navigation [13]. Path planning system is a core component of autonomous navigation, enabling UAVs to devise feasible routes, enhance navigation efficiency, and ensure safe and effective operation in complex environments [14]. Autonomous UAVs typically utilize embedded systems. However, constraints on motor power, sensor accuracy, and processor performance highlight the need to develop high-performance path-planning systems [15,16]. Single UAV is struggling to ensure the required efficiency in the mission, so the UAV swarm mission execution system is often used. This system has the advantage in terms of time and cost of performing the mission, and the mission required of one UAV can be accomplished by the other UAVs instead, which significantly improves the success rate of the mission execution [17]. In the mission execute system, different types of UAVs can collaborate on a mission and solve complex problems together [18].

### 1.2. Related work

Path planning consists of two main parts: environment modeling and path search methods [19].

Excellent environment modeling is crucial to improve the efficiency of path search algorithms. The grid method (GM) is often used to model the in-plane environment in general 2D path planning algorithms [20–22]; this modeling method was proposed by Howden in 1968 and has been widely used for its simplicity and easiness of implementation [23]. The GM modelling approach uses equidistant grids to partition a plane and binary matrices to track accessibility. While it can theoretically extend from 2D to 3D, practical implementation faces challenges. Firstly, the grid size must be set empirically; if too small, complexity increases, raising performance demands and potentially obscuring details. Secondly, as the solution space grows, the grids increase rapidly, leading to a "combinatorial explosion" problem [24]. Therefore, the classical methods that consider meshes or dispersed points are not suitable for 3D path planning since they cannot often deal with high-complexity problems successfully [25]; for this problem, Han used the COSPS modelling method to reduce reference points in 3D path planning by enclosing obstacles with a subset of mesh points [25]. Lei et al. utilized the Maklink graph approach to create a robust and scalable path-planning solution, enhancing performance and safety [26]. Ali et al. divided the UAV flying area into multiple 2D planes to analyze specific space features [27]. Ma and Chen applied the MS-DLG model, using a quad-tree structure to convert 3D data to a refined multi-scale 2D grid, significantly improving path planning efficiency [28].

Path search methods are divided into global path search algorithms (GPSA) and local path search algorithms (LPSA) based on the availability of environmental information [29]. GPSA focuses on planning the entire mission for a UAV using a known global environment and is a centralized approach. While it ensures complete paths, it struggles with large-scale UAVs and complex dynamic environments, resulting in non-real-time path generation, which does not meet realistic planning needs [29]. The common GPSA methods are A* [20], D* [30], Dijkstra [31], GA [32], DE [33], ACO [34,35], and PSO [36,37], of which GA [38], DE [39], ACO [40], and PSO [41] are all classical metaheuristic algorithms (MAs), most of these algorithms simulate the natural behaviors of animals, natural physical phenomena, human social behaviors, and some natural laws [42,43], and some other MAs have also been applied to GPSA by researchers [44–46]. Cheng et al. found that MAs based various path-planning algorithms (GA, DE, ACO, and PSO) have distinct advantages and disadvantages [47]. GA is strong in global search but weak in stability and local optimization, leading to high storage demands. DE retains GA's strengths while improving storage issues and robustness, although it still faces stability challenges. ACO benefits from its pheromone-based system, enhancing efficiency in later iterations but struggling initially due to the time required to stabilize. PSO seems to balance the above advantages and disadvantages in terms of performance, but its generated solution has a single structure and it does not have the ability to mutate similarly to GA and DE, which makes it very vulnerable to local extremes [48].

Using the GPSA method for UAV swarm path planning increases planning complexity due to overly long paths and excessive numbers of UAVs. For example, suppose ABCDE is 5 UAVs in a UAV swarm; for UAV A, all UAV BCDE are obstacles that need the UAV A to avoid during the planning process, so the UAV swarm path planning problem is a problem dealing with a dynamic environment; When adopting the GPSA method to plan the path, the routes of the two UAVs can't exist cross, however, assuming that T1 and T2 are two-time points in the UAV swarm path planning the position of UAV A at the moment of T1 doesn't affect the position of UAV B at the moment of T2, so the route can exist cross. This difficulty makes it hard for GPSA to be applied in practice.

Standard LPSA methods include rapidly exploring random trees [49], artificial potential fields [50], fuzzy logic algorithms [51], neural networks [52], and reinforcement learning [53]. Unlike GPSA, LPSA does not rely on complete environmental data, but it may

**Table 1**
Some examples and explanations of UAV applications.

| Application | Reference | Year | Application |
|---|---|---|---|
| photography | [4] | 2024 | Assess plant growth conditions on cliffs. |
| transportation | [5] | 2024 | Vertical delivery system between floors. |
| security patrol | [6] | 2024 | Traffic regulations and detect traffic collisions. |
| disaster surveillance | [7] | 2024 | In response to the January 2024 Noto Peninsula earthquake. |
| climate monitoring | [8] | 2023 | Atmospheric detection covers the lower troposphere, polar and oceanic regions. |
| environmental research | [9] | 2024 | Survey terrain and detect Pacific oysters. |
| search and rescue | [10,11] | 2024 | Application in time-sensitive emergencies. |
| agricultural inspection | [12] | 2024 | Collect images for identifying weed patches. |

produce incomplete paths and can get stuck at local minima when faced with complex obstacles. LPSA is favoured for its fast-planning speed and adaptability in real-time environments, making it ideal for autonomous UAVs. Each UAV requires specific space in swarm operations, which changes over time, complicating the environment. As the number of UAVs increases, the problem's dimensionality rises, causing traditional LPSA to struggle with the "curse of dimensionality," complicating large-scale UAV swarm control [54].

LPSA seeks the optimal path by expanding nodes, so the local path planning problem is a convex optimization problem. In reality, most optimization problems have poor continuity, high computational cost, nonlinear constraints, nonconvex search region, presence of noise, etc., and traditional optimization methods are incapable of guaranteeing the robustness and accuracy of the results [55,56]. For these problems, the approximate optimal solution can usually be used instead of the best solution, and metaheuristic algorithms are often used to find this approximate optimal solution [57]. As a random-based iterative optimization algorithm, meta-heuristic algorithms are widely used to solve optimization problems for their simple parameters, lack of dependence on gradient information, and high robustness [42,58].

Both GPSA and LPSA have their strengths and weaknesses, and combining multiple types of path-planning algorithms can complement each other and better handle unknown, dynamic and complex scenarios [48].

### *1.3. Contribution of this paper*

We combine the MAs with the GPSA and the LPSA method, optimize the three main aspects: the environment modeling, the planning process, and the path generation method. Major contributions of this paper are:

**(1) Optimization of the environment modeling method:** We adopted a hybrid modeling method to simplify the environment model for path planning. Different methods were used for key elements. This keeps the optimization problem's difficulty relatively constant, regardless of the number of drones present. This is an important innovative environmental pre-processing method for addressing the impact of problem dimension changes on solver stability.
**(2) Redesign of the UAV swarm path planning process:** We propose an innovative path planning process framework. We discretized the planning process in time and generating local paths for UAV swarms in each period. Each UAV's planning is treated as a separate 3-D fixed dimensional problem, overcoming the "curse of dimensionality." We then merge these individual UAV paths into a cohesive swarm path using temporary static obstacle regions. This method is referred to as the Plan Separation and Consolidation Method (PSCM).
**(3) Innovative design of local path evaluator:** We redesigned a local path evaluator. We consider factors like path length, collision risks with the ground, threats, and other UAVs, while also incorporating boundary constraints.
**(4) High-performance local path generator:** Due to the time discretization process, the runtime of the local path generator is limited; the limitation of the UAV's characteristics also requires the local path generator to have a very high efficiency without excessive resource consumption; in addition, since the environment complexity changes in real-time, robustness is also a focused aspect of the local path generator. For these reasons, we decided to use DE-type MAs to generate local paths. We select a new DE combinatorial algorithm in this paper, which is the Differentiated Creative Search (DCS) algorithm [59].

The DCS suffers from poor stability and poor performance in terms of local optimization ability, therefore we improved the DCS further: Using the quasi-random number generator to generate non-crowded initial positions which enhances the exploration ability in the initialization phase of DCS; improving the team diversification process in DCS, giving the Low Performer individuals some self-improvement ability while retaining their roles in the population; A local attraction and repulsion operator is integrated into the Average Performer individuals' position updating process to enhance its local exploitation ability; the population structure of the algorithm is readjusted to make the algorithm adaptable to different storage limitation requirements and running time (number of evaluations) requirements.

The structure is as follows: Section 2 introduces the hybrid modeling approach; Section 3 covers UAV swarm path planning using PSCM; Section 4 details the local path evaluator's computations; Section 5 presents the SiMlDCS algorithm; and Section 6 includes experimental results with numerical and environmental simulations.

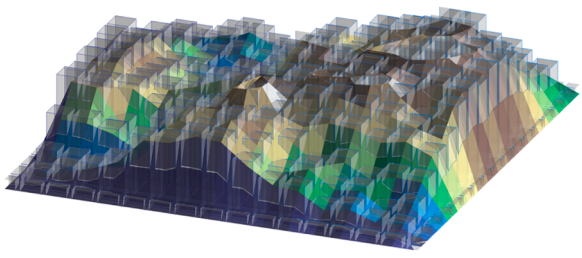

**Fig. 1.** Schematic of the DGM model.

## 2. Environment modeling method

Environment modeling encompasses digitizing complex models for path planning, focusing on four components: terrain, UAV body and motion, obstacle areas, and both global and local paths.

**Terrain model:** We employ a simplified discrete grid model (DGM) based on the MS-DLG method, storing terrain data in a grid matrix that represents maximum altitudes for each area. Fig. 1 shows a 17×17 DGM of a randomly generated map, with the height of transparent white rectangles indicating maximum altitudes. All units in this paper are uniform.

**UAV body model and motion model:** We model the UAV as a reference ball (Fig. 2). This ball has a fixed fuselage safety radius.

The UAV swarm as a whole moves in time, the initial position of the *i*th UAV in the swarm is denoted as $U_i.S$, the end point (target point) position of the *i*th UAV is denoted as $U_i.E$, and the body safety radius is denoted as $U_i.R$, where

$$U_i.S = (U_i.Sx, U_i.Sy, U_i.Sz), \tag{1}$$

$$U_i.E = (U_i.Ex, U_i.Ey, U_i.Ez), \tag{2}$$

where $U_i.Sx$, $U_i.Sy$, $U_i.Sz$, $U_i.Ex$, $U_i.Ey$, $U_i.Ez$ denotes the components of $U_i.S$ and $U_i.E$ in space in the three directions, respectively.

The UAV passes through a series of waypoints according to flying instructions. The position of the *i*th UAV at the *t*th time node is denoted as $U_i^t.P$ and the movement command is denoted as $U_i^t.\Delta$, where:

$$U_i^t.P = \left(U_i^t.Px, U_i^t.Py, U_i^t.Pz\right), \tag{3}$$

$$U_i^t.\Delta = \left(U_i^t.\Delta x, U_i^t.\Delta y, U_i^t.\Delta z\right), \tag{4}$$

where $U_i.Px$, $U_i.Py$, $U_i.Pz$, $U_i.\Delta x$, $U_i.\Delta y$, $U_i.\Delta z$ denotes the components of $U_i.P$ and $U_i.\Delta$ in space in the three directions, respectively.

The position update formula for the *i*th UAV at the *t*th time node based on the movement command $U_i.\Delta$ is as follows:

$$U_i^{t+1}.P = U_i^t.P + U_i^t.\Delta \quad . \tag{5}$$

The space occupied by the UAV reference ball over time is referred to as the local path of the UAV. The moving trajectory of the UAV each time is recorded as ($l$, θ, ϕ), then there are:

$$U_i^t.\Delta x = l \cdot \sin(\theta) \cdot \cos(\phi), \tag{6}$$

$$U_i^t.\Delta y = l \cdot \sin(\theta) \cdot \sin(\phi), \tag{7}$$

$$U_i^t.\Delta z = l \cdot \cos(\theta), \tag{8}$$

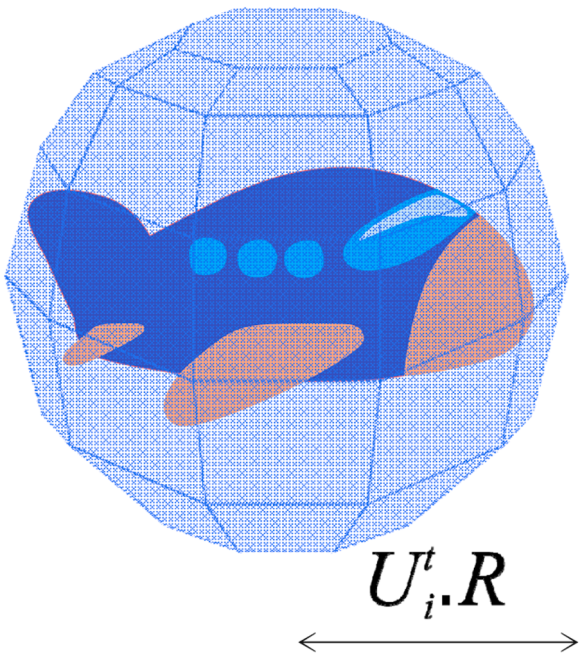


**Fig. 2.** UAV body reference ball.

where $l$ denotes the moving distance of the UAV; θ denotes the pitch angle of the UAV; and denotes the horizontal turning angle of the UAV.

**Two obstacle areas model:** The hybrid model includes two obstacle regions: The first type of obstacle area is a static, cylindrical obstacle area with a known radius and an infinite height, which serves to simulate areas such as residential areas, thunderstorms and storms, and military-restricted areas. The position of the cylindrical obstacle region is fixed, the horizontal center of the $i$th fixed cylindrical obstacle region is denoted as ($St_i.x$,$St_i.y$), and the radius is denoted as $St_i.r$, the cylindrical obstacle is expressed as:

$$(X - St_i.x)^2 + (Y - St_i.y)^2 \leq (St_i.r)^2, \tag{9}$$

where ($X$, $Y$) denotes the projection coordinates in the $xoy$ plane of a point that is inside the cylindrical obstacle region.

The second type of obstacle region is a temporary, cylindrical shape representing the paths of a single UAV over a unit of time. It is created during individual UAV path planning and disappears once all UAVs complete their planning in the unit of time. The schematic of this capsule-like obstacle region is shown in Fig. 3.

**Global paths model and local paths model:** The optimization algorithm refines several trail segments, gradually connecting them to reach the endpoint. The global path schematic made up of local paths is illustrated in Fig. 4. Hu et al. proposed a ball security corridor path planning method using the ball Said-Ball (BSB) curve and a novel super hawk optimization algorithm for UAVs [60]. This method establishes a corridor model with multiple consecutive balls, ensuring a smooth flight path while maintaining a safe range to avoid cumulative errors. This paper referenced the security corridor model for UAVs. The schematic of the local path security corridor is shown in Fig. 5.

## 3. PSCM and the termination conditions of UAV flying

We are using the PSCM for UAV swarm path planning, which simplifies optimization by focusing on portions of a single UAV's path per time unit. A schematic is shown in Fig. 6. In the $T = 1$ period, the local path of UAV A is first planned, and the blue capsule body is the trail traveled by UAV A in that period, as in Fig. 6(a). For UAV B, the security corridor corresponding to the already generated path would be considered as a temporary static obstacle region, as in Fig. 6(b). Under the constraints of the static temporary obstacle, UAV B also generates a local path, as in Fig. 6(c). Similarly, if there are more UAV swarms individually in that period, the local paths generated by UAV A and B corresponding security corridor will also be considered as a temporary static obstacle region; At the beginning of the $T = 2$ period, all the temporary static obstacle regions generated in period $T = 1$ are cleared, as in Fig. 6(d), and then the above process is continued as in Figs. 7(e,f,g,h); until all UAVs reach the target region, the path planning process concludes, as shown in Fig. 6(i).

The above process can be promoted to $n$ UAVs. Unlike GPSA's completeness process for global path generation, LPSA has diverse termination conditions for UAV path planning:

**(1)** Collision with other elements leads to mission failure and removal from PSCM;
**(2)** Collision with other UAVs leads to mission failure and removal from PSCM;
**(3)** Exceeding maximum mission time leads to mission failure and removal from PSCM;
**(4)** Successfully reaching the mission submission point records mission time and flying distance before removal from space while other UAVs continue.

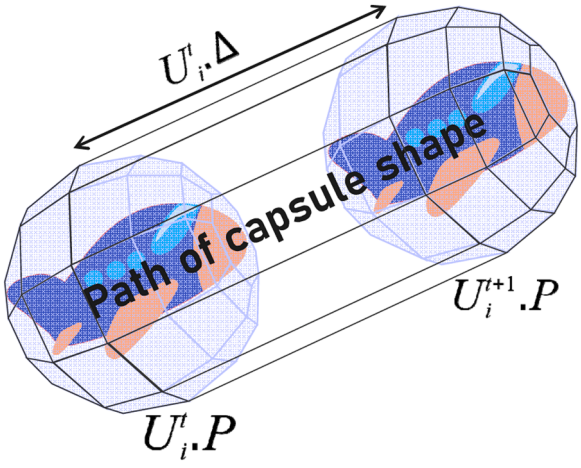


**Fig. 3.** Schematic of temporary static obstacle.

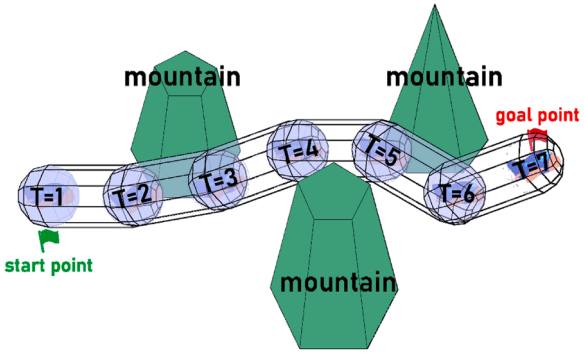


**Fig. 4.** Local paths composing global path.

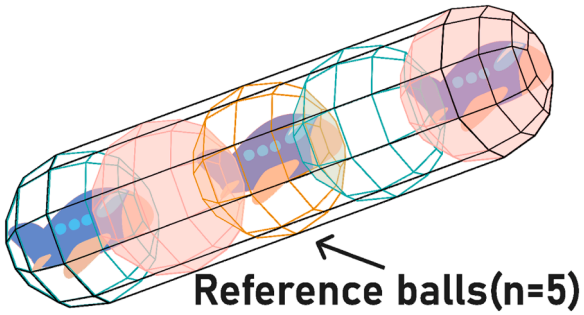


**Fig. 5.** Schematic of security corridors for local paths.

## 4. Local path evaluator

Each local path corresponds to an fitness value based on the local path evaluator. The lower the fitness value, the better the local path. We use a penalty function to combine multiple path evaluation criteria. Formula for calculating fitness values based on penalty functions is expressed as follows:

$$Cost(\mathbf{x}) = \sum_{i=1}^{l} a_i \times f_i(\mathbf{x}) + \left[ \sum_{j=1}^{m} b_j \times G_j(\mathbf{x}) + \sum_{k=1}^{n} c_k \times H_k(\mathbf{x}) \right], \tag{10}$$

where $a_i$, $b_j$, $c_k$ is the weighting coefficient of each fitness value; $f_i(x)$ indicates the penalty values of non-constraint; $G_j(x)$, $H_k(x)$ are the penalty values of the equation constraints and unequal constraints, respectively.

The local path evaluation formula considers: (1) velocity limits; (2) path length; (3) terrain and static obstacle constraints; (4) boundary constraints; (5) collision risk with other UAVs; and (6) preemptively avoiding obstacle areas. The local path fitness value is calculated using this formula:

$$\begin{aligned} Cost\left(U_i^t.P, U_i^t.\Delta\right) = {} & C^{t,1}\left(U_i^t.P, U_i^t.\Delta\right) + C^{t,2}\left(U_i^t.P, U_i^t.\Delta\right) + C^{t,3}\left(U_i^t.P, U_i^t.\Delta\right) \\ & + C^{t,4}\left(U_i^t.P, U_i^t.\Delta\right) + C^{t,5}\left(U_i^t.P, U_i^t.\Delta\right) + C^{t,6}\left(U_i^t.P, U_i^t.\Delta\right), \end{aligned} \tag{11}$$

where the definitions and calculations of each of these components are detailed in this section.

**Velocity limits**: With the PSCM process, we can manage speed constraints while generating and adjusting candidate solutions, so no further explanation is necessary.

**Path length:** Akay et al. proposed a method that calculates the Euclidean distance from the start point to the waypoint and then to the goal, enabling online path generation for UAV swarms in 2D [17]. We extend this approach to a 3D solution space, with the initial path length fitness value calculated accordingly, and the path length fitness value before optimization is given as:

$$\widetilde{C}^{t,1}\left(U_i^t.P, U_i^t.\Delta\right) = \| U_i^{t+1}.P - U_i^t.P \| + \| U_i^{t+1}.P - U_i.E \| . \tag{12}$$

The Eq. (12) has difficulty coping with complex terrain. We assign different weights to horizontal and vertical distances of UAVs for navigating complex terrains. The optimized $\widetilde{C}^{t,1}\left(U_i^t.P, U_i^t.\Delta\right)$ is given as:

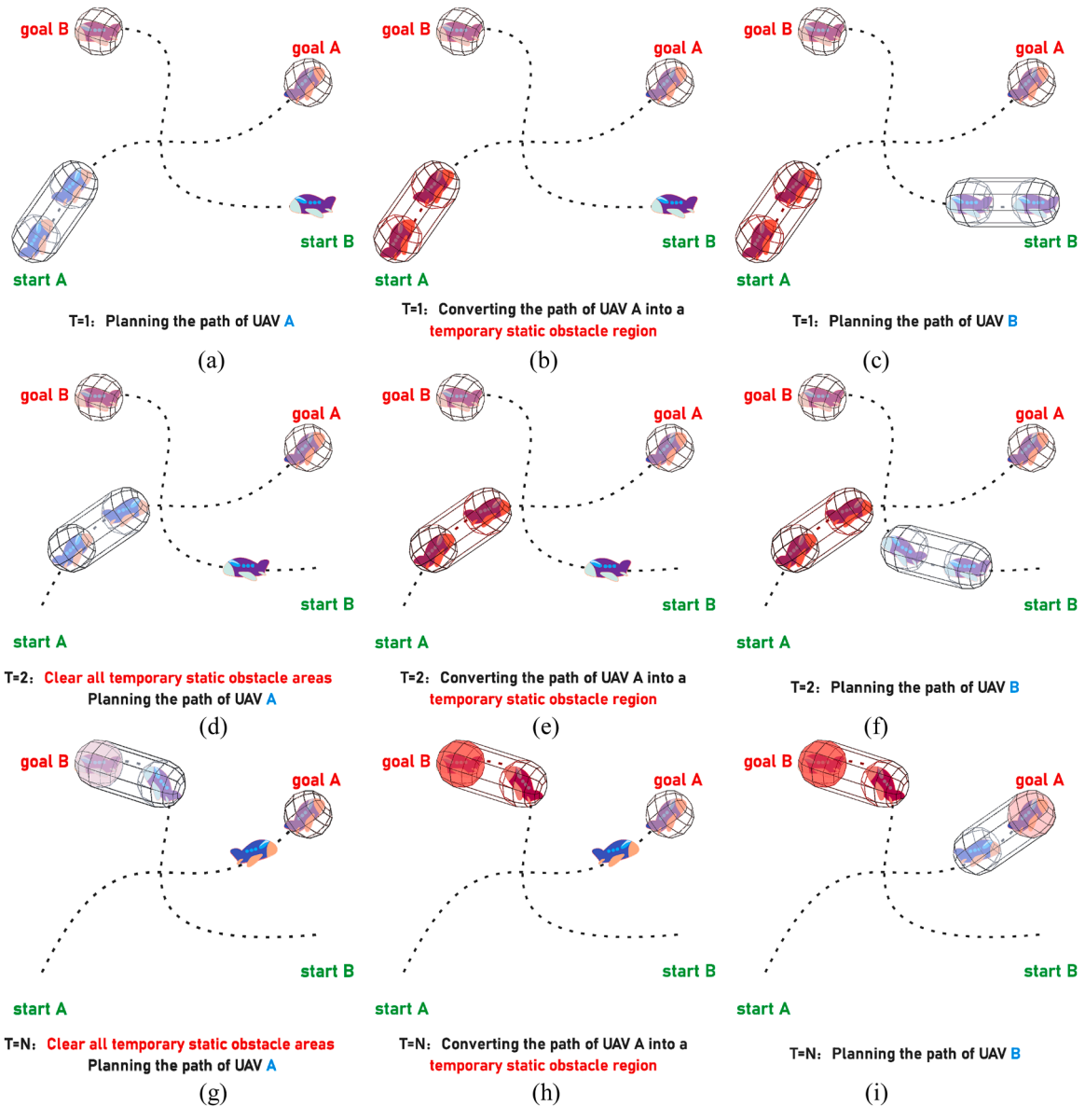


**Fig. 6.** Schematic of two UAVs performing the PSCM process.

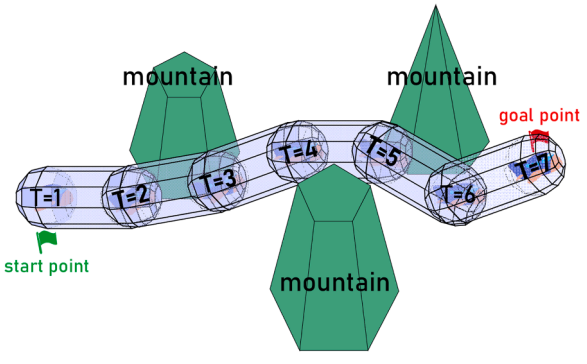


**Fig. 7.** Schematic of the formula for calculating the path length fitness value before optimization.

$$C^{t,1}\left(U_i^t.P, U_i^t.\Delta\right) = \sqrt{k^2\left(U_i^{t+1}.Px - U_i^t.Px\right) + k^2\left(U_i^{t+1}.Py - U_i^t.Py\right) + \left(U_i^{t+1}.Pz - U_i^t.Pz\right)} + \sqrt{k^2\left(U_i^{t+1}.Px - U_i.Ex\right) + k^2\left(U_i^{t+1}.Py - U_i.Ey\right) + \left(U_i^{t+1}.Pz - U_i.Ez\right)}, \tag{13}$$

where all expressions except $k$ have the same meaning as above, and $k$ denotes the weighting coefficient, calculated as:

$$k = 1 + \sqrt{\frac{\| U_i^{t+1}.P - U_i.E \|}{\| U_i.S - U_i.E \|}}. \tag{14}$$

The implication expressed in Eq. (12) is to minimize the length sum of the blue line segment and the purple line segment in Fig. 7, and the limit of this length sum is the straight line length of the current position from the target point.

**Terrain constraints:** Following the security corridor model by Hu et al., we assess the relationship between the path and the ground, providing a punitive fitness value for terrain constraint violations. The punitive fitness value is given as:

$$C^{t,2}\left(U_i^t.P, U_i^t.\Delta\right) = \sum_{n=1}^{N} HC_n, \tag{15}$$

where $N$ denotes the number of sequential balls constituting the security corridor, and we adopt $N = 21$; $HC_n$ is given as:

$$HC_n = \begin{cases} 0 & H_n - Hmap_n > Hsafe \\ (Hsafe - H_n + Hmap_n) \times Punish & Hsafe \geq H_n - Hmap_n > 0 \\ 100 \times (Hsafe + |H_n - Hmap_n|) \times Punish & H_n - Hmap_n < 0 \end{cases}, \tag{16}$$

where $Hmap_n$ denotes the terrestrial altitude corresponding to the center of the $n$th reference ball; $H_n$ denotes the altitude at the location of the $n$th reference ball; $Hsafe$ denotes the safe flying altitude(relative to the ground) of the UAV; and $Punish$ is a larger number.

**Static obstacle constraints:** Based on the local path modelling approach for security corridor, the penalty fitness value for violating the static obstacle restriction is given as:

$$C^{t,3}\left(U_i^t.P, U_i^t.\Delta\right) = \sum_{n=1}^{N}\sum_{m=1}^{M} DC_{n,m}, \tag{17}$$

where $N$ denotes the number of sequential balls constituting the security corridor, and we adopt $N = 21$; $M$ is the number of static cylindrical obstacle areas; $DC_{n,m}$ is given as:

$$DC_{n,m} = \begin{cases} 0 & D(n,m) > U_n.R + St_m.r + Rsafe \\ Punish \times (\alpha \times D(n,m))^2 & U_n.R + St_m.r < D(n,m) \leq U_n.R + St_m.r + Rsafe \\ Punish \times (1/(\alpha \times D(n,m))) & D(n,m) \leq U_n.R + St_m.r \end{cases}, \tag{18}$$

where $D(n, m)$ denotes the Euclidean distance between the $n$th reference ball constituting the security corridor and the $m$th static obstacle region in the horizontal direction; $Rsafe$ denotes the safety distance between the UAV and the static obstacle region; $Punish$ is a larger number; $\alpha$ is a coefficient that balances the magnitude relationship of the computational formulas $DC_{n,m}$ at the intersection; $St_m.r$ denotes the radius of the $m$th static obstacle region; and $U_i.R$ denotes the radius of the body of the $i$th UAV.

**Boundary constraints:** Since the relationship between the trajectory and the solution boundary isn't directly observable, we add boundary constraints to penalize paths that exceed the defined limits. The punitive fitness value for violating the boundary constraint is shown as follows:

$$C^{t,4}\left(U_i^t.P, U_i^t.\Delta\right) = \sum_{n=1}^{N} (LBxC_n + UBxC_n + LByC_n + UByC_n + LBhC_n + UBhC_n), \tag{19}$$

where $N$ denotes the number of sequential balls constituting the security corridor; $LBxC_n$, $UBxC_n$, $LByC_n$, $UByC_n$, $LBhC_n$, $UBhC_n$ denote the penalized fitness values corresponding to the $n$th reference ball constituting the security corridor that exceeds the upper and lower limits of the three dimensions. respectively.

The $LBxC_n$ is calculated as:

$$LBxC_n \begin{cases} 0 & X_n \geq LBx + U_i.R \\ Punish \times |X_n - LBx - U_i.R| & else \end{cases}, \tag{20}$$

where $X_n$ denotes the coordinates in the $x$ direction corresponding to the $n$th ball that constitutes the security corridor; $LBx$ is the lower bound of the defined region of the path planning in the $x$ axis direction; and $U_i.R$ denotes the radius of the body of the $i$th UAV, $Punish$ is a larger number. $UBxC_n$, $LByC_n$, $UByC_n$, $LBhC_n$, $UBhC_n$ is defined in the similar style to $LBxC_n$ and is not additionally described here, where $UBx$, $LBy$, $UBy$, $LBh$, $UBh$ is also define here.

**Collision risk constraints:** We redefine the relationship between the reference ball and the temporary obstacle region as the

positional relationship between a point and a line segment in space. This simplification is illustrated in Fig. 8.

Denote the distance from the point to the line segment as $D_{2l}\left(O, \overrightarrow{ab}\right)$, with:

**(1)** If the projection of a point onto the line of a segment falls on the segment, the distance to the segment is the perpendicular distance (see Fig. 9(a));
**(2)** If the projection does not fall on the segment, the distance is the shortest distance to the segment's endpoints (see Fig. 9(b)); where $O'$ is the projection point of $O$ onto the line on which the line segment $ab$ is located.

The penalty fitness value for avoiding other UAVs, based on Definition 1, is given by:

$$C^{t,5}\left(U_i^t.P, U_i^t.\Delta\right) = \sum_{n=1}^{N}\sum_{k=1}^{K} OAC_{n,k}, \tag{21}$$

where $N$ denotes the number of sequential balls constituting the security corridor, $K$ denotes the number of temporary static obstacle areas, the $OAC_{n,k}$ is calculated as as:

$$OAC_{n,k}\begin{cases} 0 & D_{2l}(O_n, L_k) \geq Lsafe + U_n.R + U_k.R \\ Punish^2 & else \end{cases}, \tag{22}$$

where $O_n$ denotes the ball center position of the $n$th reference ball in the security corridor; $L_k$ denotes the center line of the $k$th temporary static obstacle region; $Lsafe$ is the safe distance between two UAVs; $Punish$ is a larger number; $U_i.R$ denotes the radius of the body of the $i$th UAV, and $D_{2l}\left(O, \overrightarrow{ab}\right)$ is the newly defined distances according to Definition 1.

**Preemptively avoiding obstacle areas:** By adding the pre-avoidance area to enable the UAV to avoid some obstacle areas in advance in this paper, the schematic of the pre-avoidance systerm is given in Fig. 10. The penalized fitness value of the UAV sensing the obstacle ahead is given by:

$$C^{t,6}\left(U_i^t.P, U_i^t.\Delta\right) = \sum_{q=1}^{Q} Paw(q) \times Pac\left(U_i^t.P, U_i^t.P + Pal \times \frac{q}{Q} \times U_i^t.\Delta\right), \tag{23}$$

where $Paw$ is a series of weighting coefficients; $Q$ is an integer, the larger $Q$ is, the more sensitive the UAV is to the distance of obstacles, but the computational cost will increase accordingly; here $Q = 10$, to make this penalty value will change with the distance of obstacles to the UAV and the velocity of the UAV itself, this penalty value is used in a cumulative way; $Pal$ is the maximum detection ratio; and $Pac(a, b)$ denotes the point $a$ to point $b$ part of the pre-avoidance penalty value, which is calculated as:

$$Pac(a, b) = C^{t,2}(a, b-a) + C^{t,3}(a, b-a) + C^{t,4}(a, b-a), \tag{24}$$

where $C^{t,2}(a, b-a)$, $C^{t,3}(a, b-a)$, and $C^{t,4}(a, b-a)$ is given by Eqs. (15, 17, 19).

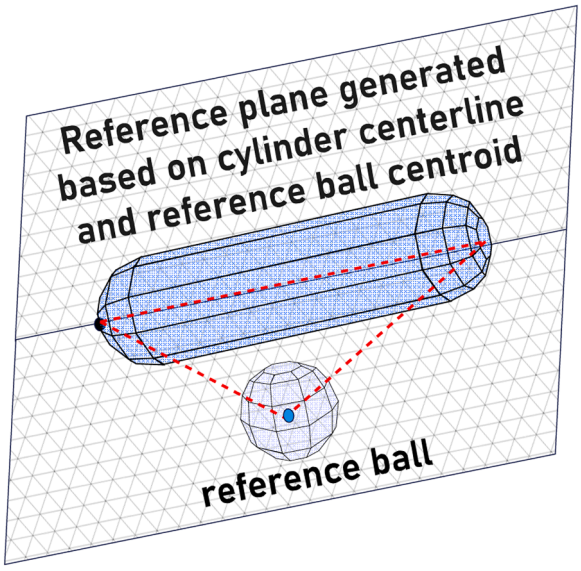


**Fig. 8.** Schematic of the reference surface.

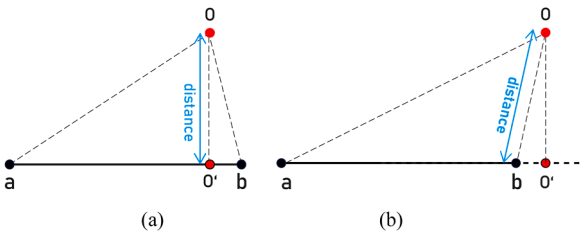


**Fig. 9.** Schematic of the newly defined distances.

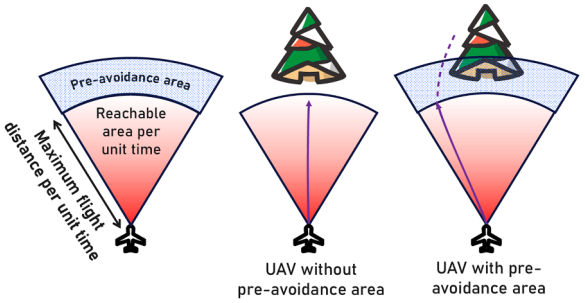


**Fig. 10.** Comparison of UAVs with vs. without a pre-avoidance system.

## 5. Local path generator

To address the autonomous flying of UAV swarms, we discretize time and break the original optimization problem into multiple 3-dimensional problems using the PSCM planning process. This requires a low-dimensional black-box problem solver with fast convergence and limited evaluations. We introduce the "Self-improving and multi-level differentiated creative search algorithm" (SiMlDCS), an enhanced version of the DCS algorithm, as an efficient optimizer for these low-dimensional challenges.

### 5.1. Basic DCS algorithm

The DCS divides the population into three subpopulations and combines a unique knowledge acquisition process with a creative realism paradigm, marking a significant advancement in traditional optimization models. The basic DCS algorithm is illustrated in Fig. 11.

The DSC algorithm's initial position of the $i$th individual is represented as:

$$\mathbf{x}_i^0 = \mathbf{Lb} + \mathbf{r} \cdot (\mathbf{Ub} - \mathbf{Lb}) \quad , \ i = 1, 2, \cdots, N, \tag{25}$$

where **Ub** and **Lb** represent the upper and lower bounds, respectively; and **r** is the D-dimensional random vector in [0,1]; and $N$ is the population size; $\mathbf{x}_i^0$ denotes the initial position of the $i$th individual. We denote the $i$th individual after the $t$th iteration as:

$$\mathbf{x}_i^t = \left(x_{i,1}^t, x_{i,2}^t, ..., x_{i,D}^t\right) \quad , i = 1, 2, \cdots, N, \tag{26}$$

where $x_{i,j}^t$ is the $j$th component of the position of the $i$th individual after the $t$th iteration.

Since there is a special boundary handling and retrospective assessment strategy in the original algorithm, we denote the $i$th generated position without processing during the $t$th iteration as $\mathbf{v}_i^t$, the position processed by the boundary handling strategy as $\widehat{\mathbf{x}}_i^t$,

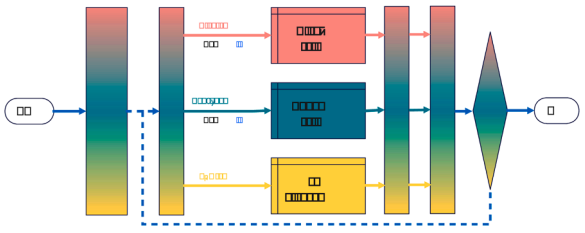

**Fig. 11.** Flowchart of the basic DCS algorithm.

and the position processed by the boundary handling and retrospective assessment strategy is denoted as $\mathbf{x}_i^t$, which is the final position after the $t$th iteration.The expression form of $\mathbf{v}_i^t$ and $\widehat{\mathbf{x}}_i^t$ is similar to $\mathbf{x}_i^t$, which is expressed as:

$$\mathbf{v}_i^t = \left(v_{i,1}^t, v_{i,2}^t, ..., v_{i,D}^t\right) \ , i = 1, 2, \cdots, N, \tag{27}$$

$$\widehat{\mathbf{x}}_i^t = \left(\widehat{x}_{i,1}^t, \widehat{x}_{i,2}^t, ..., \widehat{x}_{i,D}^t\right) \ , i = 1, 2, \cdots, N, \tag{28}$$

where the components are similar in meaning to $x_{i,j}^t$.

In the basic DCS algorithm, the individual number of High Performers is:

$$N_H = \max(6, round(N \times g_r / 3)), \tag{29}$$

where $N$ is the population size, $g_r$ is the golden section ratio, which roughly takes the value of 0.618 and is given by:

$$g_r = \left(1 + \sqrt{5}\right) \Big/ 2, \tag{30}$$

the individual number of Low Performers is:

$$N_L = 1, \tag{31}$$

the individual number of Average Performers is:

$$N_A = N - N_H - N_L, \tag{32}$$

where $N_H$ and $N_L$ is the individual number of High and Low Performers, respectively.

Before each iteration, members of the Average Performers and High Performers subpopulations will adjust their knowledge acquisition rate (Kar) based on their fitness rankings in the population, with the Best Performer at the top. This paper uses these rankings as index values for the current iteration. The Kar is calculated as follows:

$$\begin{cases} \eta = 0.5 \times ([r_1 \times \phi_i] + \langle r_2 < \phi_i \rangle) \\ \phi_i = 0.25 + 0.55 \times \sqrt{i/N} \end{cases} , i = 1, 2, ..., N \ , \tag{33}$$

where $i$ denotes the index of the individual; $\eta_i$ denotes the value of the Kar of an individual whose index (fitness value ranking) is $i$; $\phi_i$ is an intermediate variable; $N$ is the population size; $r_1$, $r_2$ are two random numbers between [0,1]; [•] denotes the rounding to the nearest integer for the values in the parentheses; and ⟨•⟩ denotes a logical value that takes the value of 1 when the conditions given in the parentheses are fulfilled, and 0 otherwise.

In DCS, individuals are categorized into three sub populations based on fitness values, each with a unique position update formula, termed creative realism. The three independent location-updating methods of Creative Realism (CR) are described below.

**Divergent thinking (High Performers):** In the DCS algorithm, High Performers use the Linnik wandering strategy to prevent being trapped in suboptimal positions, which could avoid misleading the evolutionary direction of the population. The position update formula for High Performers is:

$$v_{i,d}^{t+1} = \begin{cases} x_{Ri_i^t,d}^t + Lk(\alpha, \sigma) & r \le \eta_i \text{ or } d = j \\ x_{i,d}^t \end{cases}, \tag{34}$$

where $i$ denotes the index of the individual; $Ri_i^t$ is a index of an individual given by Eq. (35); $\eta_i$ is the Kar rate of the individual indexed (ranked) as $i$, given by Eq. (33); $j$ is a random integer in [1, $D$]; $r$ is a random number between [0, 1]; and $Lk(\alpha, \sigma)$ is the wander step of the Linnik wandering.

$Ri_i^t$ is defined as:

$$Ri_i^t \in \{2, 3, ..., i-1, i+1, ..., N\}, \tag{35}$$

where .. is the population size. According to Eq. (35), $Ri_i^t$ is any individual other than itself and Best Performer.

$Lk(\alpha, \sigma)$ is defined as:

$$Lk(\alpha, \sigma) = \sigma \times \ln\left(\frac{r_1}{r_2}\right) \times \langle r_3 \ge 0.5 \rangle \times \left(\sin\left(\frac{\alpha\pi}{2}\right) \times \tan\left(\frac{(1 - r_4\alpha)}{2}\right) - \cos\left(\frac{\alpha\pi}{2}\right)\right)^{\frac{1}{\alpha}} \tag{36}$$

where α is a shape control parameter and takes the value of 0.618, σ is a constant and takes the value of 0.05, $r_1$, $r_2$, $r_3$, and $r_4$ are random numbers between [0, 1], and ⟨•⟩ denotes a logical value with the value of 1 under the conditions given in the parentheses is satisfied, and 0 otherwise.

The Linnik distribution's characteristic function is:

$$\phi(t) = \frac{1}{1+\sigma^{\alpha}|t|^{\alpha}}, t \in R \quad , \tag{37}$$

where α is a shape control parameter and takes the value of 0.61803.

**Convergent thinking (Average Performers):** In DCS algorithms, Average Performers possess a more conservative mindset. Their characterization is taking the Best Performer as a fixed learning object and updating its position through the DCS/Xbest/Current-to-2rand strategy. The position update formula for Average Performers is:

$$v_{i,d}^{t+1} = \begin{cases} w \times x_{1,d}^{t} + \lambda_t \times \left(x_{R'i_i^t,d}^{t} - x_{i,d}^{t}\right) + \omega_{i,t} \times \left(x_{R''i_i^t,d}^{t} - x_{i,d}^{t}\right) & r \le \eta_i \text{ or } d = j \\ x_{i,d}^{t} & else \end{cases}, \tag{38}$$

where $R'i_i^t$ and $R''i_i^t$ are two indexes of individuals, given by Eq. (39); $\eta_i$ is the Kar rate of the individual ranked as *i; j* is a random integer in [1, *D*]; $\omega_{i,t}$ is a random number between [0, 1]; $\lambda_t$ is a constant for another influencing factor; and *w* takes a fixed value of 1.

$\lambda_t$ is defined as:

$$\lambda_t = 0.1 + 0.618 \times \left(1 - \sqrt{FES_t / FESMAX}\right), \tag{39}$$

where $FES_t$ is the current evaluation times and *FESMAX* is the maximum evaluation times.

$R'i_i^t$ and $R''i_i^t$ is defined as:

$$R'i_i^t \in \{2, 3, \ldots, i-1, i+1, \ldots, N\}, \tag{40}$$

$$R''i_i^t \in \{N_H + 1, \ldots, i-1, i+1, \ldots, N\}, \tag{41}$$

where *N* is the population size, $N_H$ is the subpopulation size of High Performers.

**Team diversification (Low Performers):** In the DCS algorithm, the diversity of the population is maintained by regenerating Low Performers with a certain probability. The position update formula for Low Performers is:

$$\mathbf{v}_i^{t+1} = \begin{cases} \mathbf{Lb} + \mathbf{r} \cdot (\mathbf{Ub} - \mathbf{Lb}) & r' > Pc \\ \mathbf{x}_i^t & else \end{cases}, \tag{42}$$

where **Ub** and **Lb** represent the upper and lower bounds, respectively, and **r** is the random vector in [0,1], *Pc* is the probability constant rate with a default value of 0.5 and $r'$ is a random number between [0,1].

For the new position generated by the CR process, this paper adapts and trades off the position by the following two strategies.

**Rectify the positions:** Solution boundaries represent extreme cases and often indicate impractical or impossible ideas. The DCS adjusts solutions that exceed these boundaries using a specific correction formula:

$$\widehat{x}_{i,d}^{t+1} = \begin{cases} \left(v_{i,d}^{t+1} + Lb_d\right)/2 & v_{i,d}^{t+1} < Lb_d \\ \left(v_{i,d}^{t+1} + Ub_d\right)/2 & v_{i,d}^{t+1} > Ub_d \\ v_{i,d}^{t} & else \end{cases}, \tag{43}$$

where $\widehat{x}_{i,d}^{t+1}$ denotes the rectified position; and $Lb_d$ and $Ub_d$ denote the lower and upper bounds of the optimization problem in the *d*th dimension, respectively.

**Retrospective assessment (RA):** In the original DCS algorithm, the retrospective assessment strategy is defined as a separate strategy process, and the retrospective assessment's treatment of the rectified generated position is:

$$\mathbf{x}_i^{t+1} = \begin{cases} \widehat{\mathbf{x}}_i^{t+1} & Fit\left(\widehat{\mathbf{x}}_i^{t+1}\right) < Fit\left(\mathbf{x}_i^t\right) \\ \mathbf{x}_i^t & Fit\left(\widehat{\mathbf{x}}_i^{t+1}\right) \ge Fit\left(\mathbf{x}_i^t\right) \end{cases}, \tag{44}$$

where $Fit\left(\mathbf{x}_i^t\right)$ denotes the fitness value of the $\mathbf{x}_i^t$, and $\widehat{\mathbf{x}}_i^t$ and $\mathbf{x}_i^t$ are the same as defined above.

During the iteration process, if the fitness value of a generated position is smaller than the fitness value of the Best Performers, the Best Performers are immediately set to that newly generated position, which the following equation can describe:

$$\mathbf{x}_{Best}^{t+1} = \begin{cases} \mathbf{x}_1^t & Start \\ \mathbf{x}_i^{t+1} & Fit\left(\mathbf{x}_i^{t+1}\right) < Fit\left(\mathbf{x}_{Best}^{t+1}\right) \\ \mathbf{x}_{Best}^{t+1} & else \end{cases} . \tag{45}$$

### 5.2. SiMlDCS algorithm

The DCS, leveraging the strategy combination of multiple DE algorithms, maintains the rapid convergence of DE while assigning specific optimization tasks to individuals. However, the original DCS algorithm struggles with slow convergence, instability, and evaluation inefficiencies when applied to the dynamic real-time local path search for UAVs. To address these issues, this paper proposes improvements to the DCS algorithm to enhance its convergence speed and optimize results effectively without excessive evaluation waste. Additionally, we adjust the population structure of the DCS to meet varying population requirements and improve the role of Average Performers, resulting in the improved algorithm named SiMlDCS. The improvements to the basic DCS algorithm in this paper are as follows:

#### 5.2.1. Population initialization base on HaltonRR2 (Co)prime numbers sequences

Due to the limited evaluation and iteration numbers for the optimization problem in this paper, we use the HaltonRR2 prime sequence of quasi-random number generators (QRNGs) for population initialization, ensuring non-crowded initial positions [61].

An initial set of prime bases, denoted $b_1$,$b_2$,…, $b_S$, is determined before generating the Halton prime sequence, and the Halton prime sequence generated based on this prime base is defined as:

$$x_n = \left(\Phi_{b_1}(n), \Phi_{b_2}(n), \ldots, \Phi_{b_S}(n)\right), \tag{46}$$

where $\Phi_{b_j}(n)$ is the $j$th radical inverse function, which is defined as:

$$\Phi_{b_j}(n) = \sum_i a_i(j,n) b_j^{-i-1}, \tag{47}$$

where $a_i(j,n)$ is the coefficient of the prime basis $b_1$,$b_2$,…, $b_S$ of the $j$th radial inverse function integer of the sequence of Halton primes corresponding to integer $n$, which is given by the expansion of integer n based on the prime basis, which is defined as:

$$n = \sum_{i=0} a_i(j,n) b_j^i \,. \tag{48}$$

#### 5.2.2. Self-improvement team diversification method

In this paper, referring to the nutcracker food search process, we replace team diversification to ensure that Low Performers maintain diversity while developing self-learning abilities to explore new opportunities based on their past experiences effectively [62]. We refer to this improved strategy as the Self-Improvement Team Diversification Method.

First, we define the learning rate factor $a$:

$$a(t) = \begin{cases} \left(\dfrac{t}{T\max}\right)^{\frac{2}{t}} & r_1 < r_2 \\ \left(1 - \dfrac{t}{T\max}\right)^{\frac{2\times t}{T\max}} & else \end{cases}, \tag{49}$$

where $t$ is the current iteration number, $T$max is the maximum iteration number, and $r_1$, $r_2$ are two random numbers between [0, 1].

The positions of the two uncorrected generated food hiding spots are:

$$\widehat{HF}_{1,d}^{t+1} = \begin{cases} x_{i,d}^t + \left(a \times \cos(\phi) \times \left(x_{R'i_i^t,d}^t - x_{R''i_i^t,d}^t\right)\right) & \phi \neq 90^\circ \\ x_{i,d}^t + \left(a \times \cos(\phi) \times \left(x_{R'i_i^t,d}^t - x_{R''i_i^t,d}^t\right)\right) + a \times HF_{1+\langle r_1 > r_2\rangle,d}^t & else \end{cases}, \tag{50}$$

$$\widehat{HF}_{2,d}^{t+1} = \begin{cases} x_{i,d}^t + (a \times \cos(\phi) \times (Lb_d + r_1(Ub_d - Lb_d))) \times \langle r_2 < 0.2\rangle & \phi \neq 90^\circ \\ x_{i,d}^t + (a \times \cos(\phi) \times (Lb_d + r_3(Ub_d - Lb_d))) + a \times HF_{1+\langle r_4 > r_5\rangle,d}^t \times \langle r_6 < 0.2\rangle & else \end{cases}, \tag{51}$$

where $\widehat{HF}_{1,d}^{t+1}$ and $\widehat{HF}_{2,d}^{t+1}$ denote the positions of the uncorrected generated hidden food points; $a$ is the learning rate factor, given by Eq. (49); ϕ is a random number between [0, π]; $\langle\bullet\rangle$ is a logical variable which takes 1 when the condition in the parentheses is satisfied, and 0 for the rest; $R'i_i^t$, $R''i_i^t$ are the positions of the two agents randomly selected by the individual of $i$ during the $t$th iteration, given by Eqs. (52, 53); $r_1$, $r_2$, $r_3$, $r_4$, $r_5$, and $r_6$ are random numbers between [0, 1]; $d$ is the dimension of the problem, where $d \in \{1, 2, \ldots, D\}$.

$R'i_i^t$ and $R''i_i^t$ is defined as:

$$R'i_i^t \in \{1, 2, \ldots, N\}, \tag{52}$$

$$R''i_i^t \in \{1, 2, \ldots, N\}, \tag{53}$$

where $N$ is the population size.

The corrected expression for the positions of the two generated hiding food spots is:

$$HF_{1,d}^{t+1}=\begin{cases}\widehat{HF}_{1,d}^{t+1} & Lb_d\le\widehat{HF}_{1,d}^{t+1}\le Ub_d\\ \max\left(Lb_d,\min\left(Ub_d,\widehat{HF}_{1,d}^{t+1}\right)\right) & R_1>R_2\quad and\quad \left(\widehat{HF}_{1,d}^{t+1}\le Lb_d\ or\ \widehat{HF}_{1,d}^{t+1}\ge Ub_d\right)\\ Lb_d+r\times(Ub_d-Lb_d) & R_1\le R_2\quad and\quad \left(\widehat{HF}_{1,d}^{t+1}\le Lb_d\ or\ \widehat{HF}_{1,d}^{t+1}\ge Ub_d\right)\end{cases}, \tag{54}$$

$$HF_{2,d}^{t+1}=\begin{cases}\widehat{HF}_{2,d}^{t+1} & Lb_d\le\widehat{HF}_{2,d}^{t+1}\le Ub_d\\ \max\left(Lb_d,\min\left(Ub_d,\widehat{HF}_{2,d}^{t+1}\right)\right) & R_3>R_4\quad and\quad \left(\widehat{HF}_{2,d}^{t+1}\le Lb_d\ or\ \widehat{HF}_{2,d}^{t+1}\ge Ub_d\right)\\ Lb_d+r\times(Ub_d-Lb_d) & R_3\le R_4\quad and\quad \left(\widehat{HF}_{2,d}^{t+1}\le Lb_d\ or\ \widehat{HF}_{2,d}^{t+1}\ge Ub_d\right)\end{cases}, \tag{55}$$

where $\widehat{HF}_{1,d}^{t+1}$ and $\widehat{HF}_{2,d}^{t+1}$ denote the positions of the uncorrected generated hidden food points; $HF_{1,d}^{t+1}$ and $HF_{2,d}^{t+1}$ denote the positions of the corrected generated hidden food points; $r$ is a random number between [0, 1]; $R_1$, $R_2$, $R_3$, and $R_4$are four random numbers between [0,1] which are randomly generated before the population individuals are updated; $Lb_d$ and $Ub_d$ denote the $d$th dimension of the upper and lower bounds of the optimization problem, respectively.

Thus, the position update formula for Low Performers is:

$$v_{i,d}^{t+1}=\begin{cases}x_{i,d}^{t}+r_1\times\left(x_{1,d}^{t}-x_{i,d}^{t}\right)+r_2\times\left(HF_{1,d}^{t+1}-x_{R'i_i^t,d}^{t}\right) & r_5\le r_6\ and\ R_1\le R_2\\ x_{i,d}^{t}+r_3\times\left(x_{1,d}^{t}-x_{i,d}^{t}\right)+r_4\times\left(HF_{2,d}^{t+1}-x_{R''i_i^t,d}^{t}\right) & r_7\le r_8\ and\ R_1>R_2\\ x_{i,d}^{t} & else\end{cases}, \tag{56}$$

where $r_1$, $r_2$, $r_3$, $r_4$, $r_5$, $r_6$, $r_7$, and $r_8$ are random numbers between [0, 1]; $R_1$, and $R_2$are random numbers between [0,1] which are randomly generated before the population individuals are updated; and $R'i_i^t\in\{1,2,\ldots,N\}$ and $R''i_i^t\in\{1,2,\ldots,N\}$.

### 5.2.3. Multi-level evolving strategy (hybrid community-based and target-based evolution)

To enhance convergence speed, DCS/Xbest/Current-to-2rand incorporates a local adjustment strategy called the community-based evolution method. Inspired by Cymerys and Oszust's localized attraction-repulsion operator [63], a similar model is introduced in this paper. The adjusted position update formula for the Average Performers is:

$$v_{i,d}^{t+1}=\begin{cases}w\times x_{1,d}^{t}+\lambda_t\times\left(x_{R'i_i^t,d}^{t}-x_{i,d}^{t}\right)+\omega_{i,t}\times\left(x_{R''i_i^t,d}^{t}-x_{i,d}^{t}\right) & r\le\eta_i\ \text{or}\ d=j\\ x_{i,d}^{t}+\beta\times Ci_{i,d}^{t}\left(\mathbf{x}_i^t\right) & else\end{cases}, \tag{57}$$

where $R'i_i^t$ and $R''i_i^t$ are two indexes of individuals that are randomly generated before each Average Performers member generates a new position; $\eta_i$ is the Kar rate of the individual indexed (ranked) as $i$; $j$ is a random integer in [1, $D$]; $\omega_{i,t}$ is an influence factor that takes a random number between [0, 1]; $\lambda_t$ is a constant for another influence factor; $w$ takes a fixed value of 1; $\beta$ is the community influence factor; and $Ci_{i,d}^t$ is the community influence received by the individual of the $i$th population during the $t$th iteration, calculated as:

$$Ci_{i,d}^{t}\left(\mathbf{x}_i^t\right)=\sum_{j=1}^{k}\left[\left(x_{i,d}^{t}-x_{I(j),d}^{t}\right)\times\left(1-\frac{D(i,I(j))}{\sup_{m\in\{1,2,\ldots,N\}}|D(i,m)|}\right)\times sign\left(Fit\left(\mathbf{x}_j^t\right)-Fit\left(\mathbf{x}_{I(j)}^t\right)\right)\right], \tag{58}$$

where $k$ denotes the number of neighbors affecting the $i$th population individual; $I(j)$ denotes the index of the $j$th neighbor among these $k$ neighbors; $D(a, b)$ denotes the Euclidean distance of the $a$th individual from the $b$th individual; $sign(\bullet)$ denotes the sign that takes the value in parentheses; and $Fit\left(\mathbf{x}_e^t\right)$ denotes the fitness value that corresponds to the position of the $e$th population individual after the $t$th iteration of its own position.

The number $k$ of neighbors affecting individuals of a single population is given by:

$$k=ceil\left(\left(1-\frac{t}{T\max}\right)\times N\right), \tag{59}$$

where $t$ is the current iteration number; $T$max is the maximum iteration number; $N$ denote the population size; $ceil(\bullet)$ indicates that the value in parentheses is rounded up to the nearest integer.

### 5.2.4. Rebalancing population structure

Since SiMlDCS is designed to generate the local paths for UAVs with different performances, and different kinds of UAVs have different performances in computation time and memory, etc. Therefore we redesigned the number composition of each group in the population.

The individual number of High Performers is:

$$N_H=2. \tag{60}$$

The individual number of Low Performers is:

$$N_L = 0.95 \times N, \tag{61}$$

where $N$ is the population size. The individual number of Average Performers is:

$$N_A = N - N_H - N_L, \tag{62}$$

where $N_H$ and $N_L$ is the individual number of High Performers and Low Performers,

### 5.3. Local path generator based on SiMlDCS

We use the SiMlDCS algorithm and the local path evaluator to work together to generate local paths, and the pseudo-code for this process is shown in Algorithm 1.

## 6. Experiments and analysis

In sections 2-5, we described a multi-UAV swarm 4D collision-free path planning method, which decomposes the original high-dimensional optimization problem into multiple 3D optimization problems.

### 6.1. Numerical experiment

We tested SiMlDCS using the CEC2022 benchmark suite, which contains 12 minimization test functions. Each algorithm was run 30 times independently under consistent environmental settings. The experimental setup used "Windows 10 (64-bit) 23H2," and "Matlab2023b."

According to NFL theory, no single algorithm can be applied to all optimization problems [91]. We selected 10 algorithms to be used in this paper for the most comparison and fairly test the performance of SiMlDCS algorithm according to the same evaluation criterion. These algorithms for comparison include LSHADE_cnEpSin, LSHADE_SPACMA, AROA, EO, AHA, EEFO, GKSO, MSA [59, 63–70]. The selection of the comparison algorithms are some recently proposed and state-of-art classical metaheuristic algorithms. The SiMlDCS algorithm is designed for 3D problems, so the CEC2022 test set was limited to 10 dimensions. To enhance convergence speed while managing memory and evaluations, the maximum number of evaluations was set to 10,000. The parameters for the remaining

**Algorithm 1**
local path generator base on SiMlDCS algorithm and local path evaluator.

**Input relevant information of practical problems:** Environment, information of the current UAV and other UAVs.
**Input relevant parameters of SiMlDCS algorithm:** Dimension $D$, population size $N$, maximum iterations $T_{max}$.
**Output:** Approximate optimal local path.
**Local path generator start:**
  Generate $N$ local paths candidate by Eq. (46, 47, 48).
  Evaluate the fitness value of $N$ local paths by Eq. (11), where Eq. (11) is calculated by Eq. (13, 15, 17, 19, 21, 23).
  Rank the $N$ local paths by the fitness value of $N$ local paths.
  Divide the $N$ local paths into three subgroups by Eq. (60, 61, 62),
    name them as High Performer, Average Performer and Low Performer.
  **While** $t < T_{max}$ ($t$ denotes the current iteration count)
    Calculating knowledge acquisition rates(Kar) by Eq. (33).
    **For** $i$=1:$N$ ($t$ denotes the index of candidate local path)
      **If** local path candidate $i$ belongs to High Performers
        Update local path candidate $i$ base on Kar by Eq. (34),
          where Eq. (34) is calculated by Eq. (35, 36, 37).
        **End if**
      **If** local path candidate $i$ belongs to Average Performers
        Update local path candidate $i$ base on Kar by Eq. (57),
          where Eq. (57) is calculated by Eq. (39, 40, 41, 58, 59).
        **End if**
        **If** local path candidate $i$ belongs to Low Performers
        Update local path candidate $i$ base on Kar by Eq. (56),
          where Eq. (56) is calculated by Eq. (49, 50, 51, 52, 53, 54,55).
        **End if**
        Further processing of the updated local path candidate by Eq. (43).
    **End for**
    Evaluate the fitness value of $N$ updated local paths by Eq. (11),
      where Eq. (11) is calculated by Eq. (13, 15, 17, 19, 21, 23).
    Implementing the retrospective assessment by Eq. (44).
    Rank the $N$ updated local paths by the fitness value of $N$ local paths.
    Divide the $N$ updated local paths into three subgroups by Eq. (60, 61, 62),
      name them as High Performer, Average Performer and Low Performer.
  **End while**
  Output best candidate local path.
**Local path generator end.**

algorithms, including SiMlDCS, are adjusted for memory and iteration balance, with both set to 100 to align with real-world scenarios.

The average optimal results of all algorithms in 10D CEC2022 are shown in Table 2, highlighting the top three performing algorithms for each test function in bold. Arrows indicate comparisons between the proposed SiMlDCS and the original DCS algorithm, with upward arrows signifying better performance by SiMlDCS and downward arrows indicating worse performance.

Table 2 shows that SiMlDCS achieved the top three optimization accuracies on 91.7% of the test functions. Both DCS and SiMlDCS performed at an average level on F11, possibly due to algorithm structure and test function characteristics, as suggested by NFL theory. SiMlDCS improved optimization performance on 83.3% of the test functions compared to the original DCS algorithm.

In repeated experiments, algorithm stability is indicated by the standard deviation of optimal results; a more minor standard deviation signifies greater stability. Table 3 displays the standard deviation results for 30 experimental repetitions under 10D CEC2022, highlighting the top three most stable algorithms in bold. Arrows indicate comparisons between the proposed SiMlDCS and the original DCS algorithm: upward arrows show that SiMlDCS is more stable, while downward arrows indicate less stability.

Table 3 shows that SiMlDCS achieves the best stability results on half of the test functions, surpassing other compared algorithms. It improves upon the original DCS algorithm's stability on 75% of the test functions, making it more suitable for real-world applications.

Fig. 12 shows the convergence curves of the average error values of 10 comparison algorithms under F1, F3, F7, and F10. These four test functions represent the single-modal function, multi-modal function, hybrid function, and combination function in CEC2022, respectively. SiMlDCS showed fast convergence speed in these test functions. In particular, on F3, the average error value of SiMiDCS was less than 0.001, which was significantly better than the 0.01 of the second-best algorithm EO.

### 6.2. Scenario simulation experiment

We optimize the dynamic LPSA for UAV swarms using the SiMlDCS algorithm and compare the suitability of DCS, AROA, AHA, EEFO, EO, GKSO, and MSA under different scenarios in this subsection. The population size is set to 30 to reflect real-world memory limitations, and the evaluation count is set to 900 to represent UAV computational constraints.

#### 6.2.1. Environment and parameter setting

We present two experimental scenarios for UAV swarms. The first scenario (as in Fig. 13) is a field combat setting, characterized by plains, hills, and static obstacle areas simulating hazards like air defense radars. The second scenario (as in Fig. 14) involves urban transportation, featuring a flat terrain with numerous immovable buildings integrated into the terrain data. Scenario 2 poses greater challenges for the local path planner due to the increased terrain complexity, despite the absence of static obstacles. All experimental units are standardized, and the length unit is standardized as $m$.

In Scenario 1, the horizontal extent of the experimental environment was set to $(1, 513) \times (1, 513)$, and the vertical extent was set to $(0, 256)$. The terrain environment was randomly generated with consistent data across all algorithms. The terrain data and generation algorithms can be downloaded from Supplementary Materials. A 3D view is presented in Fig. 3.

We created an experimental environment with 13 static cylindrical obstacle regions to simulate flight exclusion zones, as shown in Fig. 15. Each obstacle has a known radius and infinite height, with a safety distance equal to the body radius of 3. Their positions and sizes are detailed in Table 4.

In Scenario 2, the horizontal extent of the experimental environment was set to $1000 \times 1000$ and the vertical extent was set to $Pn_a$. The terrain environment was randomly generated with the same terrain data for all algorithms, and the terrain data is available for download at Supplementary Materials. A 3D view of the terrain is shown in Fig. 16.

The experimental environment features several UAVs. In Scenario 1, the UAV radius is 3, which is sufficient to simulate various real-life products. For the larger map in Scenario 2, the UAV radius is increased to 5. The pitch angle of all the UAVs is maximized to $\pi/2$, the flying turn direction is set to omnidirectional, and the magnitude of the velocity per unit of time to (2, 10).

In Scenario 2, integrating building information with terrain reveals significant undulations, increasing the risk of UAV collisions with the terrain. As illustrated in Fig. 17, part of the UAV's reference ball may already penetrate the building despite it being elevated from the ground. To prevent this, we preprocess the DGM model matrices using a maximal mask to expand the building's footprint horizontally, thus reducing collision risks. This is demonstrated in Fig. 18, which shows the unprocessed terrain on the left and the significantly enlarged processed terrain on the right.

According to the description of the PSCM process in Section 3, each UAV generates local paths in sequence and places temporary obstacle regions in space, so the difficulties of path planning increase with the number of UAVs.

To enable a fair measure of the ability of various optimization algorithms to optimize the dynamic local path planner for this UAV swarm, we separated the process and compared the performance strength of various local path search algorithms by counting the path of the $n + 1$th UAV (abbreviated as test UAV, described in detail in the next sentence) in multiple iterations based on controlling the same path of the first $n$ UAV (abbreviated as interference UAVs, described in detail in the next sentence), as shown in Fig. 19.

The UAVs in the solution region are divided into two kinds, one is the interference UAVs, these UAVs fly back and forth between the regions along different straight lines, which are used to interfere with the newly generated local paths of the UAVs, and the other is the test UAVs, which plan the local paths independently in each time period according to the environmental information and the paths of the UAVs, and record their trace information in real time, which is used to test the planning ability of the local path search algorithm.

There are 15 interference UAVs designed to maximize their impact on test UAVs by flying nearly perpendicular to their direction. Each jamming UAV starts at a fixed speed and reverses its direction if it's likely to exceed a preset limit. Fig. 20 illustrates the flight paths of the interfering UAVs in Scenario 1, showing 12 recorded trails at 5 time intervals. The red dashed lines represent the UAV trajectories, while the red solid areas indicate the temporary static obstacles created by each UAV during this period. Table 5 provides

**Table 2**
The comparison results of mean under 10D CEC2022.

| Algorithms | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 | F10 | F11 | F12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Original DCS Algorithm: | | | | | | | | | | | | |
| DCS | 330.379 | 407.871 | **600.117** | 830.950 | 900.207 | 3494.410 | 2025.434 | 2223.322 | 2529.285 | 2500.473 | 2708.075 | **2861.367** |
| **Enhanced DCS Algorithm Proposed In This Paper:** | | | | | | | | | | | | |
| **SiMlDCS** | **300.450↑** | **404.229↑** | **600.000↑** | **806.760↑** | **900.011↑** | **1829.058↑** | **2020.204↑** | **2220.301↑** | **2529.284↑** | **2500.391↑** | 2888.332↓ | **2863.275↓** |
| Comparison Algorithms For Testing Algorithm Performance: | | | | | | | | | | | | |
| LSHADE_cnEpSin | **300.012** | 407.021 | 600.199 | 821.732 | **900.019** | **1810.376** | 2033.947 | 2224.996 | **2528.490** | **2500.423** | 2862.041 | 2864.114 |
| LSHADE_SPACMA | 301.655 | 406.612 | 600.600 | 825.778 | 900.133 | **1874.452** | 2030.764 | 2226.341 | 2529.290 | 2500.477 | **2698.193** | 2864.240 |
| AROA | 301.688 | **403.902** | 600.207 | **812.438** | 907.200 | 2591.438 | **2020.405** | **2220.178** | 2529.284 | **2500.473** | 2895.359 | 2864.106 |
| EO | 365.619 | 412.316 | **600.011** | **810.487** | **900.070** | 4792.191 | **2020.937** | 2222.227 | **2529.284** | 2525.992 | 2793.626 | **2863.303** |
| AHA | 2284.142 | **406.120** | 601.510 | 823.840 | 902.995 | 2623.972 | 2020.984 | 2222.409 | 2539.653 | 2500.523 | 2720.111 | 2868.729 |
| EEFO | 1556.350 | 407.792 | 601.339 | 815.817 | 902.612 | 4172.014 | 2023.993 | 2226.121 | 2533.514 | 2500.586 | **2683.283** | 2867.890 |
| GKSO | **300.015** | 407.547 | 603.128 | 820.861 | 907.298 | 3000.593 | 2024.886 | **2221.921** | 2529.289 | 2508.737 | 2728.097 | 2864.442 |
| MSA | 1228.021 | 406.478 | 601.089 | 823.665 | 901.697 | 3970.499 | 2029.687 | 2224.462 | 2531.989 | 2500.519 | **2667.021** | 2866.750 |

**Table 3**
The comparison results of standard deviation under 10D CEC2022.

| Algorithms | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 | F10 | F11 | F12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Original DCS Algorithm: | | | | | | | | | | | | |
| DCS | 2.468E+01 | **1.170E+00** | **4.163E-02** | 6.600E+00 | 2.083E-01 | 1.809E+03 | **1.956E+00** | **3.754E+00** | 3.511E-04 | 8.923E-02 | 1.167E+02 | 1.453E+00 |
| **Enhanced DCS Algorithm Proposed In This Paper:** | | | | | | | | | | | | |
| **SiMlDCS** | **4.386E-01↑** | 3.650E+00↓ | **8.244E-05↑** | **4.941E+00↑** | **3.967E-02↑** | **2.116E+01↑** | 7.042E+00↓ | 6.119E+00↓ | **7.255E-12↑** | **7.908E-02↑** | 5.522E+01↑ | 1.308E+00↑ |
| Comparison Algorithms For Testing Algorithm Performance: | | | | | | | | | | | | |
| LSHADE_cnEpSin | **9.380E-03** | **2.615E+00** | 8.005E-02 | **3.574E+00** | **2.389E-02** | **6.358E+00** | **4.675E+00** | 4.309E+00 | 1.507E+00 | **8.046E-02** | 1.026E+02 | 1.072E+00 |
| LSHADE_SPACMA | 2.093E+00 | **2.449E+00** | 2.079E-01 | 5.350E+00 | **1.200E-01** | **2.722E+01** | **3.291E+00** | 4.208E+00 | 2.843E-03 | **8.075E-02** | 1.240E+02 | **9.962E-01** |
| AROA | 5.265E+00 | 1.207E+01 | 5.267E-01 | 5.645E+00 | 7.547E+00 | 1.071E+03 | 8.912E+00 | 5.293E+00 | **1.209E-04** | 1.154E-01 | **5.450E+01** | 2.097E+00 |
| EO | 6.597E+01 | 1.626E+01 | **1.125E-02** | **4.379E+00** | 1.555E-01 | 2.330E+03 | 5.409E+00 | 5.008E+00 | **1.447E-05** | 4.726E+01 | 1.434E+02 | **9.771E-01** |
| AHA | 1.127E+03 | 4.539E+00 | 7.737E-01 | 6.234E+00 | 3.512E+00 | 1.044E+03 | 6.778E+00 | **3.851E+00** | 7.752E+00 | 1.056E-01 | **2.887E+01** | 2.157E+00 |
| EEFO | 7.019E+02 | 1.035E+01 | 8.369E-01 | 5.481E+00 | 3.412E+00 | 3.400E+03 | 7.186E+00 | **3.314E+00** | 2.808E+00 | 1.391E-01 | 8.118E+01 | 1.646E+00 |
| GKSO | **2.062E-02** | 1.335E+01 | 3.293E+00 | 8.513E+00 | 1.143E+01 | 1.434E+03 | 9.346E+00 | 4.973E+00 | 7.695E-03 | 3.153E+01 | 1.561E+02 | **9.970E-01** |
| MSA | 4.690E+02 | 1.233E+01 | 4.383E-01 | 6.707E+00 | 1.548E+00 | 2.154E+03 | 4.903E+00 | 4.877E+00 | 1.530E+00 | 8.794E-02 | **3.773E+01** | 1.393E+00 |

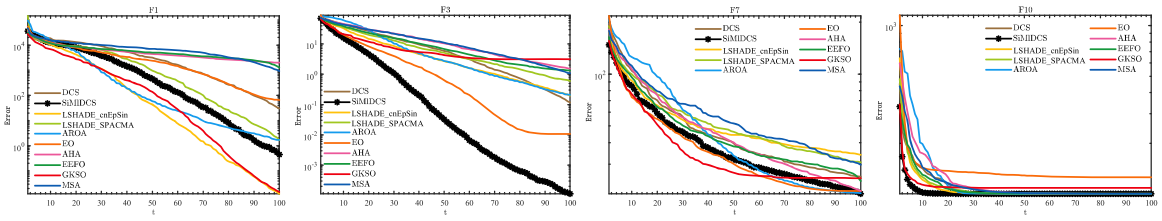


**Fig. 12.** Average error convergence curves.

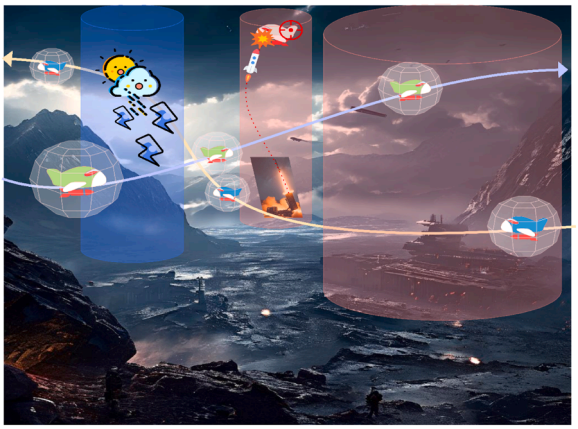

**Fig. 13.** Schematic of field combat scenario.

**Fig. 14.** Schematic of urban transportation scenario.

the relevant information about the 15 interfering UAVs in Scenario 1 and Scenario 2.

The map features multiple sets of starting and ending points, with each set representing a mission. Each scenario includes four distinct missions. In Scenario 1, the interference UAVs are positioned at varying heights between [80, 120] in a wavy pattern, while mission heights are uniform at 100. The vertical relationship between interference UAVs and missions is illustrated in Fig. 21 (taking Scenario 1 as an example), where the blue dashed line shows the projection of mission start points on the x-plane, and the red ball represents the unchanged projection of the interfering UAV.

To minimize the effect of random factors, each path-planning algorithm independently conducts 30 repetitive experiments; that is, 30 identical UAVs are used to implement the same mission simultaneously, and their position information at each moment is recorded independently to analyze the path-planning results. Table 6 shows the information related to the four missions and the 120 test UAVs. In Scenario 2, the altitude of the interfering UAV is located between [40,60] , the altitude of the city ground is set to 0, *Hsafe* to 20, and the mission altitude is set to 40; this altitude can let the floors with smaller heights influencing the UAVs on one hand, and on the other hand it can also check the effect of the adjustment of the *Hsafe* for the path of the UAVs.

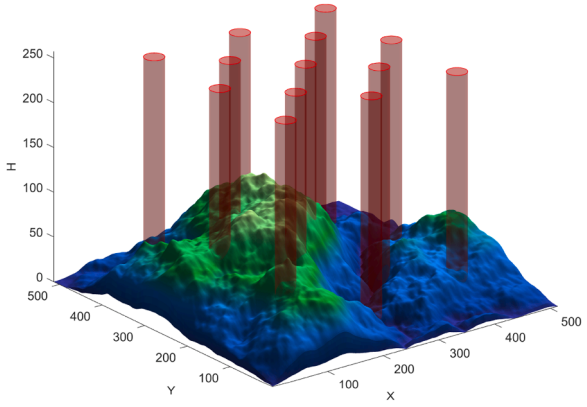


**Fig. 15.** 3D view of field combat scenario.

**Table 4**
Information about the 13 static obstacles in Scenario 1.

| *id* | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $Pn_a$ | 102.6 | 102.6 | 410.4 | 410.4 | 179.55 | 179.55 | 333.45 | 333.45 | 256.5 | 256.5 | 256.5 | 410.4 | 102.6 |
| $Pn_a$ | 410.4 | 102.6 | 102.6 | 410.4 | 333.45 | 179.55 | 184.68 | 333.45 | 256.5 | 410.4 | 102.6 | 256.5 | 256.5 |
| $Pn_a$ | 15.39 | 15.39 | 15.39 | 15.39 | 15.39 | 15.39 | 15.39 | 15.39 | 15.39 | 15.39 | 15.39 | 15.39 | 15.39 |

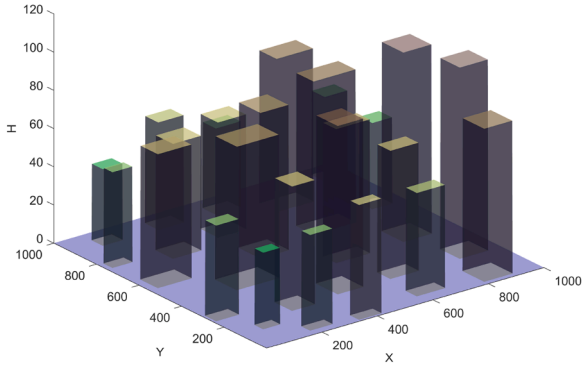


**Fig. 16.** 3D view of urban transportation scenario.

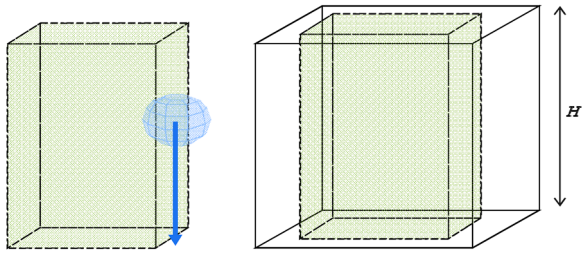


**Fig. 17.** Threats and solutions due to undulating terrain.

The default values of the other parameters in Section 4 are shown in Table 7, where "or" indicates the different values in the two scenarios in the later experiments.

### *6.2.2. Standards for analyzing the results of experiments*

This paper evaluates the optimization capability of each algorithm using performance indicators for the proposed 4D collision-free dynamic LPSA:

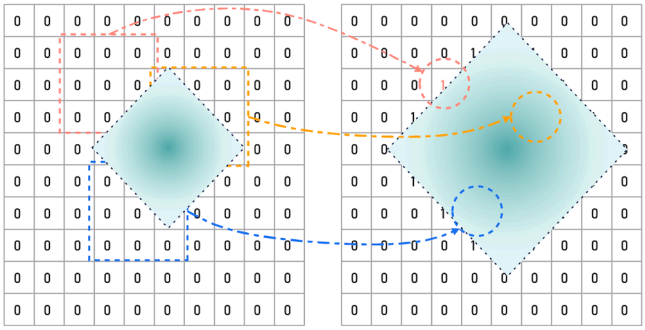

**Fig. 18.** Maximum mask expanding the terrain.

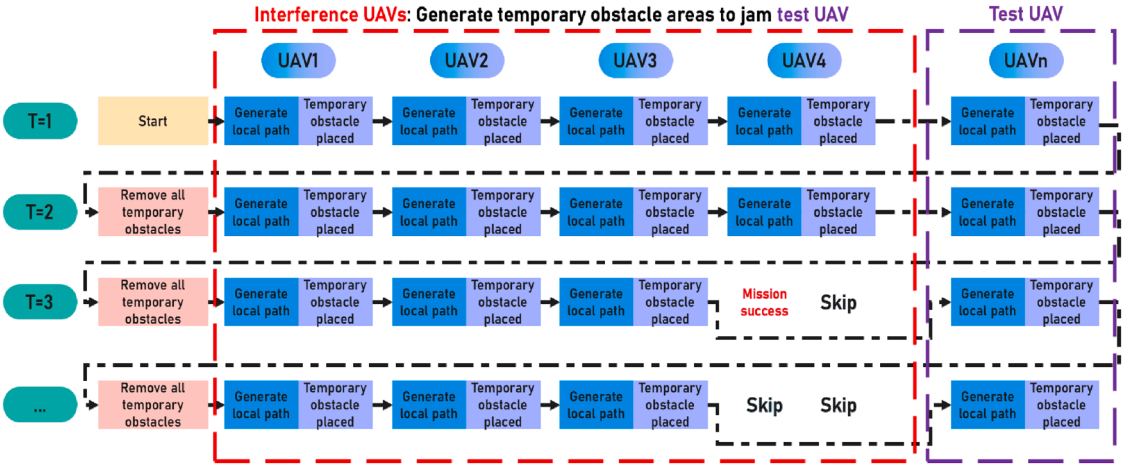


**Fig. 19.** Schema of the principles of the scenario simulation experiment.

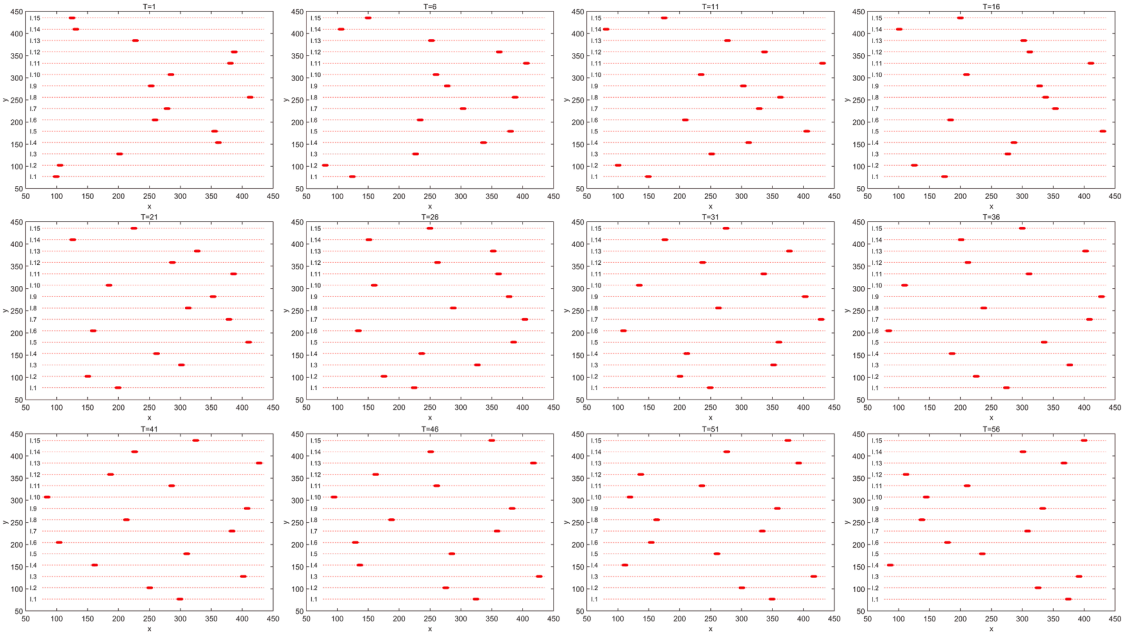


**Fig. 20.** Top view of interference in Scenario 1.

**Table 5**
Information about the interfering UAVs in Scenario 1 and Scenario 2.

| Scenario 1:battlefield in the wilderness | | | | | Scenario 2:urban transportation | | | | |
|---|---|---|---|---|---|---|---|---|---|
| IUAV | Starting Position | Velocity | One Boundary | Another Boundary | IUAV | Starting Position | Velocity | One Boundary | Another Boundary |
| I.1 | [76.80,76.80,80] | [5,0,0] | [76.80,76.80,80] | [435.200,76.80,80] | I.1 | [150,150,40] | [15,0,0] | [76.800,150,40] | [435.20,150,40] |
| I.2 | [128,102.400,100] | [-5,0,0] | [76.80,102.400,100] | [435.200,102.400,100] | I.2 | [250,200,50] | [-15,0,0] | [76.800,200,50] | [435.20,200,50] |
| I.3 | [179.200,128,120] | [5,0,0] | [76.80,128,120] | [435.200,128,120] | I.3 | [350,250,60] | [15,0,0] | [76.800,250,60] | [435.20,250,60] |
| I.4 | [384,153.600,120] | [-5,0,0] | [76.80,153.600,120] | [435.200,153.600,120] | I.4 | [750,300,60] | [-15,0,0] | [76.800,300,60] | [435.20,300,60] |
| I.5 | [332.800,179.200,100] | [5,0,0] | [76.80,179.200,100] | [435.200,179.200,100] | I.5 | [650,350,50] | [15,0,0] | [76.800,350,50] | [435.20,350,50] |
| I.6 | [281.600,204.800,80] | [-5,0,0] | [76.80,204.800,80] | [435.200,204.800,80] | I.6 | [550,400,40] | [-15,0,0] | [76.800,400,40] | [435.20,400,40] |
| I.7 | [256,230.400,80] | [5,0,0] | [76.80,230.400,80] | [435.200,230.400,80] | I.7 | [500,450,40] | [15,0,0] | [76.800,450,40] | [435.20,450,40] |
| I.8 | [435.200,256,100] | [-5,0,0] | [76.80,256,100] | [435.200,256,100] | I.8 | [850,500,50] | [-15,0,0] | [76.800,500,50] | [435.20,500,50] |
| I.9 | [230.400,281.600,120] | [5,0,0] | [76.80,281.600,120] | [435.200,281.600,120] | I.9 | [450,550,60] | [15,0,0] | [76.800,550,60] | [435.20,550,60] |
| I.10 | [307.200,307.200,120] | [-5,0,0] | [76.80,307.200,120] | [435.200,307.200,120] | I.10 | [600,600,60] | [-15,0,0] | [76.800,600,60] | [435.20,600,60] |
| I.11 | [358.400,332.800,100] | [5,0,0] | [76.80,332.800,100] | [435.200,332.800,100] | I.11 | [700,650,50] | [15,0,0] | [76.800,650,50] | [435.20,650,50] |
| I.12 | [409.600,358.400,80] | [-5,0,0] | [76.80,358.400,80] | [435.200,358.400,80] | I.12 | [800,700,40] | [-15,0,0] | [76.800,700,40] | [435.20,700,40] |
| I.13 | [204.800,384,80] | [5,0,0] | [76.80,384,80] | [435.200,384,80] | I.13 | [400,750,40] | [15,0,0] | [76.800,750,40] | [435.20,750,40] |
| I.14 | [153.600,409.600,100] | [-5,0,0] | [76.80,409.600,100] | [435.200,409.600,100] | I.14 | [300,800,50] | [-15,0,0] | [76.800,800,50] | [435.20,800,50] |
| I.15 | [102.400,435.200,120] | [5,0,0] | [76.80,435.200,120] | [435.200,435.200,120] | I.15 | [200,850,60] | [15,0,0] | [76.800,850,60] | [435.20,850,60] |

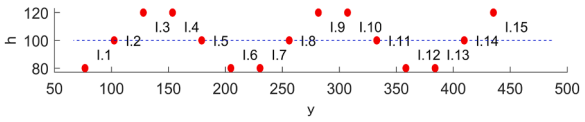


**Fig. 21.** Side view of the path of the interfering UAVs in Scenario 1 with the mission linkage.

**Table 6**
Information about the missions in Scenario 1 and Scenario 2.

| Scenario | Mission ID | Mission Start Point | Mission End Point | TUAV.id | Mission Distance | Mission time limit |
|---|---|---|---|---|---|---|
| Battlefield in the wilderness | 1 | [102.60,51.30,100] | [307.80,461.70,100] | 1 5 9 13 17 ... 105 109 113 117 | 458.841 | 100 |
| | 2 | [205.20,51.30,100] | [410.40,461.70,100] | 2 6 10 14 18 ... 106 110 114 118 | 458.841 | 100 |
| | 3 | [307.80,51.30,100] | [205.20,461.70,100] | 3 7 11 15 19 ... 107 111 115 119 | 423.030 | 100 |
| | 4 | [410.40,51.30,100] | [102.60,461.70,100] | 4 8 12 16 20 ... 108 112 116 120 | 513.000 | 100 |
| Urban transportation | 1 | [50,950,40] | [560,130,40] | 1 5 9 13 17 ... 105 109 113 117 | 965.660 | 200 |
| | 2 | [950,850,40] | [350,150,40] | 2 6 10 14 18 ... 106 110 114 118 | 921.954 | 200 |
| | 3 | [750,100,40] | [200,950,40] | 3 7 11 15 19 ... 107 111 115 119 | 1012.422 | 200 |
| | 4 | [950,50,40] | [450,950,40] | 4 8 12 16 20 ... 108 112 116 120 | 1029.563 | 200 |

**Table 7**
The default values of the other parameters in Section 4.

| Section | Name | Value | Section | Name | Value |
|---|---|---|---|---|---|
| 4.3 | $Pn_a$ | 20m | 4.5 | $Pn_a$ | 0 |
| 4.4 | $Pn_a$ | 13 | 4.5 | $Pn_a$ | 256 or 120 |
| 4.4 | $Pn_a$ | 1 | 4.6 | $Pn_a$ | 3 |
| 4.5 | $Pn_a$ | 1 or 0 | 4.7 | $Pn_a$ | 0.5 |
| 4.5 | $Pn_a$ | 513 or 1000 | 4.7 | $Pn_a$ | 10 |
| 4.5 | $Pn_a$ | 1 or 0 | 4.7 | $Pn_a$ | [0.02,0.02,0.02,0.02,0.02,0.002,0.002,0.002,0.002,0.002] |
| 4.5 | $Pn_a$ | 513 or 1000 | 4 | $Pn_a$ | 5000 |

**Running Time:** Since the number of waypoints passed by UAVs is proportional to the running time, the number of nodes in this paper reflects the running time. The total time (the number of waypoints) consumed by 120 UAVs to complete 4 missions is denoted as $Pn_a$, and the total time (the number of waypoints) consumed by 30 UAVs to complete missions 1-4 is denoted as $Pn_1$, $Pn_2$, $Pn_3$, and $Pn_4$, respectively.

$$Pn_i = \sum_{j=1}^{30} N_{i+4(j-1)}, \tag{65}$$

where $N_m$ denotes the time consumed by the $m$th UAV to accomplish its corresponding mission (the number of waypoints).

**Path length:** The shorter the length of the path that the UAV has been flying over, the less fuel is required, so the length of the UAV has been flying over is also considered one of the measures of algorithm and model suitability in this paper. The total path length consumed by 120 UAVs to complete 4 missions is denoted as $L_a$, and the total path length consumed by 30 UAVs to complete missions 1-4 is denoted as $L_1$, $L_2$, $L_3$, and $L_4$, respectively.

$$L_i = \sum_{j=1}^{30} \sum_{k=1}^{N_{i+4(j-1)}} l_{i+4(j-1),k}, \tag{66}$$

where $N_m$ denotes the number of waypoints that the $m$th UAV paths through to accomplish its corresponding mission, and $l_{m,n}$ denotes the distance traveled by the $m$th UAV to move from the $n-1$th waypoint to the $n$th waypoint.

**Average path deviation error:** The average path deviation error indicates the deviation between the average path and the actual path, reflecting the difference between the ideal path and the actual path, which can also be understood as the predictability of the generated path. This value is not the smaller, the better; for example, in the case of field combat, maintaining a certain degree of

randomness is also conducive to the protection of the UAV's safety; in addition, different route selection options can also cause a great average path deviation error, so this value is only given for reference. The average path deviation by 30 UAVs to complete missions 1-4 is denoted as $S_1$, $S_2$, $S_3$, and $S_4$, respectively.

#### 6.2.3. Experimental results and analysis

Table 8 presents performance statistics for SiMlDCS, AHA, AROA, DCS, EEFO, EO, GKSO, and MSA in Scenario 1. The four values represent four tasks in order. SiMlDCS completes all missions with a minimum of 5979 waypoints, reducing the number of waypoints by 114 compared to GKSO, which helps shorten execution time. In Mission 3, SiMlDCS excels in time and distance metrics and achieves the shortest distance in Mission 4. MSA slightly outperforms SiMlDCS in Tasks 1 and 2, recording the shortest mission time overall. Although AHA and AROA achieve the shortest execution time in Mission 2, their performance worsens in other missions, suggesting less stability and a tendency to reach local optima. All algorithms maintained safe operating conditions, avoiding dangerous behaviors in Scene 1.

Fig. 22 presents a schematic of SiMlDCS path planning in Scenario 1, highlighting a static obstacle area (red column), movement trails of interfering UAVs (red ball), and paths of UAVs performing Missions 1, 2, 3, and 4 (colored red, green, yellow, and purple). The four test UAVs operate independently without interference. Fig. 23 offers a top view of Fig. 22, omitting the interfering UAVs for clarity. Despite many dynamic UAVs present, the SiMlDCS-based local path planner effectively finds a smooth path to the mission completion point in real-time, supported by a robust pre-avoidance system that prevents the planner from becoming stuck.

Fig. 24 shows the top view of the paths for 120 UAVs planned using the SiMlDCS algorithm in Scenario 1. The results of other algorithms are not detailed in the text, but images of them are included in the supplementary material.

Although it is impossible to clearly show the differences between the various algorithms from the 3D view and the top view, the algorithm performance can be reflected from other perspectives; Fig. 25 gives a cut-away plot of the altitude of SiMlDCS, AHA, AROA, DCS, EEFO, EO, GKSO, and MSA during the execution of the 30 repetitions of the mission, where $x = 0$ indicates that the node is located at the starting point.

Fig. 25 shows that the curves of SiMlDCS completing each mission are denser, indicating that the stability of SiMlDCS is stronger. Overall, SiMlDCS and GKSO are more stable, and the stability of EO is also at a leading level. Although the MSA algorithm has a more prominent performance in the statistical results of the metrics, the performance of its altitude in a large number of repetitive experiments varies greatly, which indicates that the path generated by this algorithm is not predictable, and this algorithm is difficult to find out the optimal path in a short period. The AHA and EEFO altitudes perform similarly to MSA, with large variability between curves and dispersed altitude levels, indicating that these algorithms are less stable.

In the height curve of GKSO, we found that its height curve was obviously divided into two parts when it executed Mission 3. Therefore, we carefully examined the top view of the execution result of the GKSO algorithm, and the top view of the UAV route of GKSO in Scenario 1 is shown in Fig. 26, in which the different colors corresponding to the mission information remain the same as above. Through comparison, We found that the GKSO algorithm chooses many ways to bypass the obstacles, and all of them can maintain a more stable flight state after bypassing the obstacles, so the GKSO is more stable in dealing with LPSA, which shows great potential for optimization.

In Section 4.7, we added a pre-avoidance system to the Local Path Evaluator, which helps the UAV avoid obstacles, terrain terrain, and boundary constraints in advance. In Figs. 24 and 26, we can find that most of the routes are steered away from the static obstacle regions when the routes are about to approach them, so the ability of the system to avoid static obstacle regions in advance can be verified on the images. Since most of the routes are far away from the boundary region, the ability to avoid the boundary in advance by the local path planner needs to be verified in this part. There also exists a part of the pre-avoidance system that avoids terrain obstacles in advance; in order to analyze the effect of this part of the pre-avoidance system, we plotted some of the vehicle altitude versus terrain altitude for the SiMlDCS, AHA, AROA, DCS, EEFO, EO, GKSO, MSA algorithms in Fig. 27. The black curves in the figure are the actual terrain on the current mission path altitude, and the blue line is the terrain altitude after adding *Hsafe*.

In Fig. 27, most of the UAVs rapidly increase their altitude upon entering the region at a distance of *Hsafe* from the ground in order to cross some terrain on their course, and then, after detecting that they have crossed a terrain obstacle, slowly decrease their altitude and move rapidly in the direction pointing toward the target region. In the vicinity of (0.74, 120) for Mission 1, all of the course altitudes show a localized rise in altitude. Since there is no one-to-one correspondence between the x-axis direction and time in Fig. 27, there is no accurate explanation for this phenomenon. However, the altitude value of 120 coincides with the altitude at which some of the interfering UAVs were flying, and a more plausible explanation is that the majority of the planners chose this node to avoid other UAVs by raising their altitude. In addition, SiMlDCS and GKSO react quickly when entering areas within *Hsafe* to stay away from areas close to the ground, while AHA, AROA, and DCS all generate paths with part of their routes close to the ground, which greatly increases the risk of UAVs clipping the ground, which reflects the difference between the stability of the various path planning algorithms and is a similar finding to that of the analysis of Fig. 25 above.

In Scenario 2, the map size increases with more varied terrain, challenging the local path planner. Table 9 presents performance statistics for SiMlDCS, AHA, AROA, DCS, EEFO, EO, GKSO, and MSA across different missions. The four values represent four tasks in order. SiMlDCS completed all missions with the fewest waypoints at 13,434, 199 fewer than the EO algorithm, leading to reduced execution times. It outperformed other algorithms in mission 1, 3, and 4 regarding time and distance. Although EEFO was fastest in mission 2, its performance declined in others, suggesting instability and a tendency to get trapped in local optima. Notably, all algorithms kept UAVs safe by avoiding dangerous areas and obstacles.

Fig. 28 presents a schematic of the SiMlDCS planning paths in Scenario 2, following the same color and drawing conventions as Fig. 22. The interfering UAV is omitted to enhance clarity. Fig. 29 shows the top view at the corresponding time node. Due to the subtle

**Table 8**
Statistical results of simulation experiment indicators for the 8 algorithms in Scenario 1.

| Algorithms | $Pn_a$ | $Pn_{1\text{-}4}$ | $L_a$ | $L_{1\text{-}4}$ | $S_{1\text{-}4}$ |
|---|---|---|---|---|---|
| SiMlDCS | **5979** | 1447 | 5.731E+03 | 1.426E+04 | 6.903E+03 |
| | | 1412 | | 1.378E+04 | 1.153E+04 |
| | | **1358** | | **1.321E+04** | 9.237E+03 |
| | | 1762 | | **1.620E+04** | 9.058E+03 |
| AHA | 6010 | 1455 | 5.726E+03 | 1.418E+04 | 7.045E+03 |
| | | **1410** | | 1.374E+04 | 1.153E+04 |
| | | 1387 | | 1.333E+04 | 9.156E+03 |
| | | 1758 | | 1.626E+04 | 9.095E+03 |
| AROA | 6030 | 1462 | 5.751E+03 | 1.434E+04 | 6.937E+03 |
| | | **1410** | | 1.379E+04 | 1.150E+04 |
| | | 1372 | | 1.328E+04 | 9.227E+03 |
| | | 1786 | | 1.627E+04 | 9.026E+03 |
| DCS | 6070 | 1458 | 5.758E+03 | 1.429E+04 | 6.955E+03 |
| | | 1412 | | 1.381E+04 | 1.153E+04 |
| | | 1434 | | 1.339E+04 | 9.183E+03 |
| | | 1766 | | 1.625E+04 | 9.071E+03 |
| EEFO | 5995 | 1450 | 5.718E+03 | 1.417E+04 | 7.062E+03 |
| | | 1411 | | **1.373E+04** | 1.152E+04 |
| | | 1383 | | 1.329E+04 | 9.155E+03 |
| | | **1751** | | 1.622E+04 | 9.104E+03 |
| EO | 6054 | 1459 | 5.755E+03 | 1.429E+04 | 6.915E+03 |
| | | 1411 | | 1.380E+04 | 1.151E+04 |
| | | 1417 | | 1.333E+04 | 9.274E+03 |
| | | 1767 | | 1.628E+04 | 9.084E+03 |
| GKSO | 6093 | 1471 | 5.770E+03 | 1.439E+04 | 6.928E+03 |
| | | 1428 | | 1.392E+04 | 1.133E+04 |
| | | 1380 | | **1.321E+04** | 8.931E+03 |
| | | 1814 | | 1.630E+04 | 8.995E+03 |
| MSA | 5985 | **1442** | **5.715E+03** | **1.413E+04** | 7.032E+03 |
| | | **1410** | | 1.376E+04 | 1.155E+04 |
| | | 1367 | | 1.324E+04 | 9.124E+03 |
| | | 1766 | | **1.620E+04** | 9.080E+03 |

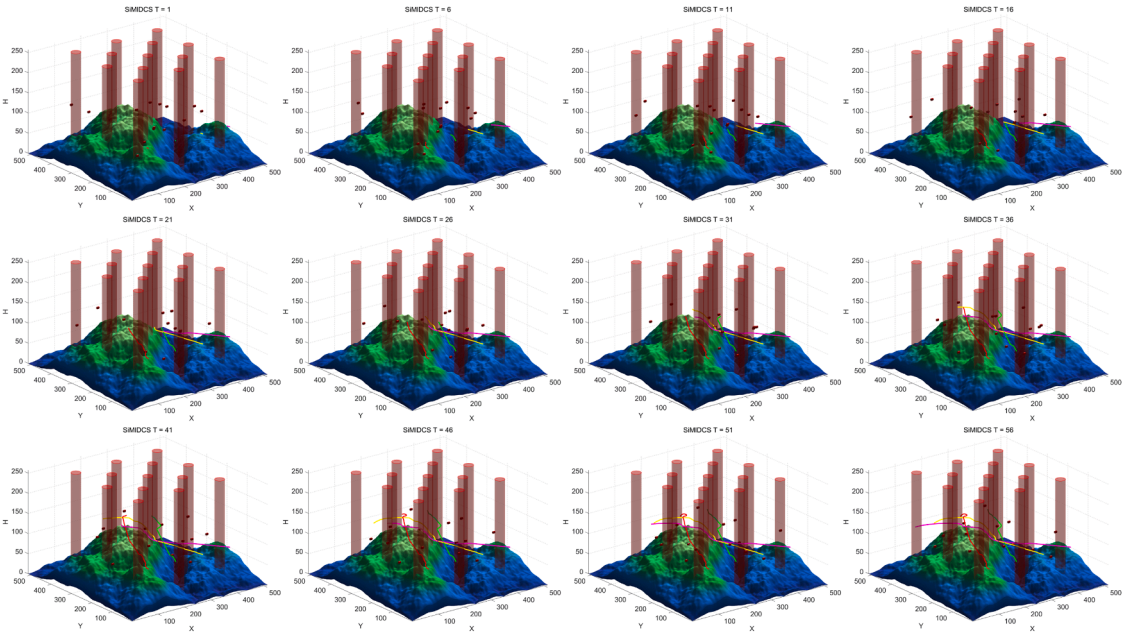


**Fig. 22.** 3D view of the UAV route for the first 4 experiments (Scenario 1).

differences between algorithms, only selected results of the SiMlDCS algorithm are included, with additional results available in the Supplementary Material.

Fig. 30 shows the altitude of SiMlDCS, AHA, AROA, DCS, EEFO, EO, GKSO, and MSA during 30 repetitions of Scenario 2. The

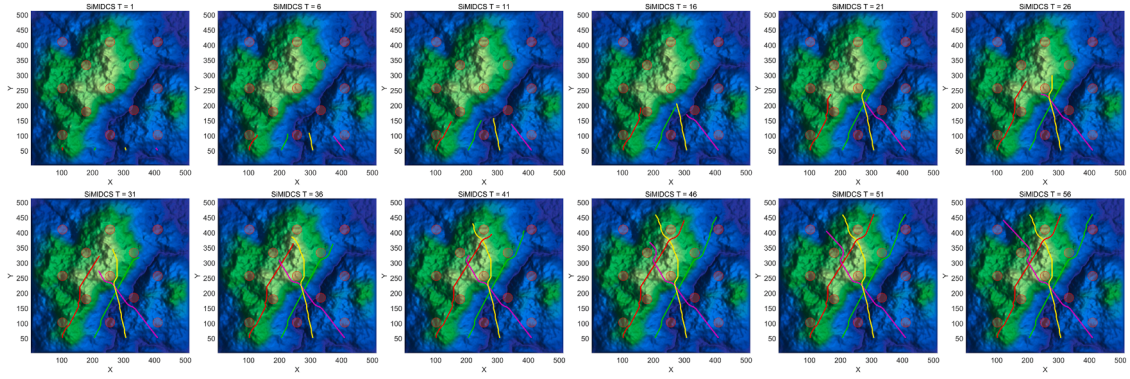


**Fig. 23.** Top view of the UAV route for the first 4 experiments (Scenario 1).

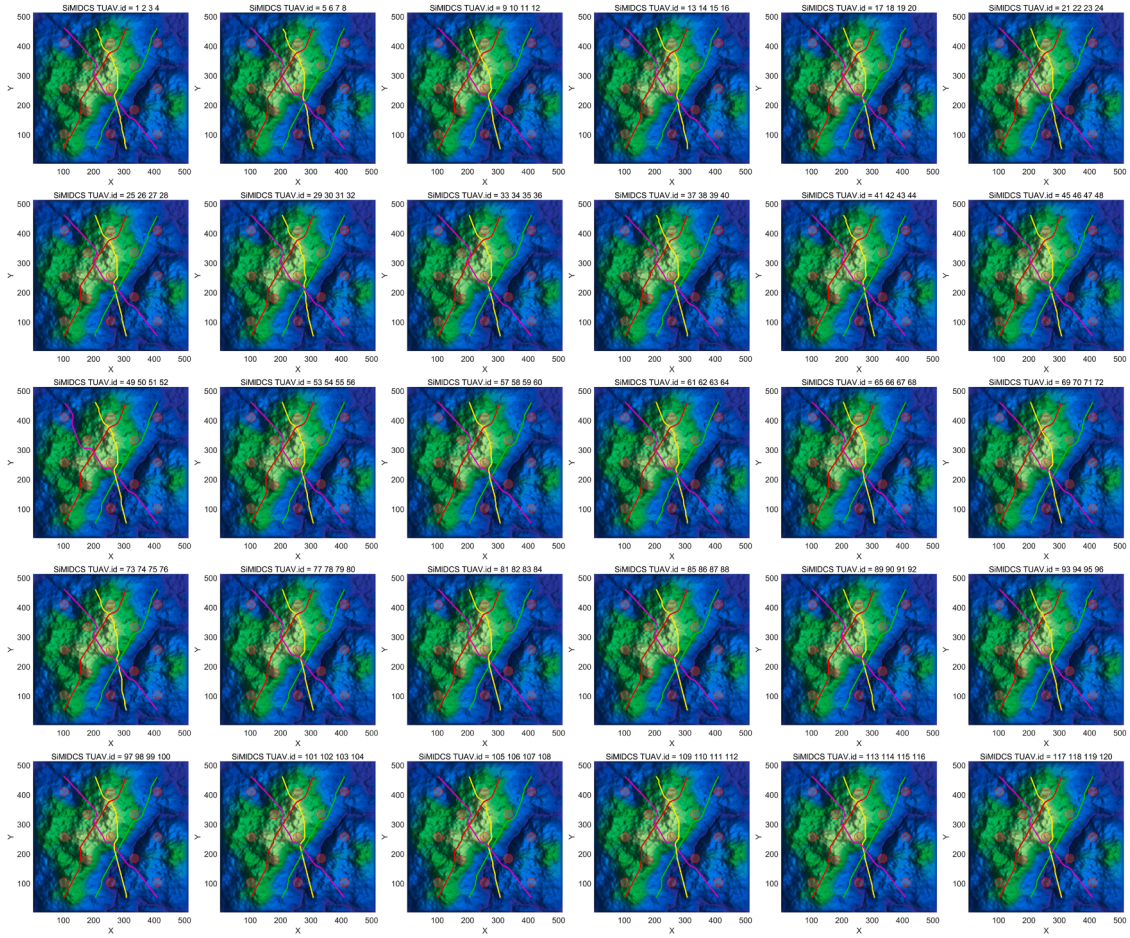


**Fig. 24.** Top view of UAV routes for all 120 experiments of the SiMlDCS algorithm (Scenario 1).

conclusions analyzed in Fig. 30 are similar to those above and will not be further explained here.

### *6.3. Dynamic pursuit experiment*

In Section 6.2, we previously validated the SiMlDCS algorithm's effectiveness for UAV swarm path planning through numerous experiments. This subsection focuses on a 4D local path planning method following the PSCM process, implementing a dynamic target pursuit experiment. The purpose of the experiment is to assess the planning process's design and to evaluate the UAVs' collision

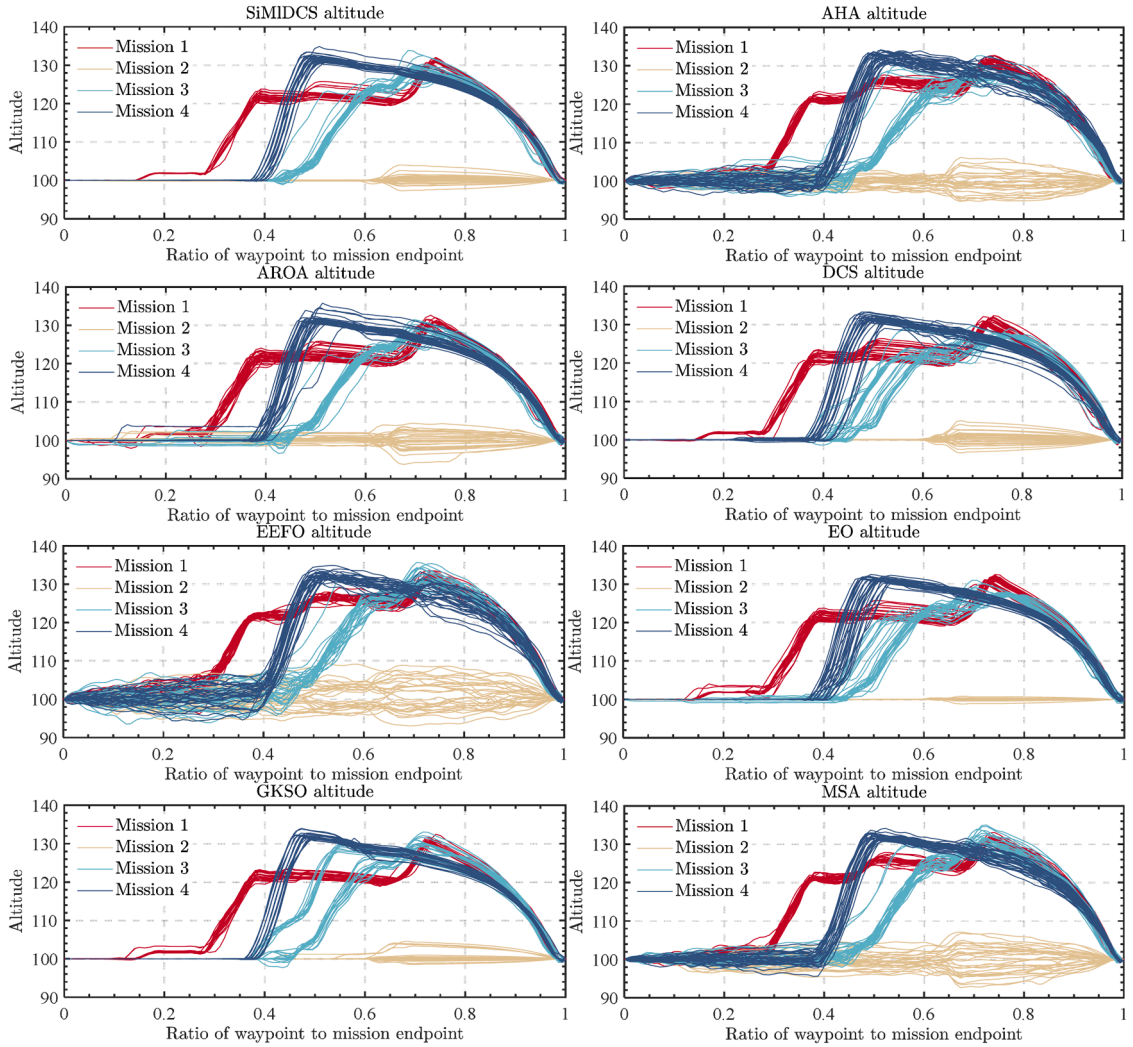


**Fig. 25.** UAV course altitude cutaway (Scenario 1).

avoidance capabilities in crowded situations.

We introduced an escaping black UAV, which moves in a circular track, while a swarm of 5 pursuit UAVs aims to catch it. The pursuit UAVs navigate using the PSCM process to assess planning effectiveness and tests collision avoidance in crowded conditions. The escape UAVs escape according to a fixed circular track in order to gradually converge the pursuit UAVs on the gathering place positions. All UAVs have a collision volume radius of 15. This experiment focuses on PSCM design rationality and collision avoidance, showcasing top views of results based on the SiMlDCS algorithm, as seen in Fig. 31.

During the experiment, none of the UAVs in the swarm collided. The top view showed that the UAVs mainly directed towards the target while maintaining a safe distance from each other, even when close. At $T = 47$, the green UAVs adjusted their flight to avoid collisions, demonstrating the effectiveness of the swarm planning and their collision avoidance capabilities in crowded scenarios. The UAV swarm's dynamic target pursuit experiment showcases the benefits of the local path planner. The 4D collision-free local path planning method retains these advantages while simplifying complex issues into multiple low-dimensional problems. This approach enhances UAV autonomous control and highlights the practical potential of the planning method.

## 7. Conclusion

A novel 4D collision-free dynamic path planning method is constructed in this paper, in which we use a hybrid modeling method to re-model the solution area, which not only solves the problem of "combinatorial explosion" when the traditional 2D grid model is extended to 3D space but also retains the detailed information of the elements in the environment as much as possible, which reduces the complexity of the environment. Moreover, this paper plans paths with the help of temporary static obstacle regions using the PSCM approach, which transforms multiple UAV global path planning missions into a single UAV local path planning mission and greatly

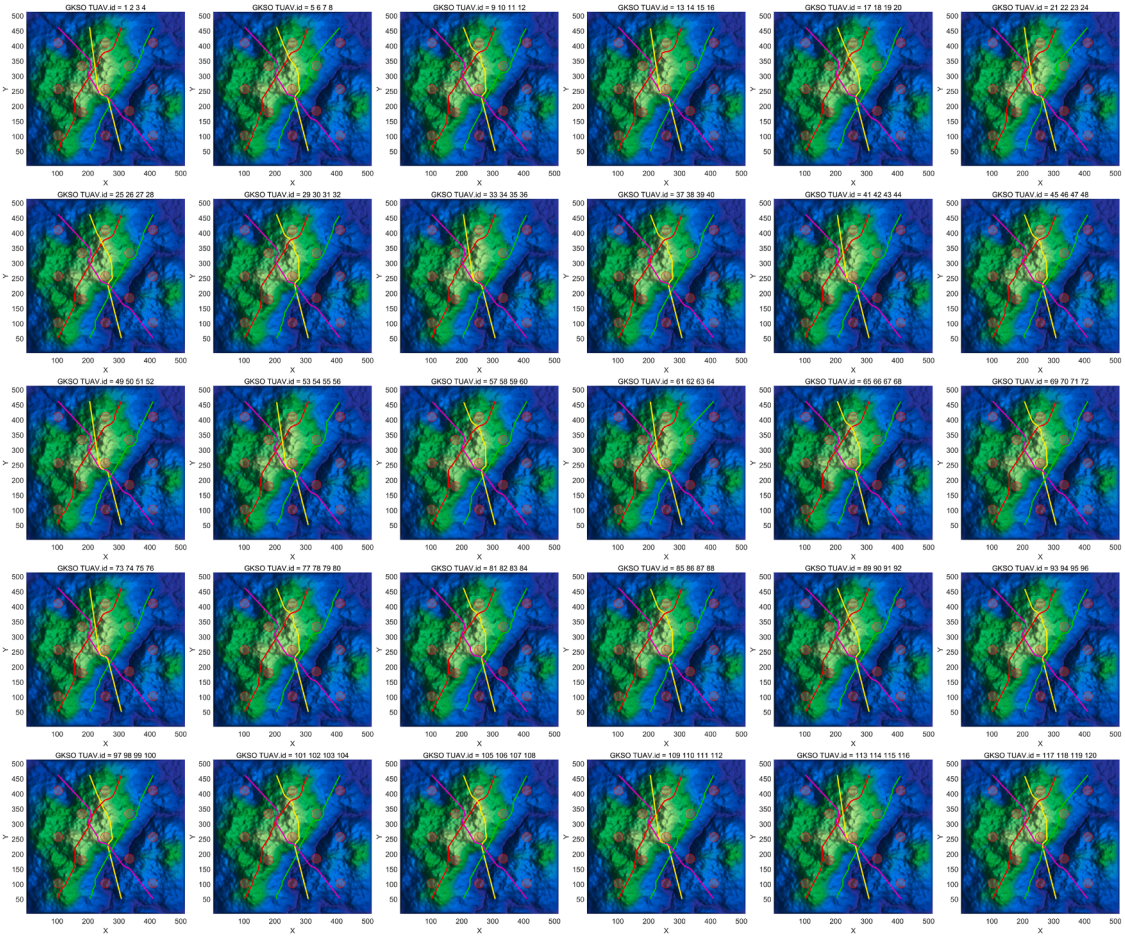


**Fig. 26.** Top view of UAV routes for all 120 experiments of the GKSO algorithm (Scenario 1).

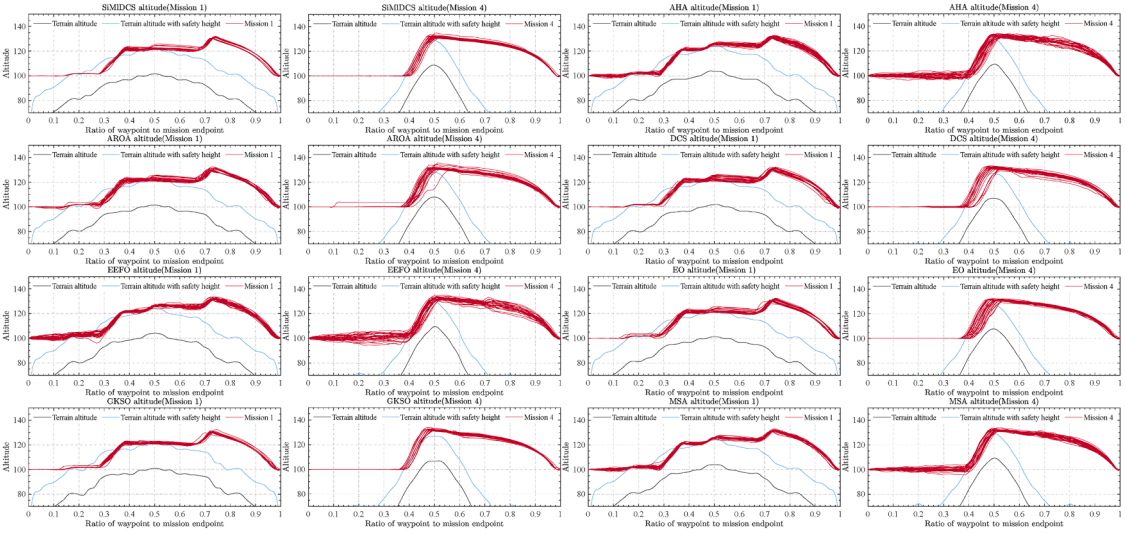


**Fig. 27.** Partial UAV route height cutaway (Scenario 1).

**Table 9**
Statistical results of simulation experiment indicators for the 8 algorithms in Scenario 2.

| Algorithms | $Pn_a$ | $Pn_{1\text{-}4}$ | $L_a$ | $L_{1\text{-}4}$ | $S_{1\text{-}4}$ |
|---|---|---|---|---|---|
| SiMlDCS | **13434** | **3484** | **1.237E+04** | 3.040E+04 | 1.431E+04 |
| | | 3558 | | 3.011E+04 | 2.573E+04 |
| | | **3242** | | **3.211E+04** | 1.854E+04 |
| | | **3150** | | **3.120E+04** | 2.872E+04 |
| AHA | 13536 | 3495 | 1.239E+04 | 3.039E+04 | 1.433E+04 |
| | | 3523 | | 3.010E+04 | 2.591E+04 |
| | | 3337 | | 3.230E+04 | 1.846E+04 |
| | | 3181 | | 3.129E+04 | 2.870E+04 |
| AROA | 13593 | 3551 | 1.238E+04 | 3.037E+04 | 1.431E+04 |
| | | 3583 | | 3.006E+04 | 2.578E+04 |
| | | 3291 | | 3.224E+04 | 1.854E+04 |
| | | 3168 | | 3.137E+04 | 2.876E+04 |
| DCS | 13550 | 3536 | **1.237E+04** | **3.032E+04** | 1.436E+04 |
| | | 3579 | | 3.006E+04 | 2.573E+04 |
| | | 3272 | | 3.220E+04 | 1.854E+04 |
| | | 3163 | | 3.132E+04 | 2.876E+04 |
| EEFO | 13564 | 3509 | 1.242E+04 | 3.040E+04 | 1.429E+04 |
| | | **3507** | | 3.017E+04 | 2.595E+04 |
| | | 3360 | | 3.244E+04 | 1.849E+04 |
| | | 3188 | | 3.141E+04 | 2.874E+04 |
| EO | 13633 | 3514 | 1.241E+04 | 3.034E+04 | 1.432E+04 |
| | | 3581 | | **2.999E+04** | 2.580E+04 |
| | | 3336 | | 3.246E+04 | 1.858E+04 |
| | | 3202 | | 3.156E+04 | 2.881E+04 |
| GKSO | 13560 | 3580 | 1.239E+04 | 3.055E+04 | 1.445E+04 |
| | | 3558 | | 3.015E+04 | 2.573E+04 |
| | | 3270 | | 3.215E+04 | 1.848E+04 |
| | | 3152 | | 3.127E+04 | 2.874E+04 |
| MSA | 13480 | 3499 | **1.237E+04** | 3.036E+04 | 1.433E+04 |
| | | 3530 | | 3.015E+04 | 2.583E+04 |
| | | 3296 | | 3.220E+04 | 1.852E+04 |
| | | 3155 | | 3.122E+04 | 2.875E+04 |

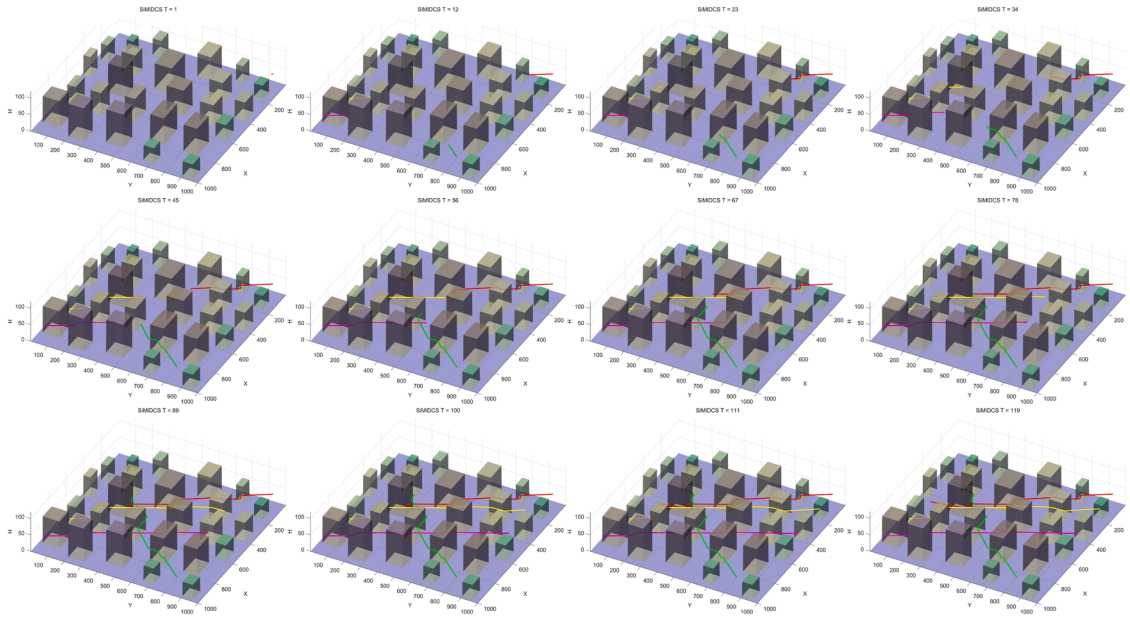

**Fig. 28.** 3D view of the UAV route for the first 4 experiments (Scenario 2).

reduces the complexity of the planning mission.

The local path generator for UAVs must provide accurate optimal paths quickly while balancing optimization time and memory consumption. To meet these requirements, this paper proposes SiMlDCS, an enhanced DCS algorithm that achieves high optimization accuracy in a short time, outperforming many other algorithms across various test functions. We simulated two scenarios, using this

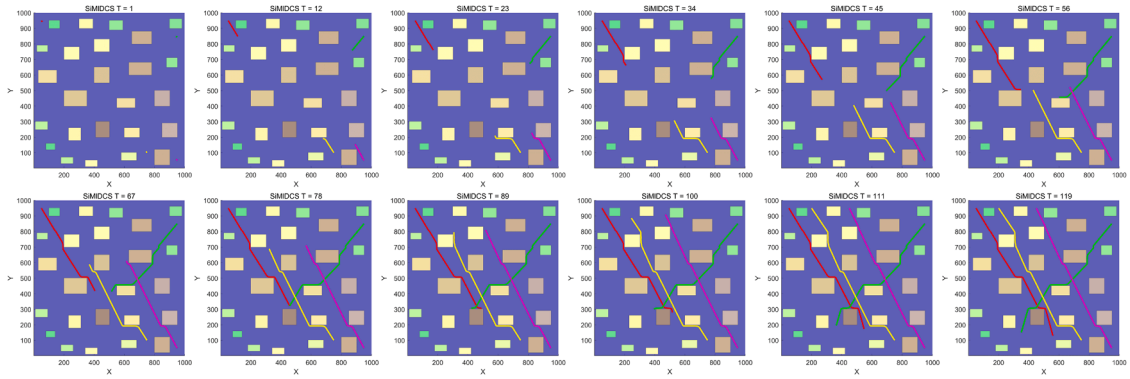


**Fig. 29.** Top view of the UAV route for the first 4 experiments (Scenario 2).

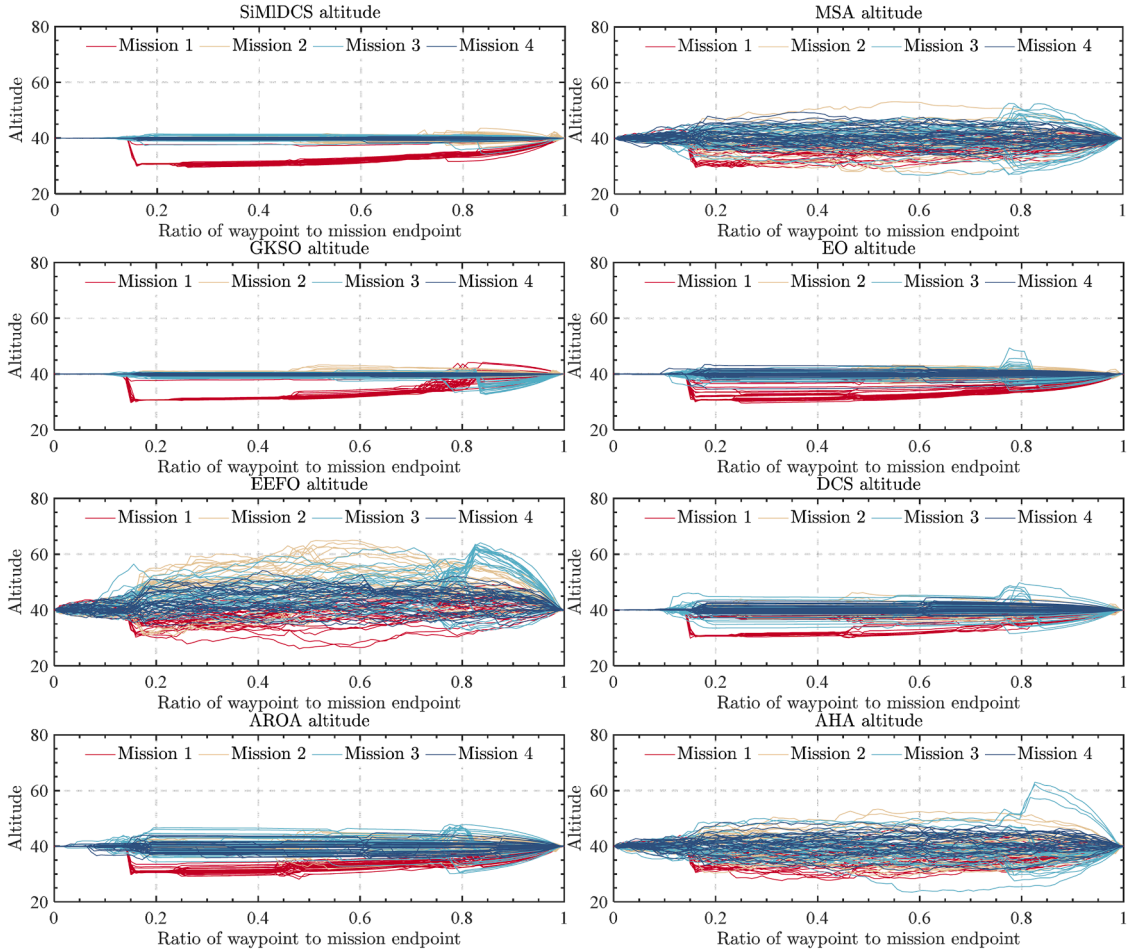


**Fig. 30.** UAV course altitude cutaway (Scenario 2).

method and found that SiMlDCS notably enhances mission execution time and flying distance while maintaining stability. The experiments, compared with seven robust algorithms, confirmed the effectiveness of SiMlDCS. Additionally, we conducted UAV swarm dynamic target-chasing experiments, carefully designing the speeds and paths for pursuing and escaping UAVs. The results demonstrated the reasonableness of the PSCM method and its effectiveness in crowded scenarios. Overall, the experiments validate the proposed environment modeling, planning process, and path generation algorithm.

Although the new PSCM-based 4D collision-free dynamic path planning method proposed in this paper can greatly reduce the complexity of the optimization problem of the planning method, the method also has certain shortcomings: (1) As a new local path

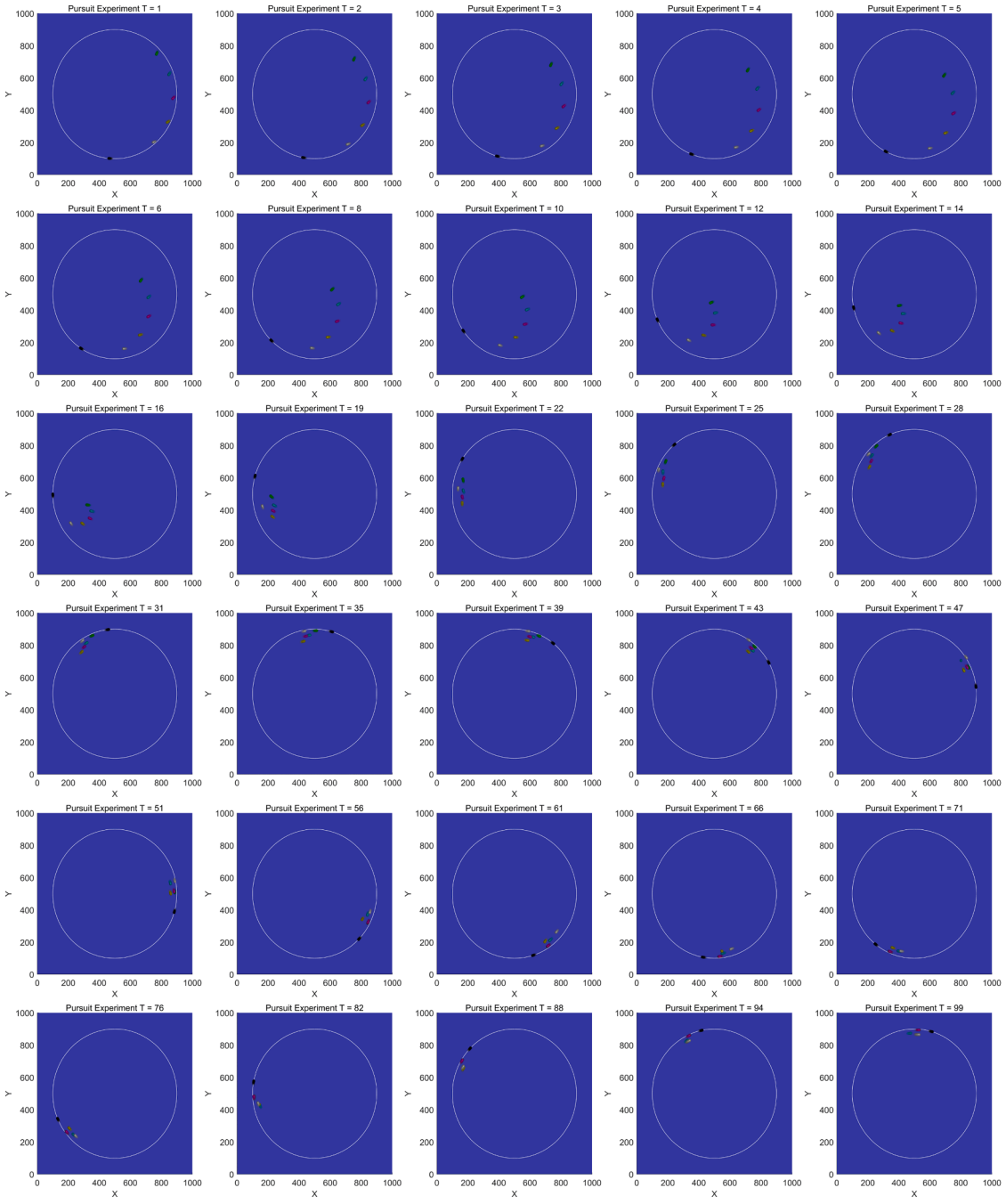


**Fig. 31.** Top view of UAV dynamic target pursuit experiment based on PSCM process.

planning method, the method has the inherent shortcomings of a local path planner, that is poorly perceived for the overall environment, and is easy to fall into the local position. (2) As the cost of model simplification, the smoothness of UAV at time nodes is not considered in the local path evaluation system. In the future, we will enhance our approach by combining GPSA and LPSA for complete path coverage. We will focus on path smoothness in local evaluations, define local paths as smooth curves, and create a path prediction system for UAVs without static obstacle regions. Additionally, we will incorporate real physical experiments into our future research to further verify the effectiveness of the methods we have proposed.

## CRediT authorship contribution statement

**Gang Hu:** Writing – review & editing, Writing – original draft, Supervision, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Peidong He:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Mahmoud Abdel Salam:** Writing – review & editing, Writing – original draft, Supervision, Resources, Methodology, Investigation, Funding acquisition, Data curation, Conceptualization. **Guo Wei:** Writing – review & editing, Writing – original draft, Supervision, Resources, Methodology, Investigation, Formal analysis, Conceptualization.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant No. 52375264).

## Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.apm.2025.116383.

## Data availability

Data will be made available on request.

## References

[1] S. Aggarwal, N. Kumar, Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges, Comput. Commun. 149 (2020) 270–299, https://doi.org/10.1016/j.comcom.2019.10.014.
[2] P. Mandal, L.P. Roy, S.K. Das, Tracking of invader drone using hybrid unscented Kalman-continuous ant colony filter (HUK-CACF), ISA Trans. (2024), https://doi.org/10.1016/j.isatra.2024.06.018.
[3] A.N. Skraparlis, K.S. Ntalianis, N. Tsapatsoulis, A novel framework to intercept GPS-denied, bomb-carrying, non-military, kamikaze drones: towards protecting critical infrastructures, Defence Technol. (2024), https://doi.org/10.1016/j.dt.2024.05.001.
[4] D.-Z. Gao, et al., A new approach to surveying cliff-dwelling endangered plants using drone-based nap-of-the-object photography: A case study of Clematis acerifolia, Glob. Ecol. Conserv. 49 (2024) e02769, https://doi.org/10.1016/j.gecco.2023.e02769.
[5] T. Ezaki, et al., Drone-based vertical delivery system for high-rise buildings: multiple drones vs. a single elevator, Commun. Transp. Res. 4 (2024) 100130, https://doi.org/10.1016/j.commtr.2024.100130.
[6] S. Choi, et al., Mathematical programming-based heuristic for highway patrol drone scheduling problem, Socioecon. Plann. Sci. 93 (2024) 101907, https://doi.org/10.1016/j.seps.2024.101907.
[7] M. Ishiwatari, Leveraging drones for effective disaster management: A comprehensive analysis of the 2024 Noto Peninsula earthquake case in Japan, Prog. Disaster. Sci. 23 (2024) 100348, https://doi.org/10.1016/j.pdisas.2024.100348.
[8] J. Inoue, K. Sato, Challenges in detecting clouds in polar regions using a drone with onboard low-cost particle counter, Atmos. Environ. 314 (2023) 120085, https://doi.org/10.1016/j.atmosenv.2023.120085.
[9] A. Mata, et al., Drone imagery and deep learning for mapping the density of wild Pacific oysters to manage their expansion into protected areas, Ecol. Inform. (2024) 102708, https://doi.org/10.1016/j.ecoinf.2024.102708.
[10] Z. Jin, et al., A risk-averse distributionally robust optimisation approach for drone-supported relief facility location problem, Transp. Res. Part E: Logist. Transp. Rev. 186 (2024) 103538, https://doi.org/10.1016/j.tre.2024.103538.
[11] T. Leith, et al., Bystander interaction with a novel multipurpose medical drone: A simulation trial, Resusc. Plus. 18 (2024) 100633, https://doi.org/10.1016/j.resplu.2024.100633.
[12] S. Meesaragandla, et al., Herbicide spraying and weed identification using drone technology in modern farms: A comprehensive review, Results. Eng. 21 (2024) 101870, https://doi.org/10.1016/j.rineng.2024.101870.
[13] C. Huang, et al., Adaptive cylinder vector particle swarm optimization with differential evolution for UAV path planning, Eng. Appl. Artif. Intell. 121 (2023) 105942, https://doi.org/10.1016/j.engappai.2023.105942.
[14] L. Han, et al., An enhanced adaptive 3D path planning algorithm for mobile robots with obstacle buffering and improved theta* using minimum snap trajectory smoothing, J. King Saud Univ. - Comput. Inf. Sci. 35 (2023) 101844, https://doi.org/10.1016/j.jksuci.2023.101844.
[15] A. Puente-Castro, et al., Q-learning based system for path planning with Unmanned aerial Vehicles swarms in obstacle environments, Expert. Syst. Appl. 235 (2024) 121240, https://doi.org/10.1016/j.eswa.2023.121240.
[16] H. Tu, et al., Improved RRT global path planning algorithm based on Bridge Test, Rob. Auton. Syst. 171 (2024) 104570, https://doi.org/10.1016/j.robot.2023.104570.
[17] R. Akay, M.Y. Yildirim, Multi-strategy and self-adaptive differential sine–cosine algorithm for multi-robot path planning, Expert. Syst. Appl. 232 (2023) 120849, https://doi.org/10.1016/j.eswa.2023.120849.
[18] Y. Cui, et al., Multi-robot path planning using learning-based artificial Bee Colony algorithm, Eng. Appl. Artif. Intell. 129 (2024) 107579, https://doi.org/10.1016/j.engappai.2023.107579.
[19] G. Li, et al., A mixing algorithm of ACO and ABC for solving path planning of mobile robot, Appl. Soft. Comput. 148 (2023) 110868, https://doi.org/10.1016/j.asoc.2023.110868.
[20] X. Xu, et al., Research on global path planning algorithm for mobile robots based on improved A*, Expert. Syst. Appl. 243 (2024) 122922, https://doi.org/10.1016/j.eswa.2023.122922.

[21] L. Wu, et al., Modified adaptive ant colony optimization algorithm and its application for solving path planning of mobile robot, Expert. Syst. Appl. 215 (2023) 119410, https://doi.org/10.1016/j.eswa.2022.119410.
[22] G. Hu, et al., SaCHBA_PDN: Modified honey badger algorithm with multi-strategy for UAV path planning, Expert. Syst. Appl. 223 (2023) 119941, https://doi.org/10.1016/j.eswa.2023.119941.
[23] W.E. Howden, The sofa problem, Comput. J. 11 (1968) 299–301, https://doi.org/10.1093/comjnl/11.3.299.
[24] L. Liu, et al., Path planning techniques for mobile robots: review and prospect, Expert. Syst. Appl. 227 (2023) 120254, https://doi.org/10.1016/j.eswa.2023.120254.
[25] J. Han, An efficient approach to 3D path planning, Inform. Sci. 478 (2019) 318–330, https://doi.org/10.1016/j.ins.2018.11.045.
[26] T. Lei, et al., Graph-based robot optimal path planning with bio-inspired algorithms, Biomimetic Intell. Robot. 3 (2023) 100119, https://doi.org/10.1016/j.birob.2023.100119.
[27] H. Ali, et al., Feature selection-based decision model for UAV path planning on rough terrains, Expert. Syst. Appl. 232 (2023) 120713, https://doi.org/10.1016/j.eswa.2023.120713.
[28] Z. Ma, J. Chen, Adaptive path planning method for UAVs in complex environments, Int. J. Appl. Earth. Obs. Geoinf. 115 (2022) 103133, https://doi.org/10.1016/j.jag.2022.103133.
[29] B. Sahu, et al., Multi-robot cooperation and path planning for stick transporting using improved Q-learning and democratic robotics PSO, J. Comput. Sci. 60 (2022) 101637, https://doi.org/10.1016/j.jocs.2022.101637.
[30] C. Saranya, et al., Terrain based D* algorithm for path planning, IFAC-PapersOnLine 49 (2016) 178–182, https://doi.org/10.1016/j.ifacol.2016.03.049.
[31] S.M. Langbak, et al., Multi-autonomous mobile robot traffic management based on layered costmaps and a modified Dijkstra's algorithm, Procedia Comput. Sci. 232 (2024) 53–63, https://doi.org/10.1016/j.procs.2024.01.006.
[32] Y. Ning, et al., Three-dimensional path planning for a novel sediment sampler in ocean environment based on an improved mutation operator genetic algorithm, Ocean Eng. 289 (2023) 116142, https://doi.org/10.1016/j.oceaneng.2023.116142.
[33] Z. Cao, et al., An adaptive population size based differential evolution by mining historical population similarity for path planning of unmanned aerial vehicles, Inf. Sci. (Ny) 666 (2024) 120432, https://doi.org/10.1016/j.ins.2024.120432.
[34] Z. Cai, et al., Cooperative path planning study of distributed multi-mobile robots based on optimised ACO algorithm, Rob. Auton. Syst. 179 (2024) 104748, https://doi.org/10.1016/j.robot.2024.104748.
[35] J. Cui, et al., Multi-strategy adaptable ant colony optimization algorithm and its application in robot path planning, Knowl. Based. Syst. 288 (2024) 111459, https://doi.org/10.1016/j.knosys.2024.111459.
[36] Y. Li, et al., Unified path planning for composite UAVs via Fermat point-based grouping particle swarm optimization, Aerosp. Sci. Technol. 148 (2024) 109088, https://doi.org/10.1016/j.ast.2024.109088.
[37] B. Tao, J.-H. Kim, Mobile robot path planning based on bi-population particle swarm optimization with random perturbation strategy, J. King Saud Univ. - Comput. Inform. Sci. 36 (2024) 101974, https://doi.org/10.1016/j.jksuci.2024.101974.
[38] J.H. Holland, Genetic algorithms, Sci. Am. 267 (1992) 66–73. http://www.jstor.org/stable/24939139.
[39] R. Storn, Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces, Diff. Evol. 11 (1997) 341–359.
[40] M. Dorigo, et al., Ant colony optimization, IEEe Comput. Intell. Mag. 1 (2006) 28–39, https://doi.org/10.1109/MCI.2006.329691.
[41] J. Kennedy', R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, 1995, pp. 1942–1948.
[42] G. Hu, et al., DRPSO:A multi-strategy fusion particle swarm optimization algorithm with a replacement mechanisms for colon cancer pathology image segmentation, Comput. Biol. Med. 178 (2024) 108780, https://doi.org/10.1016/j.compbiomed.2024.108780.
[43] G. Hu, et al., MNEARO: A meta swarm intelligence optimization algorithm for engineering applications, Comput. Methods Appl. Mech. Eng. 419 (2024) 116664, https://doi.org/10.1016/j.cma.2023.116664.
[44] Y. Huang, et al., A novel path planning approach for AUV based on improved whale optimization algorithm using segment learning and adaptive operator selection, Ocean Eng. 280 (2023) 114591, https://doi.org/10.1016/j.oceaneng.2023.114591.
[45] Y. Wang, et al., Application of multi-objective artificial gorilla troops optimization algorithm in intelligent evacuation path planning for building fires, J. Build. Eng. 93 (2024) 109757, https://doi.org/10.1016/j.jobe.2024.109757.
[46] J. Wu, Z. Su, Flavoring search algorithm with applications to engineering optimization problems and robot path planning, Appl. Math. Model. 135 (2024) 396–437, https://doi.org/10.1016/j.apm.2024.07.002.
[47] C. Cheng, et al., Path planning and obstacle avoidance for AUV: A review, Ocean Eng. 235 (2021) 109355, https://doi.org/10.1016/j.oceaneng.2021.109355.
[48] X. Long, et al., Improved particle swarm optimization with reverse learning and neighbor adjustment for space surveillance network task scheduling, Swarm. Evol. Comput. 85 (2024) 101482, https://doi.org/10.1016/j.swevo.2024.101482.
[49] Z. Huang, et al., An adaptive bidirectional quick optimal rapidly-exploring random tree algorithm for path planning, Eng. Appl. Artif. Intell. 135 (2024) 108776, https://doi.org/10.1016/j.engappai.2024.108776.
[50] X. Zheng et al. Path planning of prm based on artificial potential field in radiation environments, (2023). https://doi.org/10.2139/ssrn.4501235.
[51] M. Yao, et al., Improved dynamic windows approach based on energy consumption management and fuzzy logic control for local path planning of mobile robots, Comput. Ind. Eng. 187 (2024) 109767, https://doi.org/10.1016/j.cie.2023.109767.
[52] X. Meng, et al., Dynamic path planning of autonomous bulldozers using activity-value-optimised bio-inspired neural networks and adaptive cell decomposition, Appl. Soft. Comput. 164 (2024) 111944, https://doi.org/10.1016/j.asoc.2024.111944.
[53] S. Du, et al., Real-time local path planning strategy based on deep distributional reinforcement learning, Neurocomputing. 599 (2024) 128085, https://doi.org/10.1016/j.neucom.2024.128085.
[54] X. Li, et al., Dynamic path planning of mobile robots using adaptive dynamic programming, Expert. Syst. Appl. 235 (2024) 121112, https://doi.org/10.1016/j.eswa.2023.121112.
[55] M.H. Nadimi-Shahraki, et al., An improved grey wolf optimizer for solving engineering problems, Expert. Syst. Appl. 166 (2021) 113917, https://doi.org/10.1016/j.eswa.2020.113917.
[56] K. Rajwar, et al., An exhaustive review of the metaheuristic algorithms for search and optimization: taxonomy, applications, and open challenges, Artif. Intell. Rev. 56 (2023) 13187–13257, https://doi.org/10.1007/s10462-023-10470-y.
[57] K. Hussain, et al., Metaheuristic research: a comprehensive survey, Artif. Intell. Rev. 52 (2019) 2191–2233, https://doi.org/10.1007/s10462-017-9605-z.
[58] G. Hu, et al., CGKOA: an enhanced Kepler optimization algorithm for multi-domain optimization problems, Comput. Methods Appl. Mech. Eng. 425 (2024) 116964, https://doi.org/10.1016/j.cma.2024.116964.
[59] P. Duankhan, et al., The Differentiated Creative Search (DCS): leveraging differentiated knowledge-acquisition and creative realism to address complex optimization problems, Expert. Syst. Appl. 252 (2024) 123734, https://doi.org/10.1016/j.eswa.2024.123734.
[60] G. Hu, et al., Super eagle optimization algorithm based three-dimensional ball security corridor planning method for fixed-wing UAVs, Adv. Eng. Inform. 59 (2024) 102354, https://doi.org/10.1016/j.aei.2024.102354.
[61] L. Kocis, W.J. Whiten, Computational investigations of low-discrepancy sequences, ACM Trans. Math. Softw. 23 (1997) 266–294, https://doi.org/10.1145/264029.264064.
[62] M. Abdel-Basset, et al., Nutcracker optimizer: A novel nature-inspired metaheuristic algorithm for global optimization and engineering design problems, Knowl. Based. Syst. 262 (2023) 110248, https://doi.org/10.1016/j.knosys.2022.110248.
[63] K. Cymerys, M. Oszust, Attraction–Repulsion optimization algorithm for global optimization problems, Swarm. Evol. Comput. 84 (2024) 101459, https://doi.org/10.1016/j.swevo.2023.101459.
[64] M. Abdel-Basset, et al., Mantis Search algorithm: A novel bio-inspired algorithm for global optimization and engineering design problems, Comput. Methods Appl. Mech. Eng. 415 (2023) 116200, https://doi.org/10.1016/j.cma.2023.116200.

[65] N.H. Awad, et al., Ensemble sinusoidal differential covariance matrix adaptation with euclidean neighborhood for solving CEC2017 benchmark problems, in: 2017 IEEE Congress on Evolutionary Computation (CEC), Donostia, San Sebastián, Spain, IEEE, 2017, pp. 372–379, https://doi.org/10.1109/CEC.2017.7969336.
[66] Q. Fan, et al., A modified equilibrium optimizer using opposition-based learning and novel update rules, Expert. Syst. Appl. 170 (2021) 114575, https://doi.org/10.1016/j.eswa.2021.114575.
[67] G. Hu, et al., Genghis Khan shark optimizer: A novel nature-inspired algorithm for engineering optimization, Adv. Eng. Inf. 58 (2023) 102210, https://doi.org/10.1016/j.aei.2023.102210.
[68] A.W. Mohamed, et al., LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems, in: 2017 IEEE Congress on Evolutionary Computation (CEC), Donostia, San Sebastián, Spain, IEEE, 2017, pp. 145–152, https://doi.org/10.1109/CEC.2017.7969307.
[69] W. Zhao, et al., Artificial hummingbird algorithm: A new bio-inspired optimizer with its engineering applications, Comput. Methods Appl. Mech. Eng. 388 (2022) 114194, https://doi.org/10.1016/j.cma.2021.114194.
[70] W. Zhao, et al., Electric eel foraging optimization: A new bio-inspired optimizer for engineering applications, Expert. Syst. Appl. 238 (2024) 122200, https://doi.org/10.1016/j.eswa.2023.122200.