Computers and Electrical Engineering 123 (2025) 110034

Contents lists available at ScienceDirect

Computers and Electrical Engineering

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

Path planning for unmanned aerial vehicles in complex environment based on an improved continuous ant colony optimisation

Ben Niu [a], Yongjin Wang [a], Jing Liu [b,*], Gabriel Xiao-Guang Yue [c]

[a] College of Management, Shenzhen University, Shenzhen 518060, China
[b] School of Systems and Computing, University of New South Wales, Canberra, Australia
[c] Department of Computer Science and Engineering, European University Cyprus, Nicosia, Cyprus

ARTICLE INFO

Keywords:
Ant colony optimisation
Unmanned aerial vehicle
Path planning
Unknown scenario

ABSTRACT

To address the complex challenge of unmanned aerial vehicle (UAV) path planning, a novel continuous ant colony optimisation with an improved state transition probability, a random-walk strategy and an adaptive waypoints-repair method ($ACOSRA_R$) is proposed to enhance the efficiency and accuracy of UAV 3D path planning. In $ACOSRA_R$, an improved state transition probability is integrated to simplify the search process, enabling the algorithm to converge rapidly. A random-walk strategy involves switching between employing Brownian motion and Lévy flight to help it escape from local optima in the later stage and increase the possibility of exploring new solutions. An adaptive waypoints-repair method is proposed to repair waypoints in the infeasible domain to enhance flight efficiency. To validate its performance, $ACOSRA_R$ is compared with seven advanced meta-heuristic algorithms on 9 real digital elevation model maps. Experimental results show that $ACOSRA_R$ outperforms other comparison algorithms, efficiently generating higher-quality UAV paths in different environments. Additionally, we successfully integrated the dynamic window approach with $ACOSRA_R$ to solve UAV path planning in a partially unknown scenario with static and moving obstacles.

1. Introduction

In recent years, Unmanned Aerial Systems (UAS) have been widely used in various civil fields, such as agricultural monitoring and logistics transportation. As a key component of UAS, unmanned aerial vehicles (UAVs) rely on a reliable path planning algorithm to assist in mission completion. Path planning is the key technology for the autonomous UAV flight and requires a high degree of intelligence and flexibility to determine the best feasible path for each UAV. Due to the autonomy and complexity of the UAV, it becomes a challenging problem for the planned paths to be collision-free and feasible in a partially unknown environment with a continuous search space.

In general, path planning problems can be classified into global path planning based on a static environment and local path planning based on a dynamic environment according to the difference of environmental information they deal with [1]. For global path planning, a lot of work has been done by researchers in recent years and a variety of algorithms have been proposed, including node based methods [2], sampling based methods [3], potential field methods [4], and evolutionary computation algorithms

* Corresponding author.
E-mail addresses: drniuben@gmail.com (B. Niu), 2210135009@email.szu.edu.cn (Y. Wang), jing.liu5@unsw.edu.au (J. Liu), X.Yue@external.euc.ac.cy (G.X.-G. Yue).

https://doi.org/10.1016/j.compeleceng.2024.110034
Received 5 June 2024; Received in revised form 9 December 2024; Accepted 20 December 2024
Available online 8 January 2025

(EC) [5]. However, the node based method often consumes more memory when facing a large search space. In the sampling based method, environmental sampling is required. Insufficient sampling leads to inferior path performance, while excessive sampling increases the computational burden of the entire planning process, reducing real-time capabilities in path planning. In the potential field method, when the distance between the robot and the obstacle exceeds the influence range of the obstacle, the robot is not affected by the repulsive potential field, thus it is prone to falling into local minima. Additional information regarding these global path planning algorithms can be found in Section 2.

As a significant subset of EC, swarm intelligence algorithms are a type of heuristic search algorithm based on the collective behaviour of agents, whose core lies in the use of agents in the swarm for information exchange and collaboration to achieve optimisation, showing excellent performance in solving Non-deterministic Polynomial (NP)-complete problems such as global path planning problem [6]. As a result, swarm intelligence algorithms, such as Particle Swarm Optimisation (PSO) [7] and Ant Colony Optimisation (ACO) [8], have been widely investigated by scholars since their inception. They have successfully been applied in many areas of global path planning, such as UAV trajectory planning [9–12], robot navigation [13–15], and logistics transportation [16].

As one of the most famous swarm intelligence algorithms, ACO simulates ants' information exchange and path selection process when searching for food, with strong global search ability and excellent group collaboration characteristics. This makes ACO promising to find high-quality solutions in complex environments compared to other algorithms. Most improvements to ACO focus on two aspects: hybridisation and multi-strategy enhancement. Hybridisation [17] combines ACO's advantages with other algorithms to achieve enhanced performance. Multi-strategy approaches [18] enable ants to compete and collaborate, exploring the solution space and avoiding being caught in local optima. Although these improvements can help ACO avoid static obstacles and obtain feasible paths in UAV path planning, they still have shortcomings in most practical tasks: (1) The algorithms are prone to struggling with finding high-quality solutions in complex path models like 3D environments with multiple constraints. Moreover, the convergence speed is also a challenge for ACO and its variants; (2) Most existing problem models are typically based on a simplified 2D or 3D environment in a discrete search space, which may not fully simulate the complexity and dynamic changes of the real world. When the 3D environment is partially unknown, the algorithms cannot effectively find a feasible path. Additionally, when the search space is continuous, it increases the number of possible paths, reducing search efficiency. Therefore, designing an algorithm that combines ACO and a local path planning algorithm in a complex 3D environment with continuous search space is necessary to enable UAV to fly collision-free and improve flight quality in the presence of static and moving obstacles.

To solve UAV path planning in a dynamic environment, scholars have proposed various local path planning methods, such as circle rapidly exploring random tree (circle-RRT) [19], optimised Q-learning (O-QL) [20], improved reinforcement learning [21] and dynamic window approach (DWA) [22]. Compared to other local path planning algorithms, DWA considers the kinematic constraints of robots and is easy to implement, with high computational efficiency, making it very suitable for integration with global planning algorithms to solve complex path planning problems.

Based on the above, we propose a novel variant of continuous ant colony optimisation with improved state transition probability, random walk strategy, and adaptive waypoints-repair method ($ACOSRA_R$) to address complex 3D path planning problems in continuous search space. Additionally, we successfully integrated the dynamic window approach (DWA) with $ACOSRA_R$ to solve UAV path planning problems in a partially unknown scenario with static and moving obstacles.

The main contributions of this paper are summarised as follows:

- A novel variant of continuous ant colony optimisation with improved state transition probability, random walk strategy, and adaptive waypoints-repair method is proposed for UAV global path planning to ensure collision-free and minimise cost travel to the destination.
- A new adaptive waypoints-repair method is presented which combines the single-coordinate axis repair approach with adaptive selection. This method effectively addresses path planning problems in complex environment with multiple obstacles, enhancing the feasibility of generated paths and the search efficiency of the algorithm.
- The effectiveness of the proposed algorithm is validated by comparing its results with those of competing algorithms in various complex scenarios generated using real digital elevation model (DEM) maps. We also effectively combined the dynamic window approach (DWA) with $ACOSRA_R$ to address UAV path planning in a partially unknown scenario with static and moving obstacles.

The rest of this paper is structured as follows: Section 2 provides an overview of traditional path planning algorithms and evolutionary computation algorithms. Section 3 outlines the procedure for defining the objective function. Section 4 proposes a detailed description of the proposed algorithm, followed by the presentation of experimental results and analysis in Section 5. Finally, Section 6 provides the conclusion and future research directions.

2. Related work

In this section, we introduce traditional path planning algorithms, including both global and local path planning, and the application of evolutionary computation algorithms in path planning.

2.1. Traditional path planning algorithms

Path planning problems can be categorised into global and local planning based on different environments [1]. Traditional global path planning algorithms include node-based methods, sampling-based methods, potential field algorithms, and coverage path planning (CPP) approaches. Dijkstra's algorithm [23] is a prominent node-based method that finds the shortest path from a

source node to all others in a weighted graph. Building upon the foundational concept it, A* can efficiently find the shortest path by balancing the actual cost incurred from the starting node and an estimated cost to reach the goal node simultaneously [2]. However, the node-based methods may struggle against modern challenges, as alternative algorithms can deliver similar performance with less computational overhead [24].

Probabilistic roadmap (PRM) is a popular sampling-based method for UAV path planning that involves randomly selecting nodes from the search space to create a graph [25]. However, during the exploration of the search space, sampling based methods may generate suboptimal paths, which require a large number of iterations to converge on an optimal solution [26].

The potential field method treats the robot as a point, guiding it from the starting point to the goal while avoiding obstacles by constructing a potential field in the environment [27]. A distance correction factor has been added to enhance the artificial potential field function for path planning [28]. While this method effectively avoids unknown obstacles, it can struggle with local minima [29].

In addition to existing path planning methods, coverage path planning (CPP) approaches have gained significant attention recently. For example, Bialas et al. [30] utilised deep reinforcement learning for CPP in complex 3D environments, demonstrating its effectiveness in dynamic settings. Similarly, Sanna et al. [31] developed a neural network-based algorithm for multi-UAV coverage path planning, emphasising collaboration. Bayerlein et al. [32] focused on multi-UAV path planning for wireless data harvesting using deep reinforcement learning, showing improved data collection efficiency.

In the field of local path planning and obstacle avoidance for robots in an unknown environment, a variety of traditional algorithms have been proposed and applied, including dynamic window approach (DWA) [22], time-elastic-band (TEB) [33,34], and model predictive control (MPC) [35]. However, the complex optimisation problems solved by MPC may lead to excessive computation times, while TEB optimises trajectories along the time dimension and is suitable for certain specific motion planning. DWA has fewer parameters and demonstrates significant advantages in a real-time responsive dynamic path planning environment. Therefore, it is more suitable to be combined with global path planning algorithms.

2.2. Evolutionary computation algorithms in path planning

In recent years, evolutionary computation algorithms have gained widespread popularity for solving path planning problems, as they are effective in addressing NP-complete issues. For example, a spherical vector PSO (SPSO) is proposed for solving global path planning problems of the UAV with multiple threats [9]. A hybrid chaotic aquila optimisation with simulated annealing (CAOSA) is designed for UAV trajectory planning [36]. An enhanced vulture optimisation algorithm (CCO-AVOA) is used to solve the UAV path planning problem [11]. An interval multi-objective particle swarm optimisation (IMOPSO) method is used in nuclear power plant [13]. Although these algorithms perform excellently in solving path planning problems, their fundamental algorithms often perform poorly in global search capabilities. Research on improving fundamental algorithms with strong global search abilities remains limited.

As one of the most famous evolutionary computation algorithms, Ant Colony Optimisation (ACO) [8] demonstrates excellent performance compared to other evolutionary computation algorithms in handling global path planning problems due to its robustness and powerful search capability. Nowadays, many scholars have used ACO and its variants to solve global path planning problems. For example, the ACO with A* search (ACO-A*) is established on a two-layer algorithm model to deal with autonomous underwater vehicles (AUVs) path planning [37]. The aging-based ACO (ABACO) is proposed to address the problem of navigating the safest and most optimal path for mobile robots in a 2D environment [38]. The fuzzy logic ACO (FLACO) can identify the most cost-effective shortest virtual path for unmanned vehicles [39]. Due to the outstanding performance of ACO in searching for global optimal solutions, choosing to combine it with local path planning algorithms is a better choice. However the previous investigations have made the process of path planning more manageable and computationally efficient by creating simplified graphical representations within a discrete environment, these approaches may sometimes overly simplify the actual environment, potentially leading to suboptimal solutions that may not be directly applicable to partially unknown environments with continuous search space.

ACO_R [40] was proposed, converting ACO's discrete probabilistic model into a seamless probability density function (PDF). It is the first time ACO has also been extended to continuous domain problems. After that, a substantial amount of research has been dedicated to solving continuous space path planning problems using ACO_R. For example, Liu et al. [41] proposed a multi-strategy ACO_R to address path planning problems in 2D environments with continuous search space while it neglected the potential presence of moving obstacles in practical environments. Due to the complexity of models in continuous search space, most literature has not taken into account the presence of unknown obstacles in 3D environments, which limits the applicability and effectiveness of algorithms when dealing with complex real-world scenarios. Research on ACO_R for path planning in 3D partially unknown environment with continuous search space is relatively limited.

3. Problem description

In this paper, we improve the comprehensive flight cost function presented by Phung et al. [9], which considers various factors, including the distance, potential collisions, altitude, and turning angles (as shown in Fig. 1). We consider the possibility of the UAV flying over obstacles, which is not mentioned in the original model.

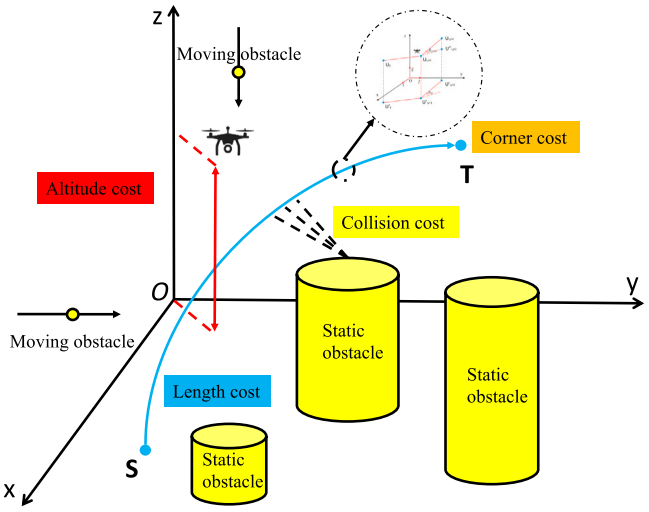


Fig. 1. The cost of the UAV path planning.

3.1. Path length cost

To ensure efficient UAV operation and reduce the UAV's flight time, the planned path must be optimal according to specific criteria based on the application and path length is given primary consideration. In general, the 3D spatial position of the UAV of the jth path point in the ith flight path is represented by $U_{ij} = (x_{ij}, y_{ij}, z_{ij})$. The whole flight path X_i is a path list containing n waypoints. The Euclidean distance between two waypoints is denoted as the path segment $\|U_{ij}U_{i,j+1}\|$, and the path length cost is denoted as:

$$F_1(X_i) = \sum_{j=1}^{n} \left\| U_{ij}U_{i,j+1} \right\| \tag{1}$$

3.2. Threat and collision cost

Safety is another constraint that needs to be considered in UAV path planning, which implies that the UAV has no collision with obstacles. In this paper, we set K cylinders with centre C_k, radius R_k, and height h as obstacles on the path, and consider the UAV size D and the threat distance from the UAV approaching the obstacle L. Fig. 2(a) shows the positional relationship between the path segment $\|U_{ij}U_{i,j+1}\|$ and the obstacle or threat region in 3D space. We establish the threat cylinder equation $F(R_k + D + L)$, obstacle cylinder equation $F(R_k + D)$, and line segment equation $F(LS)$.

$$F_{i1} = F\left(R_k + D + L\right) - F(LS). \tag{2}$$

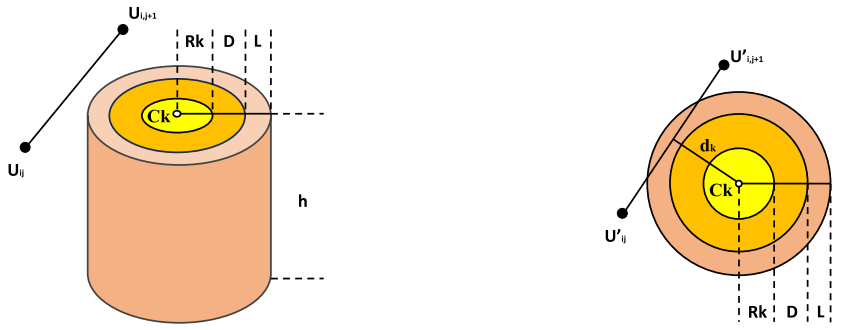
$$F_{i2} = F\left(R_k + D\right) - F(LS). \tag{3}$$

The intersection of the line segment with a cylinder depends on the discriminant of F_{i1} and F_{i2}. The cost calculation for threat and collision cost is shown in Eq. (4).

$$F_2(X_i) = \sum_{j=1}^{n-1} \sum_{k=1}^{K} T_k\left(\overrightarrow{U_{ij}U_{i,j+1}}\right). \tag{4}$$

$$T_k\left(\overrightarrow{U_{ij}U_{i,j+1}}\right) = \begin{cases} 0, & \text{if } \Delta 1 < 0, \Delta 2 < 0 \\ R_k + L + D - d_k, & \text{if } \Delta 1 > 0, \Delta 2 < 0 \\ \infty, & \text{otherwise .} \end{cases} \tag{5}$$

where $\Delta 1$ and $\Delta 2$ are discriminants of F_{i1} and F_{i2}. d_k is the distance between the projection of the line segment to the projection of the cylinder's centre when the line segment intersects the cylinder, as shown in Fig. 2(b).



(a) 3D view **(b)** Top view

Fig. 2. Path segment and obstacle/threat view.

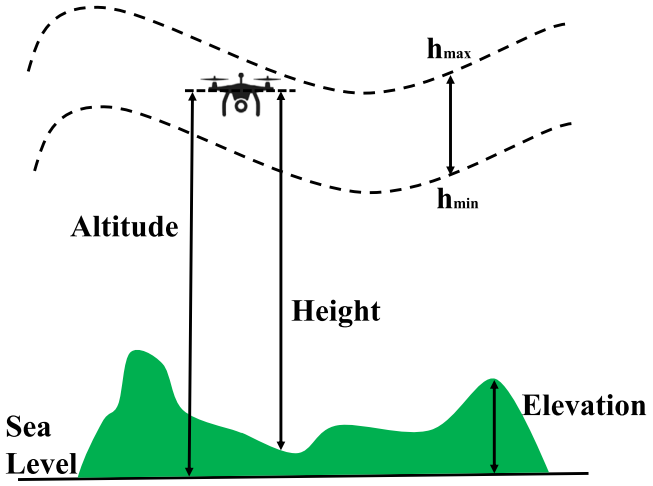


Fig. 3. Altitude cost explanation.

3.3. Altitude cost

Altitude is also a limiting factor during the UAV flight to ensure its stability. Calculating altitude cost helps avoid unnecessary altitude changes, which can lead to increased energy consumption and reduced operational efficiency. Relative to the ground, it is assumed that the minimum and maximum flight heights of the UAV are $h_{\min}$ and $h_{\max}$, respectively. We penalise flight heights that are not in the prescribed range. Specifically, the cost of the UAV's altitude at the waypoint U_{ij} is:

$$H_{ij} = \begin{cases} \left| h_{ij} - \frac{(h_{\max}+h_{\min})}{2} \right|, & \text{if } h_{\min} \le h_{ij} \le h_{\max} \\ \infty, & \text{otherwise .} \end{cases} \tag{6}$$

Adding up the altitude costs for all waypoints is the total altitude cost (see Fig. 3) :

$$F_3\left(X_i\right) = \sum_{j=1}^{n} H_{ij} \tag{7}$$

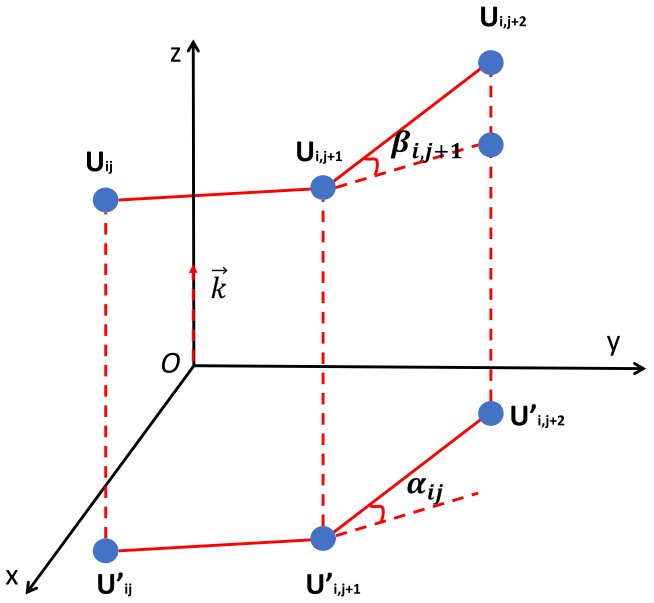


Fig. 4. Corner cost explanation.

3.4. Corner cost

Adjacent segments on the path form a corner. Considering corner cost is important for avoiding sharp turns, which can reduce flight stability and increase the risk of collisions. In 3D space, the horizontal corner parameter is controlled by the turning angle α, while the vertical corner parameter is controlled by the climbing angle β. Their calculation methods are as follows:

$$\alpha_{ij} = \arctan\left(\frac{\left\|\overrightarrow{U'_{ij}U'_{i,j+1}} \times \overrightarrow{U'_{i,j+1}U'_{i,j+2}}\right\|}{\overrightarrow{U'_{ij}U'_{i,j+1}} \cdot \overrightarrow{U'_{i,j+1}U'_{i,j+2}}}\right). \tag{8}$$

$$\beta_{ij} = \left(\frac{z_{i,j+1} - z_{ij}}{\left\|\overrightarrow{U'_{ij}U'_{i,j+1}}\right\|}\right). \tag{9}$$

$$\overrightarrow{U'_{ij}U'_{i,j+1}} = \vec{k} \times \left(\overrightarrow{U_{ij}U_{i,j+1}} \times \vec{k}\right) \tag{10}$$

where $\vec{k}$ is the unit vector on the z-axis, and $\overrightarrow{U'_{ij}U'_{i,j+1}}$ is the projected vector of $\overrightarrow{U_{ij}U_{i,j+1}}$ onto the Oxy plane. Then, the cost function F_4 for the UAV flight corner is expressed as follows, where γ_1 and γ_2 are penalty coefficients (see Fig. 4).

$$F_4(X_i) = \gamma_1 \sum_{j=1}^{n-2} \alpha_{ij} + \gamma_2 \sum_{j=1}^{n-1} \left|\beta_{ij} - \beta_{i,j-1}\right| \tag{11}$$

3.5. The total flight cost function

Based on the above, we construct a multi-factor flight cost function considering the path length, collision, altitude, and corner constraints as shown in Eq. (12), where k is the number of cost functions and b_k is the weight coefficient.

$$F(X_i) = \sum_{k=1}^{4} b_k F_k(X_i) \tag{12}$$

4. ACOSRA$_R$ for path planning

In this section, we introduce the standard continuous ACO (ACO$_R$), the proposed algorithm ACOSRA$_R$ for global path planning, and the application of ACOSRA$_R$ combined with the Dynamic Window Approach (DWA) for UAV path planning in 3D environments with static and moving obstacles.

4.1. Standard continuous ant colony optimisation

ACO_R's structure was first proposed by Socha [40] for ACO to address the optimisation problem in the continuous domain, and the core idea is to use a Gaussian distribution to simulate the pheromone distribution in the algorithm. In general, we define a continuous domain problem as follows:

$$\min_{x \in D} f(x) \in R \tag{13}$$

where x is a vector belonging to the continuous domain D, and f is the objective function. We assume there is a solution archive which has k solutions, each solution is represented as $X_i^d = \{x_i^d, d = 1, 2, \ldots, D, i = 1, 2, \ldots, k\}$. During the iteration process, the solutions within each archive are sorted by their fitness values in descending order. Each solution is assigned a weight coefficient according to Eq. (14):

$$w_i = \frac{1}{l \cdot k \cdot \sqrt{2\pi}} \cdot e^{-\frac{(i-1)^2}{2 \cdot l^2 \cdot k^2}} \tag{14}$$

where a larger l implies that the selected solutions tend to be more uniformly distributed, while a smaller l implies that the selected solutions tend to be closer to the optimal value. Then, Eq. (15) is used to calculate the probability of each solution being selected as a guiding solution.

$$p_{si} = \frac{w_i}{\sum_{r=1}^{N} w_r} \tag{15}$$

Subsequently, novel solutions are generated through a Gaussian distribution sampling process, setting the mean to $u_i^d = x_i^d$ and the standard deviation to σ_i^d. This computation adheres to Eq. (16) where σ_i^d signifies the average extent across dimension dth from the guiding solution x_i^d to within the archive, as delineated by Eq. (17).

$$\mathrm{x}_{\mathrm{new}}^d \leftarrow \mathcal{N}\left(u^d, \sigma_i^d\right), d = 1, 2 \ldots, D \tag{16}$$

$$\sigma_i^d = \xi \cdot \sum_{r=1}^{k} \frac{\left|x_r^d - x_i^d\right|}{k-1}, d = 1, 2 \ldots, D \tag{17}$$

where the parameter ξ influences the convergence rate of the algorithm. A higher value of ξ results in a slower convergence.

4.2. Improved continuous ant colony optimisation

4.2.1. Improved state transition probability

The probability of each solution being selected as a guiding solution is calculated by Eq. (15) in original ACO_R, which is highly random and exploratory but inefficient.

To enhance the algorithm's local exploitation capability and solution efficiency, we introduce the greedy algorithm mechanism to improve the state transition probability as shown in Eq. (18). The greedy algorithm involves selecting local optima at each step of the problem-solving process without considering the global optima. Before the ants select the next node, we calculate and rank the probabilities of all potential points. If $q < q_0$, we choose the node with the highest probability as the next state node for the ants. Otherwise, we use the traditional state selection method.

$$x_i = \begin{cases} \operatorname{argmax}\left(p_{si}\right), & q < q_0 \\ p_{si}, & q \geq q_0 \end{cases} \tag{18}$$

$$q_0 = m \cdot e^{-\lambda\left(\frac{k}{K}\right)}, k = 1, 2 \ldots, n \tag{19}$$

where q follows a uniform distribution over the range $[0, 1]$. m is a proportion coefficient with a value in the $[0.1, 0.5]$ range. λ is an adjustment coefficient, K is the maximum number of iterations, and k is the current iteration number. To ensure ants can choose their preferred paths to accelerate convergence speed in the early stage and strengthen the global search ability to prevent convergence to local optima in the later stage, we design an adaptive transition probability threshold q_0 which is calculated in Eq. (19) and its variation is shown in Fig. 5.

4.2.2. Random walk strategy

The new solutions generated in ACO_R are based on the construction of the Gaussian distribution. This means that the random walk strategy in ACO_R follows a Brownian motion pattern with step length conforming to the Gaussian distribution, which keeps the new solutions relatively close to the current solution, potentially leading to local optima in the later stage. To address this shortcoming of ACO_R, we adopted Lévy flight to improve the exploration ability and refine the discovery process. In Lévy flight, each step in the random walk is dictated by a power-law distribution, assigning a probability to step lengths greater than μ that is proportional to the negative α power of the step length, expressed as $P(z > \mu) \propto |z|^{-\alpha}$. α plays a role in the distribution index, determining the level of stability, typically ranging in $[0, 2]$. The probability distribution of step length usually has a long tail, which implies the existence of a few large jumps and a majority of short movements in Lévy flight. This power-law distribution property can make Lévy flight strengthen its global exploration ability and help to jump out of the local optima. In recent years, Lévy flight

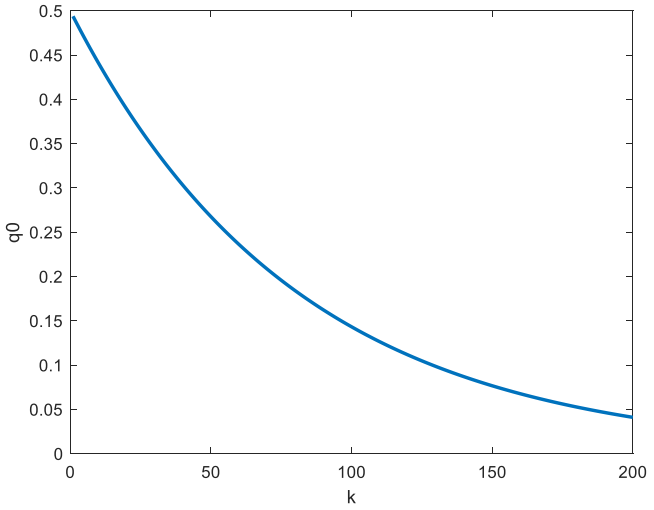


Fig. 5. q_0 of each iteration.

has been applied to many optimisation algorithms such as DE [42] and PSO [43].

We designed a new random walk strategy, where the algorithm assigns a random number to each solution in the archive. If the random number is less than p_r, each dimension of the solution adopts a step length following a Gaussian distribution, and the new solution is calculated using Eq. (16). Otherwise, it adopts the step length of Lévy flight, and the new solution is determined by Eq. (20). The random walk strategy strives to achieve a balance between exploration and exploitation of the algorithm.

$$x_{new}^{d} \leftarrow x_i^d + \sigma_i^d \cdot \text{levy}(\alpha), d = 1, 2 \ldots, D \tag{20}$$

$$p_r = p_f + \left(p_0 - p_f\right) \cdot e^{-\mu \cdot k}, k = 1, 2 \ldots, i \tag{21}$$

where levy(α) is the step length in the Lévy flight, p_r is the selection probability for each solution as shown in Fig. 6. u is the acceleration coefficient. The p_r values range from p_0 to p_f for the solutions from the 1st to the kth.

4.2.3. Adaptive waypoints-repair method

In a path planning problem, the search space of the algorithm is divided into feasible and infeasible areas by obstacles, constraint boundaries, or other limited factors. During the search process, the algorithm may generate waypoints in the infeasible domain. For example, an ant may construct a waypoint that exceeds the constraint boundary or is located inside an obstacle, implying that the path is infeasible. To reduce the difficulty of the searching, we fully consider the distribution of obstacles and propose an adaptive waypoints-repair method (Algorithm 1), which consists of two parts: single-coordinate axis repair approach and adaptive selection.

Algorithm 1 Adaptive waypoints-repair method

Require: X_{new}, E
for $d = 1$ to D **do**
  Choose waypoints which are in unfeasible areas
  if rand $< c_1$ **then**
    Select the first repair approach: moving along the x-axis
  else if $c_1 \leq$ rand $< c_2$ **then**
    Select the second repair approach: moving along the y-axis
  else
    Select the third repair approach: moving along the z-axis
  end if
end for
Evaluate the fitness $f(X)$
Update the E_j^t according to Eq. (23)
Update the p_{rwj}^t according to Eq. (22)
return $X_{\text{new}}, f(X)$

Algorithm 1 shows the adaptive waypoints-repair method to move infeasible waypoints in UAV path planning. It begins by randomly selecting one of the repair strategies to adjust waypoints positions and enhance the path's feasibility. Subsequently, it

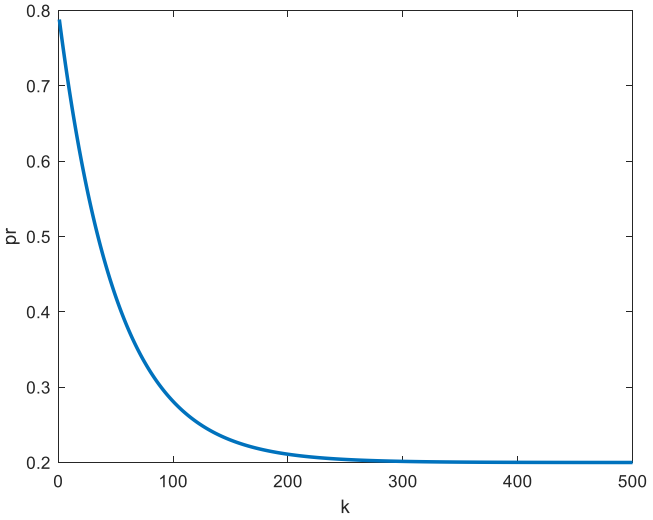


Fig. 6. p_r of each iteration.

continuously updates parameters and learns from experience to assist the UAV in making wiser choices during subsequent path planning. As shown in Fig. 7(a), the flight path of the UAV is composed of several waypoints and line segments between them. The coordinates corresponding to each waypoint are represented as (x, y, z). We assume that the UAV can only change one dimension of the 3D coordinates in the process of repairing infeasible waypoints. For example, Fig. 7(b) shows the projection of Fig. 7(a) on the Oxy plane. The single-coordinate repair approach will traverse each infeasible waypoint along the path and identify that X_3 is within an infeasible domain. Then, it will move along the x-axis. From Fig. 7(b), we can see that the domain along the x-axis is divided into feasible domains $[x'_1, x'_2]$, $[x'_3, x'_4]$, $[x'_5, x'_6]$ and infeasible domains $[x'_2, x'_3]$, $[x'_4, x'_5]$. The shifted waypoints fall into random locations within the feasible domain, thus improving the feasibility of the path. For each solution, three repair approaches are provided for selection, including movement along the x-axis, y-axis, and z-axis, as described below:

- Moving along the x-axis: randomly move impractical waypoint along the x-axis to a position within the entire feasible range.
- Moving along the y-axis: randomly move impractical waypoint along the y-axis to a position within the entire feasible range.
- Moving along the z-axis: randomly move impractical waypoint along the z-axis to a position within the entire feasible range.

Due to the constraints on both the angle and length of each path segment, we cannot ensure the feasibility of the path even if all waypoints are located within the feasible domain. Therefore, continuous iteration based on excellent solutions is crucial. Adaptive selection is proposed to enhance the quality of solutions [41,44]. Better solutions are more likely to be selected during the iteration process. The probability of selecting the jth remediation strategy throughout the tth cycle of execution is as follows:

$$p^t_{rwj} = \frac{E^t_j}{\sum_{j=1}^{3} E^t_j} \tag{22}$$

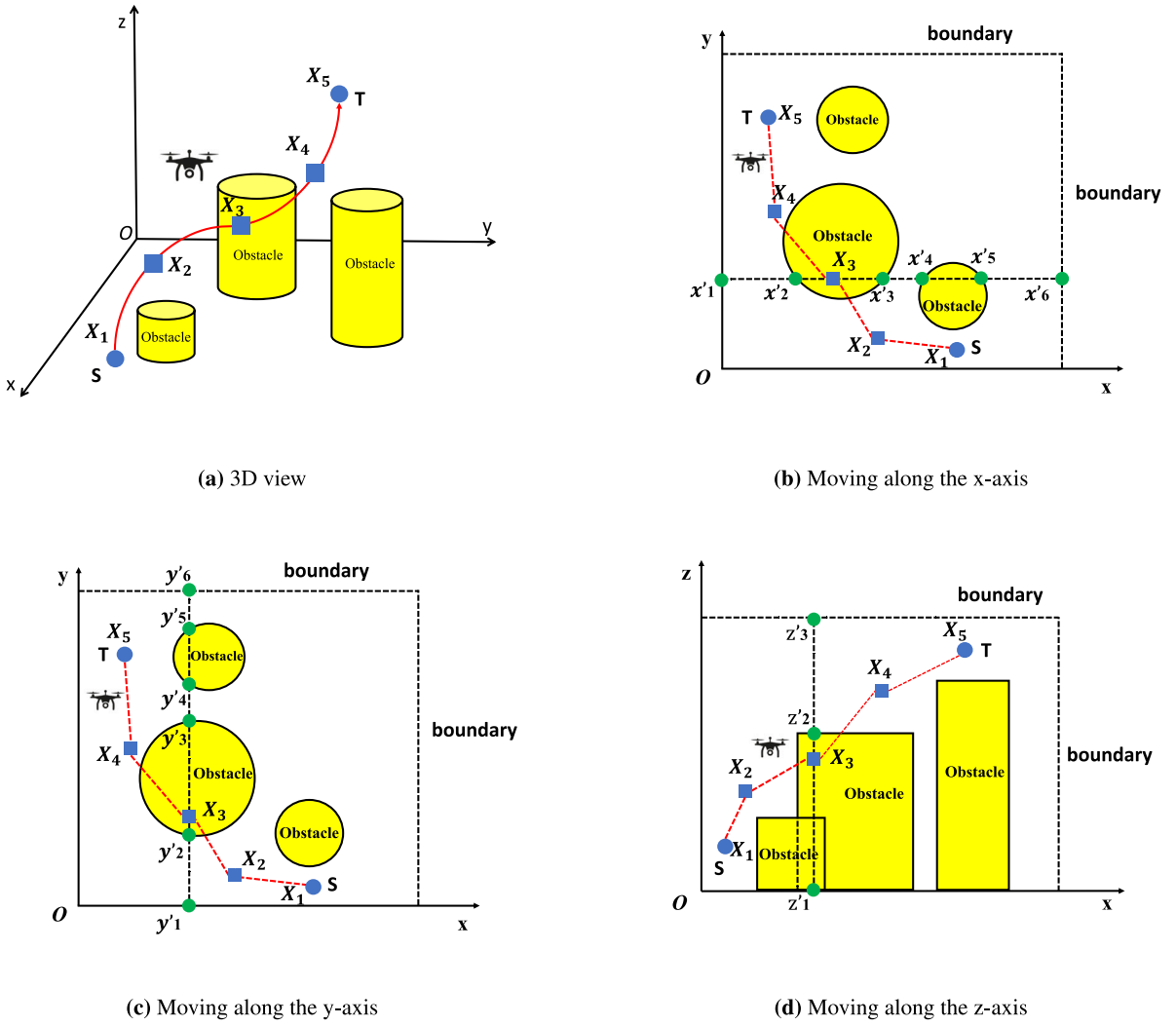
where E^t_j represents the experience of jth remediation strategy throughout the tth cycle of execution, calculated as follows:

$$E^t_j = E^{t-1}_j + \left| f^{t-1}_b - f^t_b \right| \tag{23}$$

where f^t_b represents the optimal fitness value achieved by the tth iteration, and $f^{(t-1)}_b$ denotes the optimal fitness value from $(t-1)$th iteration.

4.2.4. $ACOSRA_R$

By combining improved state transition probability, random walk strategy and adaptive waypoints-repair method, we design a new meta-heuristic algorithm named $ACOSRA_R$ to address path planning problems in 3D environments with continuous search space. The pseudo code is as Algorithm 2, detailing the logical flow and main steps of the algorithm.



(a) 3D view

(b) Moving along the x-axis

(c) Moving along the y-axis

(d) Moving along the z-axis

Fig. 7. The adaptive waypoints-repair method.

From Algorithm 2, we can see that the proposed algorithm randomly selects k initial solutions from the archive, evaluates their fitness, and sorts them. Then, it computes probabilities to choose appropriate methods for updating solutions, including Gaussian distribution and Lévy flight. The algorithm performs repair operations for infeasible solutions during each iteration and resorts the solution space based on fitness values. Ultimately, when the stopping criteria are met, the algorithm returns the optimal solution and its corresponding objective function value.

4.2.5. Time complexity analysis

To evaluate the efficiency of the ACOSRA_R in solving practical problems, we need to analyse its time complexity, which is an important metric typically represented using big-O. In the proposed algorithm, N is the population size, n_1 is the number of waypoints, n_2 is the number of obstacles, and max_iter is the maximum number of iterations. Based on this assumption, the time complexity analysis of the algorithm is as follows:

The proposed algorithm can be divided into two main stages. The first stage, population initialisation, is executed only once. During this process, the repair operation is required to ensure the feasibility of the generated solutions, which involves calculating the relationship between each waypoint and each obstacle, resulting in a time complexity of $O(N \times n_1 \times n_2)$. The second stage is the optimisation process performed in each iteration. This stage can be further broken down into four key components: ant generation and mutation, repair operation, fitness evaluation, and the update and selection of the global best solution.

Algorithm 2 The pseudo code of ACOSRA_R algorithm

Require: D, N, k, ξ, q, objective function $f(X) = F_{cost}$, environment information
Initialise: select k solutions X from archive randomly, $E_j = 1$, $j = 1, 2, 3$, $t = 1$
Evaluate the fitness $f(X)$
Arrange solutions in ascending order based on $f(X)$
Compute p_s using Eqs. (14) and (15)
Compute p_r using Eq. (21)
while $t \leq$ max_iterations **do**
  for $i = 1$ to N **do**
    Select a solution X_i from the archive of solutions using Eq. (18) and let $\mu = X_i$
    Calculate the standard deviation σ_i using Eq. (17)
    for $d = 1$ to D **do**
      if rand $< p_r$ **then**
        $x^d_{\text{new}} \leftarrow N(\mu^d, \sigma^d_i)$
      else
        $x^d_{\text{new}} \leftarrow \mu^d + \sigma^d_i \cdot \text{levy}(\alpha)$
      end if
    end for
  end for
  Execute **Algorithm 1** to repair infeasible solutions
  Incorporate X_{new} into the solution archive, then arrange all solutions in ascending order and remove the N worst solutions
  $t \leftarrow t + 1$
end while
return the optimal solution X_1, and its corresponding objective value $f(X_1)$

• Generation and mutation: In this paper, we use spherical coordinates. Therefore, the update of each position of an ant (r, ψ, ϕ) includes the selection of Gaussian mutation or Lévy flight. The time complexity of the update operation for each coordinate of each ant is $O(1)$. However, since each ant has multiple dimensions(r, ψ, ϕ), the time complexity for each update in each iteration with N ants is $O(N \times 3) = O(N)$.

• Repair operation: The repair operation for each ant involves traversing all obstacles to check the distance. Therefore, the time complexity of total population is $O(N \times n_1 \times n_2)$.

• Fitness evaluation: Each time the ant's cost function (fitness) is evaluated, the algorithm computes the path length, the effect of obstacles, the flight height, etc. The time complexity for each evaluation is $O(n_1 \times n_2)$; Therefore, the total fitness evaluation time complexity for all ants in each iteration is $O(N \times n_1 \times n_2)$.

• Global best solution update and selection: The sorting and selection operations are typically done using quicksort or mergesort, with a time complexity of $O(N \times logN)$.

In summary, the total time complexity of the algorithm is $O(max\_iter \times (N \times n_1 \times n_2 + N \times logN))$. When $N \gg n_1$ and $N \gg n_2$, the time complexity simplifies to $O(max\_iter \times (N \times logN))$, which is consistent with the traditional ACO_R. Thus, we demonstrate that the proposed algorithm does not increase the time complexity and maintains good execution speed.

4.3. $ACOSRA_R$ with DWA for UAV path planning

In this section, we introduce the principles of the dynamic window approach (DWA) and combine ACOSRA_R with DWA to address the problem of UAV path planning in partially unknown environments.

4.3.1. Dynamic window approach

As a well-known local path planning algorithm, the dynamic window approach (DWA) was first proposed by Fox [22] for mobile robot local obstacle avoidance. The mobile robot can determine a sampling velocity in the velocity space (v, ω) that satisfies the constraints according to the current environmental changes and kinematic models, including linear velocity, acceleration, angular acceleration, and dynamic obstacles. Then, the trajectory of the mobile robot in a certain period under the velocity is computed and evaluated by an evaluation function to select the best-evaluated trajectory, and this process continues until the mobile robot reaches the target point.

4.3.2. $ACOSRA_R$ with DWA algorithm flow for UAV path planning

In a known environment with continuous search space, ACOSRA_R can successfully plan a global path. However, during the actual flight of a UAV, the environment is often partially unknown. Therefore, we proposed ACOSRA_R with DWA, it can effectively avoid both moving and static obstacles during the UAV's flight. Firstly, we decompose the global path generated by ACOSRA_R into a series of waypoints that serve as a reference for DWA. Then, we use the global path information to generate sampling velocities and calculate trajectories. Finally, we select the trajectory with the optimal evaluation. During the decision-making process of DWA, we

use the global path as a hard constraint to ensure that the UAV does not deviate too far when planning locally. The pseudo code of $ACOSRA_R$ with DWA is shown in Algorithm 3.

Algorithm 3 The pseudo code of $ACOSRA_R$ with DWA for UAV path planning

Require: $X_{new}, n, f(X)$, objective function $f(X)$, environment information
Generate n local sub-targets along the global path
for $i = 1$ to n **do**
  Calculate velocity sampling space $V_s(v, \omega)$ according to the hardware, structure, and environment of the UAV
  $V_{s,j} = V_m \cap V_d \cap V_a$ for $j = 1, 2, \ldots, m$
  for $j = 1$ to m **do**
    Generate trajectories $Path1, Path2, \ldots, Path_m$ based on robot's kinematic model
    Calculate each trajectory cost according to heading angle $h(v, \omega)$, distance to moving obstacles $d(v, \omega)$, and velocity $v(v, \omega)$
    $C(v, \omega) = \alpha h(v, \omega) + \beta d(v, \omega) + \gamma v(v, \omega)$
  end for
  Choose the best local trajectory according to the cost
end for
return the best path X_{new}

5. Experimental results and analysis

5.1. Experimental setting

In order to assess the effectiveness of the $ACOSRA_R$ algorithm in global path planning, this paper conducted two sets of experiments in simulated environments which are based on precise digital elevation model (DEM) maps provided by Phung et al. [9]. The first set was used to analyse the specific impact of each improvement strategy on the algorithm performance. The second set compared $ACOSRA_R$ with several other advanced algorithms (PSO [7], ACO_R [40], $ACOPA_R$ [41], SPSO [9], EEFO [45], HLOA [46]and CCO-AVOA [11]) in 9 cases with different numbers of obstacles and waypoints to verify its applicability in diverse scenarios. Table 1 displays the specifications for both the tasks and the static environment information. The UAV travels from the start point S to the target point T, which are denoted as S_i $(x_{s_i}, y_{s_i}, z_{s_i}, h_{s_i})$ and T_i $(x_{g_i}, y_{g_i}, z_{g_i}, h_{g_i})$ respectively. The static obstacles are denoted as O_i $(x_i, y_i, z_i, R_i, h_i)$. The default length unit in the environment is a meter. In the two sets of experiments, we ran each case 30 times to ensure the reliability and statistical significance of the results, with the best cost value in bold for each case. Wilcoxon rank-sum test was used to examine statistical significance. For each case, if the best result significantly differed from the results of other algorithms at a 5% level, it was marked with *. Across all experiments, the parameter T represents the maximum iteration count, set to 200, and the total population size n is set to 500.

To test the ability of $ACOSRA_R$, when combined with the Dynamic Window Approach (DWA), to avoid static and moving obstacles in the environment, we conducted a simulation experiment on case 3. In the experiment, we set the moving obstacles as spheres moving back and forth in a straight line within the environment. The source codes for the algorithms can be accessed at: https://github.com/Yongjin30826.

5.2. Analysis of experimental results

5.2.1. The impacts of various strategies

We conducted an in-depth ablation analysis of the proposed algorithm to evaluate the contribution of specific components to the overall performance by gradually removing them from the algorithm. We compared the basic ACO_R and its variants (ACO_R-S, ACO_R-R and ACO_R-A) with the proposed $ACOSRA_R$. ACO_R-S is the ACO_R with only the improved state transition probability. ACO_R-R is the ACO_R with only the random walk strategy and ACO_R-A is the ACO_R with only the adaptive waypoints-repair method. In the 10D path planning problem, we tested and compared the performance of these algorithms across 9 cases. Tables 2 and 3 show the results and statistical analysis of different strategies in the 10D case. The ↓ indicates that a smaller value represents better performance.

From Table 2, we can observe that the performance of $ACOSRA_R$ significantly outperforms all other algorithms in all cases, indicating the excellent performance of ACO_R in path planning after integrating three strategies. It shows that the combined results of the three strategies exceed the performance of any individual strategy.

To facilitate the study of the enhancement effect of each strategy on the algorithm, we conducted a Wilcoxon rank-sum test to compare the algorithm results presented in Table 3 for each pair. s* and i* represent that algorithm B in 'Algorithm A vs. Algorithm B' is significantly superior or inferior to A. - denotes no statistically significant difference between them. From Table 3, we observe that ACO_R-S significantly outperforms ACO_R in most cases. This result reveals the positive impact of the improved state transition probability on the performance of path planning. Although ACO_R-S performs worse than ACO_R in 2 cases, the Wilcoxon rank-sum test indicates no significant difference between the two algorithms. This suggests that while ACO_R-S may be more prone to getting stuck in local optima in certain situations, it does not change the overall trend that ACO_R-S can find better paths.

Table 1
Task and environment information of 9 cases.

Case		Details
1	S;T O	(200,100,150); (800,800,150) O1(300,200,150,100,250); O2(500,500,150,100,250); O3(650,550,150,100,100); O4(80,180,150,80,100); O5(500,700,150,179,500); O6(700,800,150,70,100); O7(920,680,150,120,100); O8(500,360,150,110,500)
2	S;T O	(250,80,150); (950,650,150) O1(200,320,150,100,350); O2(430,150,150,150,350); O3(440,520,150,130,350); O4(750,550,150,100,350); O5 (650,320,150,90,170)
3	S;T O	(220,80,150); (850,750,150) O1(170,330,150,130,350); O2(430,150,150,130,350); O3(300,240,150,50,200); O4(750,640,150,70,180); O5(500,450,150,70,350);O6(620,730,150,100,350); O7(880.550,150,100,350)
4	S;T O	(220,80,150); (850,750,150) O1(380,460,150,80,350); O2(720,320,150,70,350); O3(550,390,150,80,350); O4(740,560,150,60,150); O5(350,240,150,60,350); O6(540,140,150,60,350); O7(570,620,150,60,350)
5	S;T O	(360,100,150); (850,750,150) O1(300,380,150,130,350); O2(580,190,150,150,350); O3(590,540,150,145,350); O4(585,350,150,80,150); O5(800,630,150,50,150); O6(850,550,150,80,350)
6	S;T O	(200,100,150); (850,750,150) O1(400,230,150,90,350); O2(250,310,150,80,200); O3(550,140,150,80,200); O4(550,450,150,80,200); O5(360,580,150,150,350); O6(750,350,150,150,350); O7(950,620,150,85,350); O8(650,730,150,120,350)
7	S;T O	(280,100,150); (1000,600,150) O1(240,450,150,135,350); O2(450,160,150,135,350); O3(520,400,150,90,200); O4(550,450,150,80,200); O5(360,580,150,150,350)
8	S;T O	(200,100,150); (850,750,150) O1(120,360,150,120,350); O2(850,590,150,110,350); O3(350,220,150,120,350); O4(450,590,150,110,350); O5(650,460,150,100,150); O6(650,305,150,150,350); O7(650,100,150,100,350)
9	S;T O	(220,130,150); (960,430,150) O1(150,420,150,100,350); O2(760,720,150,130,350); O3(400,220,150,160,350); O4(450,600,150,160,350); O5(500,400,150,100,150); O6(800,400,150,130,350)

Table 2
Comparison of results in terms of total cost (↓) among algorithms with different strategies in 10D.

Case	ACO_R [40]	ACO_R-S	ACO_R-R	ACO_R-A	$ACOSRA_R$
1	5908 ± 295*	5747 ± 484*	6037 ± 104*	5661 ± 69*	**5322 ± 389**
2	5227 ± 468*	5181 ± 434*	5121 ± 358*	5880 ± 275*	**4967 ± 206**
3	5665 ± 229*	5718 ± 388*	5510 ± 289*	5653 ± 137*	**5135 ± 393**
4	5361 ± 139*	5304 ± 125*	5141 ± 203*	5199 ± 97*	**4792 ± 102**
5	5698 ± 82*	5701 ± 273*	5720 ± 95*	5638 ± 31*	**4796 ± 381**
6	6395 ± 503*	6326 ± 639*	5493 ± 593*	6605 ± 142*	**4877 ± 169**
7	5281 ± 444*	5193 ± 442*	5050 ± 390*	5430 ± 215*	**4714 ± 225**
8	6381 ± 618*	5516 ± 935*	6483 ± 919*	6114 ± 242*	**4635 ± 102**
9	9466 ± 119*	8816 ± 123*	8737 ± 654*	9203 ± 374*	**6625 ± 1416**

Table 3
Comparison of statistical analysis results across different versions of ACO in 10D.

Case	ACO_R [40] vs. ACO_R-S	ACO_R [40] vs. ACO_R-R	ACO_R [40] vs. ACO_R-A	ACO_R [40] vs. $ACOSRA_R$
1	s*	i*	s*	s*
2	s*	s*	i*	s*
3	–	s*	s*	s*
4	s*	s*	s*	s*
5	–	i*	s*	s*
6	s*	s*	i*	s*
7	s*	s*	–	s*
8	s*	–	s*	s*
9	s*	s*	s*	s*

ACO_R-R better addresses the issue of ACO_R getting trapped in local optima in some instances. Table 3 illustrates that when we introduce the random walk strategy into the algorithm, although ACO_R-R performs worse than ACO_R in 2 cases, in 5 cases, it significantly outperforms ACO_R compared to ACO_R-S. Specifically, in cases 2, 4, 6, 7, and 9, ACO_R-R achieves an average improvement of 7% over ACO_R, which is higher than the average improvement of 3% by ACO_R-S over ACO_R. This finding illustrates the importance of the random walk strategy in the algorithm, as it strikes a good balance between exploration and exploitation by incorporating two different step sizes, thus avoiding being trapped in local optima.

Table 4
Total cost (↓) of different parameter combinations.

μ, λ	0.02, 0.05	0.05, 0.15	0.10, 0.25	0.30, 0.35	0.50, 0.50
Case1, 10D	5588 ± 338*	**5322 ± 389**	5692 ± 251*	5819 ± 201	5971 ± 113*
μ, λ	0.02, 0.50	0.05, 0.35	0.10, 0.15	0.30, 0.15	0.50, 0.05
Case1, 10D	5523 ± 391*	5593 ± 244*	5718 ± 225*	5774 ± 121	5911 ± 188*

Table 5
Parameter settings in various comparison algorithms.

Algorithm	Parameters	Reference
PSO [7]	$w = 0.729,\ c1 = c2 = 1.496$	Kennedy and Eberhart (1995)
ACO_R [40]	$\xi = 0.6,\ l = 0.2$	Socha and Dorigo (2008)
$ACOPA_R$ [41]	$\xi = 0.6,\ l = 0.2$	Liu et al. (2022)
SPSO [9]	$w(t) = w(t-1) \times \text{damp},\ w(0) = 1,\ \text{damp} = 0.98,\ c1 = c2 = 1.5$	Phung and Ha (2021)
EEFO [45]	Does not use additional parameters	Zhao et al. (2024)
HLOA [46]	Does not use additional parameters	Peraza-Vázquez et al. (2024)
CCO-AVOA [11]	$\alpha = 0.8,\ \beta = 0.2,\ \gamma = 2.5,\ P_1 = 0.6,\ P_2 = 0.4,\ P_3 = 0.6,\ L_0 = 0.7$	Ait-Saadi et al. (2023)
$ACOSRA_R$	$\xi = 0.6,\ l = 0.2,\ u = 0.05,\ \lambda = 0.15$	Our work

Although in some cases ACO_R-A may not significantly surpass the ACO_R, it maintains high-quality solutions while reducing performance fluctuations. This indicates that ACO_R-A provides better adaptability when facing different obstacles and constraints. It can adjust the intensity and direction of the repair strategy according to the specific problem, thus effectively balancing exploring new solutions and utilising known ones. However, it is also important to note that the adaptive repair method is not a panacea. In some cases, excessive repairs may cause the algorithm to overlook potentially promising exploration directions. The effect of a single strategy may vary depending on the case, while the $ACOSRA_R$ integrates all mentioned strategies into a comprehensive approach. It optimises the search process through the improved state transition probability, increases the likelihood of exploring new solutions through the random walk strategy, and improves the found paths through the adaptive repair method. The integrated application of these strategies allows $ACOSRA_R$ to show excellent performance in all cases.

5.2.2. The comparison to different algorithms

In UAV path planning problems, the adaptability of algorithms across multiple scenarios and dimensions is a key research focus. We compared our proposed algorithm $ACOSRA_R$ with PSO [7], ACO_R [40], $ACOPA_R$ [41], SPSO [9], EEFO [45], HLOA [46], CCO-AVOA [11] across 9 different cases in 3 different dimensions: 10D, 20D, and 30D. Table 4 presents the tuning process and results. Tables 5 and 6 display the parameters and results of each algorithm. The ↓ indicates that a smaller value represents better performance. Fig. 8 provides the convergence graphs for each algorithm under the 10D case to illustrate the convergence characteristics of the algorithms. Figs. 9 and 10 shows the optimal paths produced by each algorithm in the 10D scenario. The parameters for other algorithms were referenced from relevant literature. In our study, four parameters were used across all methods. Two of these parameters were directly adopted from previous studies as they have been extensively validated in similar contexts. For the two parameters specific to our proposed algorithm, we conducted extensive tuning experiments on case 1 with 10D, running 30 times, as it represents a realistic and commonly encountered scenario.

Before comparing the results of the algorithms, we found in our experiments that the irregular distribution of obstacles poses a challenge to finding feasible paths. This also resulted in difficulties for all algorithms except $ACOSRA_R$ to find feasible solutions in the search space, as $ACOSRA_R$ could utilise adaptive waypoints-repair method to find feasible initial solutions. To ensure a fair comparison of the experimental results, we applied the adaptive waypoints-repair method to feasible initial solutions in PSO, ACO_R, $ACOPA_R$, SPSO, EEFO, HLOA, and CCO-AVOA before optimisation.

From Table 6, it can be observed that in the 10D scenario, PSO, ACO_R, $ACOPA_R$, SPSO, and CCO-AVOA exhibit similar performance levels with slight fluctuations in cost across different cases. Although $ACOSRA_R$ shows a higher path cost than PSO in case 9, it demonstrates optimal or comparable performance in other cases. It is significantly better than the other algorithms, especially in cases 1, 2, 4, 5, 6, 7, and 8. EEFO and HLOA exhibit higher path costs than other algorithms, indicating that EEFO and HLOA are not as efficient as the other algorithms in solving low-dimensional problems. Case 9 demonstrates that PSO has its advantage in certain specific cases at low dimensions. However, overall, the $ACOSRA_R$ demonstrates excellent performance in the 10D path planning problem, consistently finding high-quality path solutions across eight cases. This reflects its adaptability and robustness when facing low-dimensional problems.

We can also observe that the increase in dimensionality has hurt the cost. This is because the number of optimised parameters doubled from 30 to 60, doubling the algorithm's search space, which poses a challenge for the algorithm. For instance, contrary to its performance in 10D, PSO performs the worst or second worst in 9 cases in 20D. In contrast to PSO, EEFO performs equally or second best in 9 cases, indicating that EEFO is more applicable to high-dimensional problems than PSO. Although the cost of $ACOSRA_R$ increases with the increase in dimensionality, compared to other algorithms in 20D, it demonstrates significant performance advantages and maintains high stability in most cases.

As the dimensionality of the problem increases to 30D, with 90 optimised parameters, it can be observed that other algorithms exhibit significant inadequacies in solving 30D problems from Table 6. For example, PSO, ACO_R, $ACOPA_R$, SPSO, EEFO, HLOA, and

Table 6
Comparison of algorithm results in terms of total cost (↓) in addressing different problems.

10D	PSO [7]	ACO_R [40]	$ACOPA_R$ [41]	SPSO [9]	EEFO [45]	HLOA [46]	CCO-AVOA [11]	$ACOSRA_R$
1	5712 ± 552*	5908 ± 295*	6332 ± 300*	5582 ± 437*	6186 ± 216*	8293 ± 495*	6590 ± 490*	**5322 ± 389**
2	6131 ± 750*	5227 ± 468*	5670 ± 410*	5372 ± 448*	5251 ± 271*	7263 ± 989*	6226 ± 445*	**4967 ± 206**
3	5534 ± 776	5665 ± 229*	5832 ± 45*	5516 ± 384*	5812 ± 201*	7061 ± 956*	5817 ± 540*	**5135 ± 393**
4	5265 ± 210*	5361 ± 139*	4794 ± 182*	5230 ± 191*	4835 ± 29*	4878 ± 37*	4802 ± 136*	**4792 ± 102**
5	5925 ± 48*	5698 ± 82*	5539 ± 370*	5090 ± 555*	5013 ± 358*	7017 ± 958*	5692 ± 448*	**4796 ± 381**
6	5519 ± 961*	6395 ± 503*	6272 ± 553*	5731 ± 637*	5358 ± 489*	7219 ± 997*	5811 ± 396*	**4877 ± 169**
7	5267 ± 401*	5281 ± 444*	4890 ± 68*	5167 ± 444*	5537 ± 332*	6444 ± 643*	5666 ± 663*	**4714 ± 225**
8	7579 ± 1218*	6381 ± 618*	5863 ± 396*	6060 ± 732*	6482 ± 766*	7453 ± 489*	6740 ± 255*	**4635 ± 102**
9	**5405 ± 530**	9466 ± 119*	9672 ± 540*	8606 ± 396*	8197 ± 1124*	11 450 ± 2045*	9058 ± 1335*	6625 ± 1416*
20D	PSO [7]	ACO_R [40]	$ACOPA_R$ [41]	SPSO [9]	EEFO [45]	HLOA [46]	CCO-AVOA [11]	$ACOSRA_R$
1	8915 ± 1032*	6672 ± 306*	8160 ± 258*	7304 ± 957*	7098 ± 376*	12 459 ± 1879*	7947 ± 718*	**6045 ± 642**
2	11 518 ± 2685*	6602 ± 334*	8035 ± 453*	7543 ± 984*	6487 ± 593*	10 377 ± 2583*	8572 ± 653*	**5904 ± 532**
3	9308 ± 1648*	6495 ± 144*	7293 ± 388*	6304 ± 440*	6076 ± 107	7889 ± 1429*	6588 ± 409*	**5959 ± 400**
4	6600 ± 962*	5826 ± 116*	**4848 ± 17**	5810 ± 594*	4849 ± 16	4901 ± 50*	4990 ± 51*	5236 ± 249*
5	7846 ± 707*	6305 ± 205*	6805 ± 131*	6382 ± 430*	5535 ± 609	11 920 ± 3785*	6807 ± 1365*	**5271 ± 331**
6	9003 ± 1223*	7611 ± 385*	7812 ± 755*	7612 ± 1034*	7262 ± 785*	11 573 ± 3137*	6734 ± 721*	**6489 ± 919**
7	7264 ± 854*	6680 ± 327*	6036 ± 422*	6053 ± 373*	6052 ± 170*	7471 ± 1027*	6365 ± 1214*	**5135 ± 410**
8	7308 ± 1578*	8571 ± 434*	8924 ± 900*	7265 ± 525*	6972 ± 132*	8748 ± 1004*	8091 ± 1510*	**5075 ± 553**
9	13 339 ± 2771*	11 242 ± 824*	11 445 ± 392*	9447 ± 416*	9640 ± 362*	14 086 ± 3335*	10 979 ± 771*	**8425 ± 1145**
30D	PSO [7]	ACO_R [40]	$ACOPA_R$ [41]	SPSO [9]	EEFO [45]	HLOA [46]	CCO-AVOA [11]	$ACOSRA_R$
1	8716 ± 1243*	7940 ± 429*	9526 ± 215*	8784 ± 795*	9043 ± 735*	15 253 ± 2828*	9439 ± 989*	**7033 ± 649**
2	16 758 ± 2815*	9139 ± 1010*	10 346 ± 706*	9776 ± 1146*	7956 ± 893*	14 186 ± 480*	9702 ± 1523*	**6469 ± 619**
3	16 577 ± 2716*	8075 ± 347*	11 070 ± 799*	7290 ± 905*	7177 ± 119*	8581 ± 1622*	7987 ± 1231*	**6705 ± 405**
4	12 763 ± 2233*	6891 ± 260*	5278 ± 29	7034 ± 988*	5299 ± 76	5354 ± 102*	5320 ± 172*	**5272 ± 221**
5	20 994 ± 1859*	7766 ± 780*	8606 ± 639*	7794 ± 1037*	6735 ± 736*	13 543 ± 1832*	8871 ± 1687*	**5703 ± 401**
6	15 187 ± 2782*	9592 ± 627*	10 002 ± 856*	9964 ± 1615*	8260 ± 871	14 349 ± 2296*	11 095 ± 1753*	**7894 ± 725**
7	9396 ± 1238*	8766 ± 416*	7710 ± 733*	6908 ± 581*	6566 ± 244*	8883 ± 1514*	7058 ± 1077*	**6063 ± 312**
8	13 433 ± 575*	10 781 ± 666*	10 933 ± 847*	8514 ± 445*	7432 ± 224*	9772 ± 1360*	9514 ± 875*	**6986 ± 702**
9	25 127 ± 2935*	15 269 ± 1776*	13 870 ± 497*	10 522 ± 754*	10 271 ± 428*	23 245 ± 1875*	13 543 ± 1439*	**9517 ± 477**

CCO-AVOA all have costs exceeding 10 000 in different cases. In contrast, the performance of the $ACOSRA_R$ algorithm surpasses other algorithms in 9 cases.

From the convergence curves in Fig. 8, it is observable that $ACOSRA_R$ typically exhibits rapid convergence in the early stage, identifying a feasible solution before continuing to evolve in search of superior solutions rather than becoming trapped in local optima. In some situations, such as case 4, $ACOPA_R$ initially converges more swiftly than $ACOSRA_R$ in the early stage. However, as $ACOPA_R$ progresses and encounters challenges, it ultimately discovers an improved solution. This further substantiates the superiority of $ACOSRA_R$ in addressing problems of UAV path planning.

In Figs. 9 and 10, purple cylinders represent obstacles, which are laid out on the terrain, with the colour of the terrain directly related to the altitude—higher altitude areas are depicted as green, while lower altitude areas appear red. The black square and circle denote the starting and ending points of the drones, respectively. The UAV path from the starting point to the target point is shown as a curve with differently coloured waypoints, representing key points of the route and indicating changes in flight altitude and direction. From Figs. 9 and 10, we can see that the UAV can avoid obstacles not only in the horizontal direction but also vertically—when the height of an obstacle is less than the flight altitude of the UAV, the UAV can directly fly over the obstacle to seek an optimal path. Additionally, to ensure that the lines connecting each waypoint avoid obstacles, we have established an intersection function in our collision cost calculations, as mentioned in Section 3.

From the above analysis, it can be seen that $ACOSRA_R$ outperforms the other seven algorithms in most cases. This can be attributed to key features in its design. Firstly, the improved state transition probability helps it quickly locate better regions in the vast search space. Secondly, the crossover flight strategy of $ACOSRA_R$ enables the algorithm to escape local optima. Lastly, the adaptive waypoints-repair method makes $ACOSRA_R$ handle obstacles effectively, further distinguishing it from the other algorithms. When encountering obstacles, it can quickly find alternative feasible paths. This capability not only enhances the algorithm's robustness but also ensures that it can maintain high performance even in dynamic environments.

5.2.3. Computational time

To compare the complexity of the algorithms, we tested the computational time required for each algorithm to solve specific problems. Specifically, we recorded the total time taken by each algorithm in the 10D, 20D, and 30D problems for all cases, with 10,000 fitness evaluations, and conducted 30 experiments to obtain the average value. The results are presented in Table 7, where a smaller value (↓) indicates better performance. From Table 7, it can be observed that: first, as the dimension increases, the execution time of all algorithms increases slightly. This is because higher dimensions result in a more complex search space, requiring more computational resources to explore the larger solution space. Second, the execution time of $ACOSRA_R$ is higher than that of PSO,

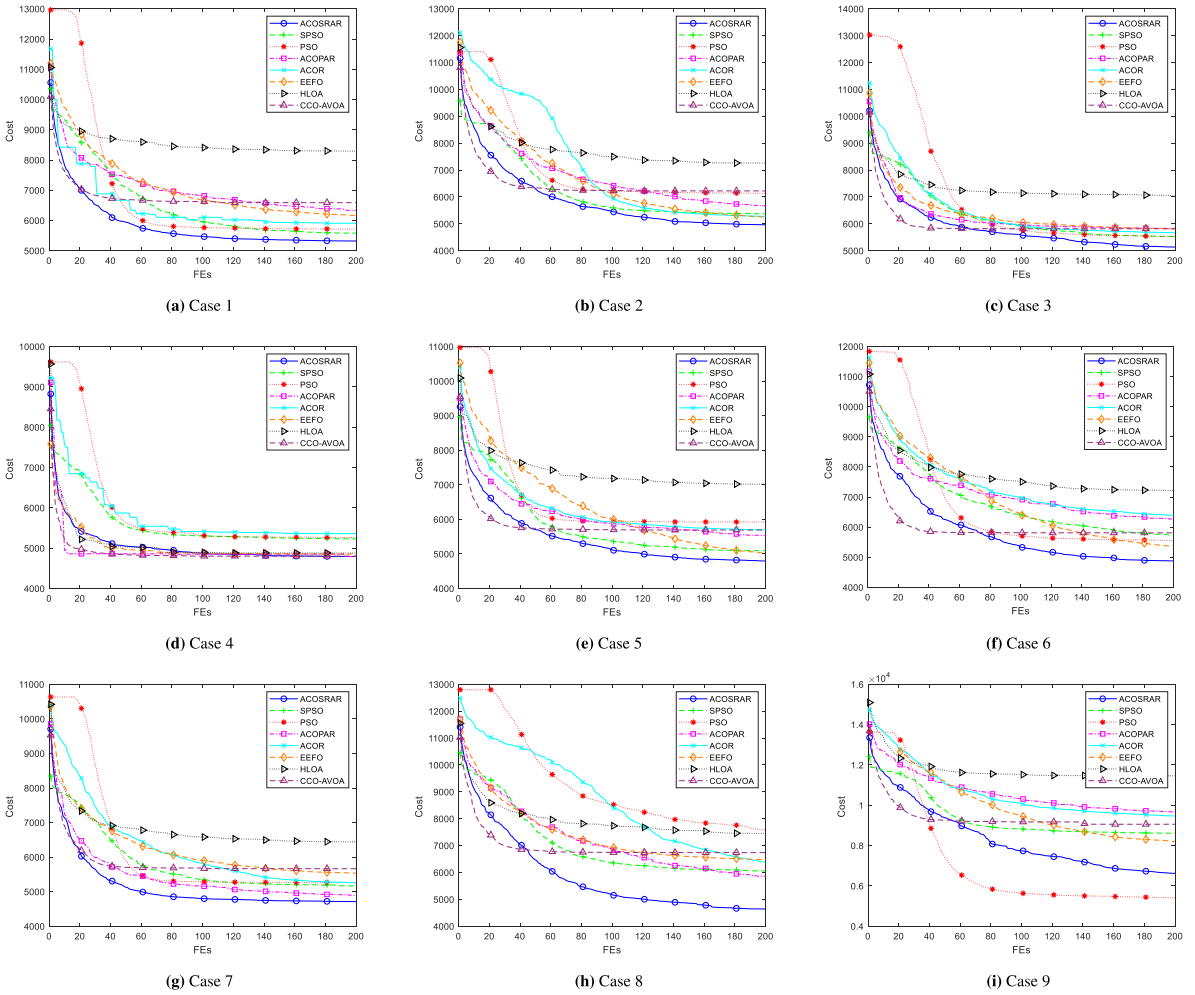
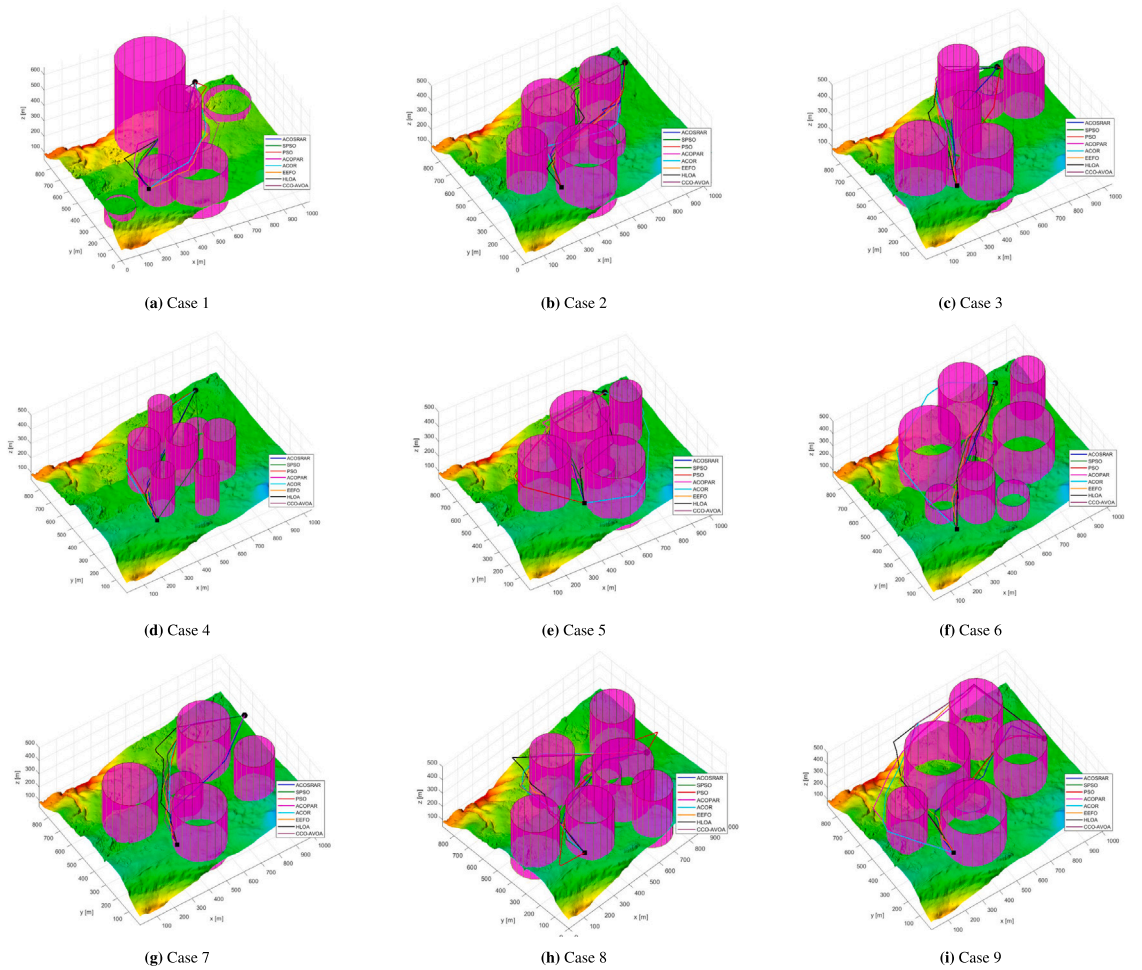


Fig. 8. The convergence graphs of comparison algorithms for 10D cases.

SPSO, EEFO, HLOA, and CCO-AVOA, as $ACOSRA_R$ typically requires more complex pheromone update rules when dealing with multi-dimensional problems. Third, the execution times of ACO_R and $ACOPA_R$ are close to that of $ACOSRA_R$, indicating that the integration of our proposed strategies did not increase the computational burden of the original ACO_R. Based on the above analysis, the slight difference in computational complexity does not outweigh the clear advantages of $ACOSRA_R$ in path planning problems, especially in finding feasible paths with lower costs, where $ACOSRA_R$ demonstrates strong competitiveness. Given this significant performance improvement, the slight increase in computational cost is reasonable and acceptable. The additional computational resources typically lead to better results in complex environments, making $ACOSRA_R$ a more effective choice in path planning, particularly in reducing path costs and improving search accuracy.

5.2.4. The validation of $ACOSRA_R$ with DWA in the dynamic environment

$ACOSRA_R$ has already planned a global path in a known environment, and then DWA (Dynamic Window Approach) is introduced to evade moving obstacles and re-plan the path. To test the obstacle avoidance capability of DWA, we conducted a simulation experiment on case 3. The information of moving obstacles is shown in Table 8. The default length unit in the environment is the meter, and the time unit is the second. Fig. 11 shows the generated path at five different moments (Ta, Tb, Tc, Td and Te) during local path planning after the completion of global path planning and top view of the end. The global path planned by $ACOSRA_R$ is presented in the form of a black dashed line, while the path planned by $ACOSRA_R$ combined with DWA is shown as a solid blue line with obstacle 1 moving horizontally and obstacle 2 moving vertically. In the experiment, we set the maximum linear velocity of the UAV to be 10 m/s and the maximum acceleration to be 5 m/s^2.



(a) Case 1 (b) Case 2 (c) Case 3

(d) Case 4 (e) Case 5 (f) Case 6

(g) Case 7 (h) Case 8 (i) Case 9

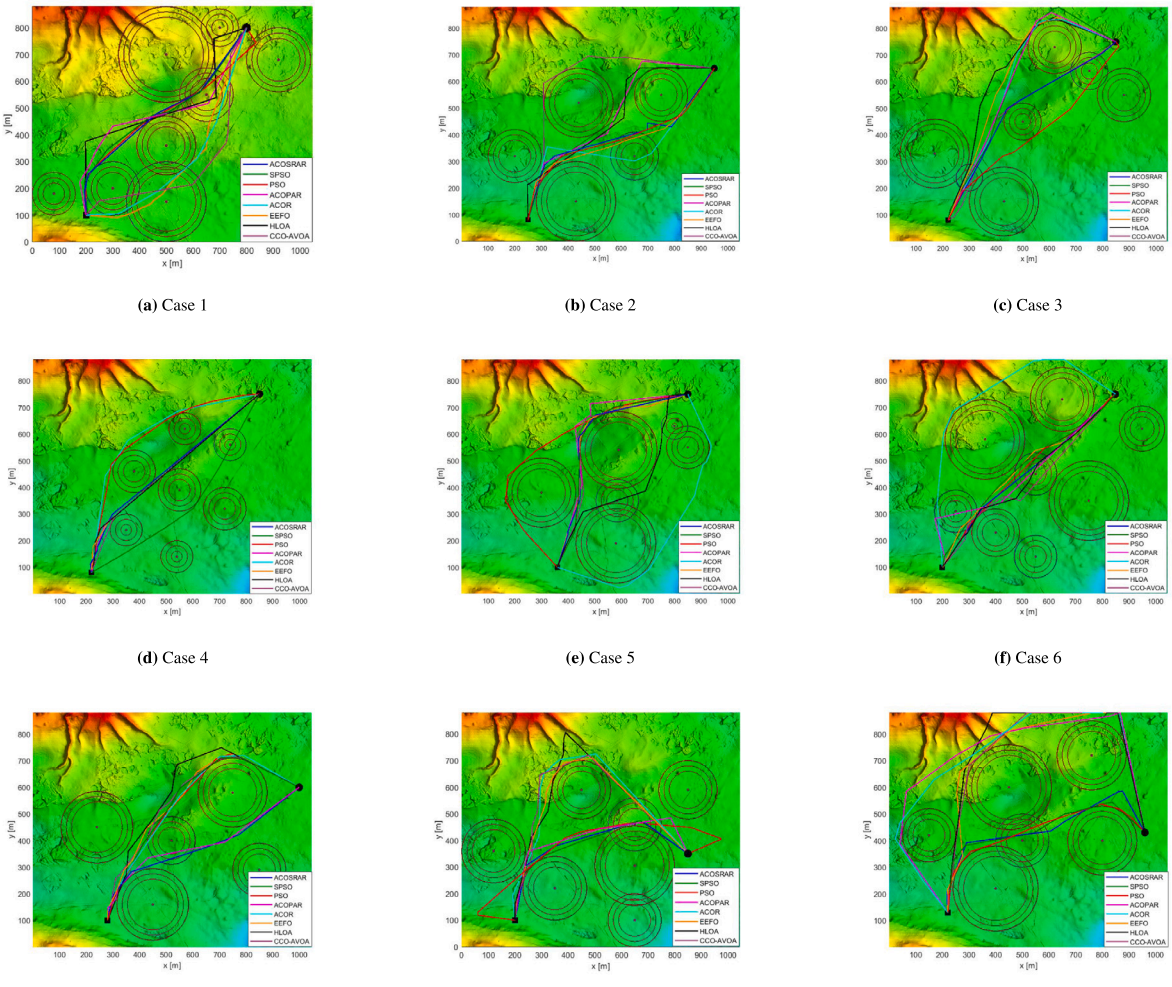
Fig. 9. The best generated 3D view paths in 10D cases by different algorithms.

From Fig. 11, it can be observed that at Ta, as the UAV approaches the moving obstacle 1, the DWA effectively guides the UAV to turn towards the direction of the obstacle's movement while maintaining a safe distance to avoid collision. At Tb, after successfully avoiding the first moving obstacle, the UAV can quickly reconnect to the planned initially global path and continue along it, minimising the cost of heading and time. By Tc, the vertically moving obstacle 2 enters the UAV's safety range, prompting it to adjust its trajectory to avoid the obstacle. At Td, the UAV successfully manoeuvres around the second obstacle and swiftly approaches the target point, ultimately reaching it by Te.

Through observation of Fig. 11, we note that although the UAV temporarily deviates from the global path during the dynamic obstacle avoidance, it can swiftly return to the original route after the obstacles are cleared, ultimately reaching the destination successfully. This demonstrates DWA's excellent performance in avoiding dynamic obstacles and reflects the superior effectiveness of combining DWA with $ACOSRA_R$ for UAV path planning in scenarios involving both static and moving obstacles.

6. Conclusion

In this paper, we propose a novel path planning algorithm named $ACOSRA_R$, aimed at addressing complex path planning problems. It employs improved state transition probability, random walk strategy and adaptive waypoints-repair method to improve the efficiency and accuracy of path planning. Firstly, it enhances the efficiency of the search process through the improved state transition probability, which facilitates rapid convergence. Additionally, it incorporates the random walk strategy to generate steps with different distributions. This feature prevents the algorithm from being trapped in local optima and increases the likelihood of discovering novel solutions. Lastly, the adaptive waypoints-repair method is used to correct and refine the paths found.



(a) Case 1 **(b)** Case 2 **(c)** Case 3

(d) Case 4 **(e)** Case 5 **(f)** Case 6

(g) Case 7 **(h)** Case 8 **(i)** Case 9

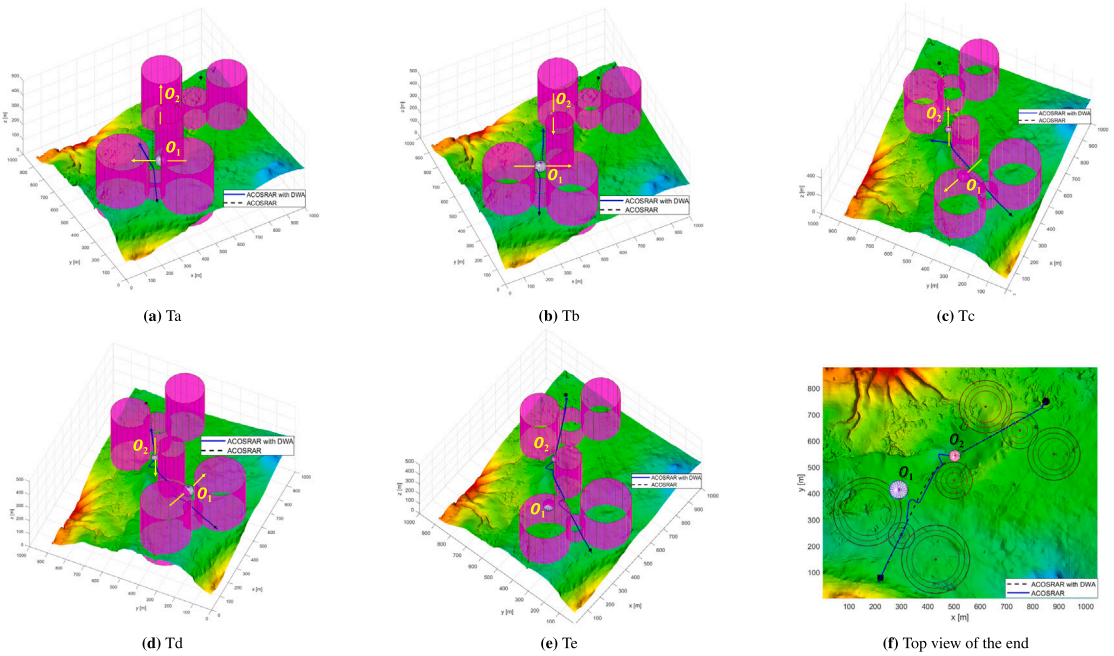
Fig. 10. The best generated top view paths in 10D cases by different algorithms.

The effectiveness of $ACOSRA_R$ is tested on 9 different real DEM map scenarios. Compared to most existing research on UAV path planning in 3D environments, our work enables the UAV to navigate horizontally to avoid obstacles and facilitates vertical traversal over obstacles, significantly enhancing flight quality. Finally, we successfully integrate DWA with $ACOSRA_R$ to jointly solve path planning problems for UAV with static and moving obstacles. Experimental results demonstrate that our proposed algorithm $ACOSRA_R$ can obtain high-quality solutions in various environments, effectively addressing the 3D path planning problem of UAV in complex environments.

Looking ahead to future research directions, while our current algorithm has shown promise in providing effective path planning solutions for single UAV in dynamic 3D environments, certain limitations need to be addressed. One major limitation is our assumption that moving obstacles maintain a constant speed and trajectory. In reality, obstacles may exhibit unpredictable behaviours, such as varying speeds and sudden direction changes, which can challenge the algorithm's adaptability. Therefore, a key focus of future research will be the development of algorithms capable of robustly handling moving obstacles with random speeds and frequent changes. Moreover, we aim to extend our work to support multi-UAV systems, which present additional complexities in coordination and collision avoidance. This will entail designing more sophisticated coordination mechanisms and collision avoidance strategies to ensure the safe and efficient operation of multiple UAVs in shared airspace environments. Additionally, we see great potential in integrating our algorithm with other advanced technologies, such as machine learning. By leveraging machine learning techniques, we can enhance the accuracy and efficiency of path planning by enabling the algorithm to learn from past experiences and adapt to evolving environmental conditions.

Table 7
Computational time (↓) for different algorithms.

10D	PSO [7]	ACO_R [40]	$ACOPA_R$ [41]	SPSO [9]	EEFO [45]	HLOA [46]	CCO-AVOA [11]	$ACOSRA_R$
1	7.46	12.51	13.34	8.26	7.91	8.21	8.47	12.93
2	7.67	13.54	13.61	7.78	6.02	7.25	8.30	13.25
3	8.30	13.58	14.28	8.13	6.99	6.69	8.01	12.39
4	8.13	12.62	13.97	7.53	7.45	6.82	7.96	13.60
5	7.56	12.29	12.75	7.47	7.20	7.66	9.04	13.99
6	8.73	13.91	14.13	8.50	6.93	6.75	8.65	13.42
7	7.04	14.01	13.85	8.95	7.09	6.97	8.92	13.46
8	8.82	13.33	14.66	8.41	6.83	7.43	7.81	14.37
9	7.80	13.72	14.54	7.52	6.34	6.76	8.43	13.79
20D	PSO [7]	ACO_R [40]	$ACOPA_R$ [41]	SPSO [9]	EEFO [45]	HLOA [46]	CCO-AVOA [11]	$ACOSRA_R$
1	9.85	14.93	15.02	10.46	10.67	9.36	10.61	15.97
2	10.78	15.01	15.88	10.98	9.33	10.07	11.51	14.00
3	10.42	15.68	14.52	10.83	9.81	9.72	11.20	15.52
4	10.11	14.19	15.13	11.32	9.27	10.13	11.25	14.66
5	10.35	15.83	15.24	10.57	9.21	10.32	10.58	15.59
6	11.92	14.17	15.08	10.99	10.09	9.78	10.42	14.68
7	10.81	14.85	15.09	11.21	9.96	9.10	11.73	14.25
8	9.96	15.40	14.21	10.77	10.18	9.87	10.61	15.34
9	10.58	15.29	15.30	9.86	10.32	9.60	10.94	14.95
30D	PSO [7]	ACO_R [40]	$ACOPA_R$ [41]	SPSO [9]	EEFO [45]	HLOA [46]	CCO-AVOA [11]	$ACOSRA_R$
1	13.48	17.33	18.59	12.94	12.77	12.24	14.17	17.59
2	12.57	17.19	18.03	13.03	12.93	11.37	14.39	18.42
3	12.18	18.30	17.10	12.61	12.68	12.64	13.92	18.68
4	12.22	17.25	18.17	13.19	12.49	12.16	15.42	17.79
5	11.06	16.98	18.51	13.63	11.44	13.52	13.98	19.12
6	13.46	18.24	17.14	13.31	13.12	12.18	15.85	18.45
7	10.98	17.11	17.67	12.07	12.74	12.44	14.41	16.79
8	13.91	19.61	18.37	13.04	12.71	12.89	14.53	18.68
9	11.78	17.22	18.95	13.96	12.45	12.03	15.22	17.60



(a) Ta **(b)** Tb **(c)** Tc

(d) Td **(e)** Te **(f)** Top view of the end

Fig. 11. The paths generated in 10D case 3 by $ACOSRA_R$ with DWA.

Table 8
Moving obstacles settings.

Moving obstacles	1	2
Details	O1 (400,300,350) OR1 = 30 OV1 = (−6,6,0) OT1 = 70	O2 (510,540,375) OR2 = 20 OV2 = (0;0;6) OT2 = 50

CRediT authorship contribution statement

Ben Niu: Conceptualization, Data analysis, Writing – original draft, Writing – review & editing, Supervision, Funding Acquisition. **Yongjin Wang:** Conceptualization, Methodology, Data curation, Visualization, Writing – original draft. **Jing Liu:** Conceptualization, Methodology, Supervision, Writing – review & editing. **Gabriel Xiao-Guang Yue:** Conceptualization, Validation, Writing – review & editing.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the authors used ChatGPT in order to improve the language and readability. After using the tool, the authors reviewed the content as needed and take full responsibility for the content of the publication.

Funding information

The study is supported by Key Program of National Natural Science Foundation of China (Nos. 72334004), General Program of National Natural Science Foundation of China (Nos. 71971143), Guangdong Provincial Philosophy and Social Sciences Planning Project, China (No. GD22CGL35), Special Projects in Key Fields of Ordinary Colleges and Universities in Guangdong Province, China (No. 2022ZDZX2054), and University Innovation Team Project of Guangdong Province, China (No. 2021WCXTD002).

Declaration of competing interest

No potential conflict of interest was reported by the authors.

Data availability

Data will be made available on request.

References

[1] Zafar MN, Mohanta J. Methodology for path planning and optimization of mobile robots: A review. Procedia Comput Sci 2018;133:141–52.
[2] Hart PE, Nilsson NJ, Raphael B. A formal basis for the heuristic determination of minimum cost paths. IEEE Trans Syst Sci Cybern 1968;4(2):100–7.
[3] Kavraki LE, Svestka P, Latombe J-C, Overmars MH. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans Robot Autom 1996;12(4):566–80.
[4] Hwang YK, Ahuja N, et al. A potential field approach to path planning. IEEE Trans Robot Autom 1992;8(1):23–32.
[5] Roberge V, Tarbouchi M, Labonté G. Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning. IEEE Trans Ind Inform 2012;9(1):132–41.
[6] Paden B, Čáp M, Yong SZ, Yershov D, Frazzoli E. A survey of motion planning and control techniques for self-driving urban vehicles. IEEE Trans Intell Veh 2016;1(1):33–55.
[7] Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of ICNN'95-international conference on neural networks. Vol. 4, IEEE; 1995, p. 1942–8.
[8] Dorigo M, Stützle T. The ant colony optimization metaheuristic: Algorithms, applications, and advances. In: Handbook of metaheuristics. Springer; 2003, p. 250–85.
[9] Phung MD, Ha QP. Safety-enhanced UAV path planning with spherical vector-based particle swarm optimization. Appl Soft Comput 2021;107:107376.
[10] Jiaqi S, Li T, Hongtao Z, Xiaofeng L, Tianying X. Adaptive multi-UAV path planning method based on improved gray wolf algorithm. Comput Electr Eng 2022;104:108377.
[11] Ait-Saadi A, Meraihi Y, Soukane A, Yahia S, Ramdane-Cherif A, Gabis AB. An enhanced african vulture optimization algorithm for solving the unmanned aerial vehicles path planning problem. Comput Electr Eng 2023;110:108802.
[12] Tan L, Zhang H, Shi J, Liu Y, Yuan T. A robust multiple unmanned aerial vehicles 3D path planning strategy via improved particle swarm optimization. Comput Electr Eng 2023;111:108947.
[13] Chen Z, Wu H, Chen Y, Cheng L, Zhang B. Patrol robot path planning in nuclear power plant using an interval multi-objective particle swarm optimization algorithm. Appl Soft Comput 2022;116:108192.
[14] Mohanta JC, Parhi DR, Patel SK. Path planning strategy for autonomous mobile robot navigation using Petri-GA optimisation. Comput Electr Eng 2011;37(6):1058–70.
[15] Shen J, Fu X, Wang H, Shen S. Fast path planning for underwater robots by combining goal-biased Gaussian sampling with focused optimal search. Comput Electr Eng 2021;95:107412.
[16] Lin S, Liu A, Wang J, Kong X. An intelligence-based hybrid PSO-SA for mobile robot path planning in warehouse. J Comput Sci 2023;67:101938.

[17] Duan H, Yu Y, Zhang X, Shao S. Three-dimension path planning for UCAV using hybrid meta-heuristic ACO-DE algorithm. Simul Model Pract Theory 2010;18(8):1104–15.
[18] Cui J, Wu L, Huang X, Xu D, Liu C, Xiao W. Multi-strategy adaptable ant colony optimization algorithm and its application in robot path planning. Knowl-Based Syst 2024;288:111459.
[19] Meng W, Gong Y, Xu F, Tao P, Bo P, Xin S. Efficient path planning for AUVs in unmapped marine environments using a hybrid local–global strategy. Ocean Eng 2023;288:116227.
[20] Zhou Q, Lian Y, Wu J, Zhu M, Wang H, Cao J. An optimized Q-learning algorithm for mobile robot local path planning. Knowl-Based Syst 2024;286:111400.
[21] Yang X, Han Q. Improved reinforcement learning for collision-free local path planning of dynamic obstacle. Ocean Eng 2023;283:115040.
[22] Fox D, Burgard W, Thrun S. The dynamic window approach to collision avoidance. IEEE Robot Autom Mag 1997;4(1):23–33.
[23] Dijkstra E. A note on two problems in connexion with graphs. Numer Math 1959;1:269–71.
[24] Foead D, Ghifari A, Kusuma MB, Hanafiah N, Gunawan E. A systematic literature review of A* pathfinding. Procedia Comput Sci 2021;179:507–14.
[25] Pettersson PO, Doherty P. Probabilistic roadmap based path planning for an autonomous unmanned aerial vehicle. In: Proc. of the ICAPS-04 workshop on connecting planning theory with practice. 2004.
[26] Noreen I, Khan A, Habib Z. Optimal path planning using RRT* based approaches: a survey and future directions. Int J Adv Comput Sci Appl 2016;7(11).
[27] Barraquand J, Langlois B, Latombe J-C. Numerical potential field techniques for robot path planning. IEEE Trans Syst Man Cybern 1992;22(2):224–41.
[28] Fan X, Guo Y, Liu H, Wei B, Lyu W. Improved artificial potential field method applied for AUV path planning. Math Probl Eng 2020;2020:1–21.
[29] Cheng C, Sha Q, He B, Li G. Path planning and obstacle avoidance for AUV: A review. Ocean Eng 2021;235:109355.
[30] Bialas J, Doller M. Coverage path planning for unmanned aerial vehicles in complex 3D environments with deep reinforcement learning. In: 2022 IEEE international conference on robotics and biomimetics. ROBIO, 2022, p. 1080–5.
[31] Sanna G, Godio S, Guglieri G. Neural network based algorithm for multi-UAV coverage path planning. In: 2021 international conference on unmanned aircraft systems. ICUAS, 2021, p. 1210–7.
[32] Bayerlein H, Theile M, Caccamo M, Gesbert D. Multi-UAV path planning for wireless data harvesting with deep reinforcement learning. IEEE Open J Commun Soc 2021;2:1171–87.
[33] Quinlan S, Khatib O. Elastic bands: Connecting path planning and control. In: [1993] proceedings IEEE international conference on robotics and automation. IEEE; 1993, p. 802–7.
[34] Kurniawati H, Fraichard T. From path to trajectory deformation. In: 2007 IEEE/RSJ international conference on intelligent robots and systems. IEEE; 2007, p. 159–64.
[35] Raffo GV, Gomes GK, Normey-Rico JE, Kelber CR, Becker LB. A predictive controller for autonomous vehicle path tracking. IEEE Trans Intell Transp Syst 2009;10(1):92–102.
[36] Ait-Saadi A, Meraihi Y, Soukane A, Ramdane-Cherif A, Benmessaoud Gabis A. A novel hybrid chaotic aquila optimization algorithm with simulated annealing for unmanned aerial vehicles path planning. Comput Electr Eng 2022;104:108461.
[37] Yu X, Chen W-N, Gu T, Yuan H, Zhang H, Zhang J. ACO-A*: Ant colony optimization plus A* for 3-D traveling in environments with dense obstacles. IEEE Trans Evol Comput 2018;23(4):617–31.
[38] Ajeil FH, Ibraheem IK, Azar AT, Humaidi AJ. Grid-based mobile robot path planning using aging-based ant colony optimization algorithm in static and dynamic environments. Sensors 2020;20(7):1880.
[39] Song Q, Zhao Q, Wang S, Liu Q, Chen X. Dynamic path planning for unmanned vehicles based on fuzzy logic and improved ant colony optimization. IEEE Access 2020;8:62107–15.
[40] Socha K, Dorigo M. Ant colony optimization for continuous domains. European J Oper Res 2008;185(3):1155–73.
[41] Liu J, Anavatti S, Garratt M, Abbass HA. Modified continuous ant colony optimisation for multiple unmanned ground vehicle path planning. Expert Syst Appl 2022;196:116605.
[42] Sharma VP, Choudhary HR, Kumar S, Choudhary V. A modified DE: Population or generation based levy flight differential evolution (PGLFDE). In: 2015 international conference on futuristic trends on computational analysis and knowledge management. ABLAZE, IEEE; 2015, p. 704–10.
[43] Gao Y, Zhang H, Duan Y, Zhang H. A novel hybrid PSO based on levy flight and wavelet mutation for global optimization. PLoS ONE 2023;18(1):e0279572.
[44] Liu J, Anavatti S, Garratt M, Abbass HA. Multi-operator continuous ant colony optimisation for real world problems. Swarm Evol Comput 2022;69:100984.
[45] Zhao W, Wang L, Zhang Z, Fan H, Zhang J, Mirjalili S, Khodadadi N, Cao Q. Electric eel foraging optimization: A new bio-inspired optimizer for engineering applications. Expert Syst Appl 2024;238:122200.
[46] Peraza-Vázquez H, Peña-Delgado A, Merino-Treviño M, Morales-Cepeda AB, Sinha N. A novel metaheuristic inspired by horned lizard defense tactics. Artif Intell Rev 2024;57(3):59.

Ben Niu He is the Professor and Head of Management Department at Shenzhen University, China. He is a Seasoned Researcher with a good number of research publications. His research appeared in well reputed journals including IEEE Transactions on Evolution Computation, IEEE Transactions on System Man and Cybernetics, Journal of Cleaner Production, and International Journal of Information Management.

Yongjin Wang He is a master student at School of Management, Shenzhen University, China. His research focuses on the application of swarm intelligence optimisation algorithms in the areas of UAV path planning and intelligent transportation systems.

Jing Liu She is a Postdoctoral Researcher at School of Systems and Computing, University of New South Wales, Canberra, Australia. Her research interests include evolutionary computation, swarm optimisation, multi-UGV mission planning, path planning, etc.

Gabriel Xiao-Guang Yue He is currently a Research Fellow of IIEP. He is an IETI Fellow, RSA Fellow and Pakistan Academy of Engineering Fellow. Since 2022, he has been ranked as the Stanford World Top 2 Scientist. He is also the Executive Secretary General of South-South Biodiversity Science Project of China Biodiversity Conservation and Green Development Foundation. His research, spanning intelligent information processing, computational social science, and sustainability, has resulted in numerous publications in refereed International journals, proceedings and book chapters.