Received: Added at production | Revised: Added at production | Accepted: Added at production

ORIGINAL RESEARCH

A game-theoretic cooperative path planning strategy using hybrid heuristic optimization algorithm

Yutong Zhu[1] | **Ye Zhang**[1,2]

[1]School of Astronautics, Northwestern Polytechnical University, Xi'an, China

[2]Research and Development Institute of Northwestern Polytechnical University in Shenzhen, Shen Zhen, China

Correspondence
Ye Zhang, School of Astronautics, Northwestern Polytechnical University, Xi'an, China.
Email: zhang_ye@nwpu.edu.cn

Abstract

A novel method based on game theory and LCD-SCA optimization algorithm is proposed for solving the cooperative path planning challenge for multiple UAVs in a desired formation configuration. The cooperative path planning problem is solved by identifying the optimal strategy for the Stackelberg-Nash game. The conventional sine-cosine algorithm method is enhanced by incorporating linear differential decrement, chaos theory, and differential evolution, and the proposed heuristic method is integrated into the path planning problem. An optimal strategy for finding the game by minimising the global cost function via the heuristic method is integrated. Extensive simulation and comparison results are provided to evaluate the performance through simulation, compared with our previous work on path planning.

KEYWORDS
Game theory, Sine cosine algorithm, Cooperative path planning, Stackelberg-Nash game, Optimal strategy

1 | INTRODUCTION

The development of autonomy has led to an increasing involvement of unmanned aerial vehicles (UAVs), especially quadrotors, in applications that cannot be easily accomplished by human beings. These include search and rescue, reconnaissance, resource exploration, forest fire prevention and military missions [1,2]. The key challenge of UAV formation control is to identify and address potential conflicts and interactions between members of the group, in order to ensure a consistent shape and maintain a desired trajectory when undertaking a robotic task. In this paper, the principles of game theory, a key area of mathematics that studies interactions between rational decision-makers [3], offer a valuable framework for identifying an optimal strategy.

In a game, players can choose to take action based on not only their own strategy but also on the strategies of others. Consequently, the optimal strategy is often determined by the player's expectations of others' actions. Games can be broadly categorised as either cooperative or non-cooperative [4]. In cooperative games, several players share a common goal of winning or achieving a profit that is better than that achievable by playing alone. A significant challenge in cooperative games is the trade-off between stability and efficiency of the overall system, as discussed in Ref. [5]. In contrast, players in non-cooperative games, who possess available information about their own intentions, payoff functions, and procedural details of the game, are able to pursue their own strategies. Although each player is aware of the decisions of others, they must simultaneously make their own decision in a symmetric competition. This may be to find optimal control parameters, as in the case of the game of control presented in Ref. [6], or to seek strategies for multiple clusters in a distributed way, as in the case of the game of clusters presented in Ref. [7]. In contrast, in the game of Stackelberg, players must adopt sequential steps depending on the moves of the leading players, as presented in Ref. [8]. Once the leaders have initiated their initial actions, the followers are then able to adjust their own strategies in accordance with the leaders' actions. This paper presents a comprehensive transformation of the problem of cooperative path planning for multiple UAVs. The transformation involves the minimisation of a multi-objective cost function and the solution of the problem using the Stackelberg-Nash game. Each UAV in the group is equipped with a self-executing

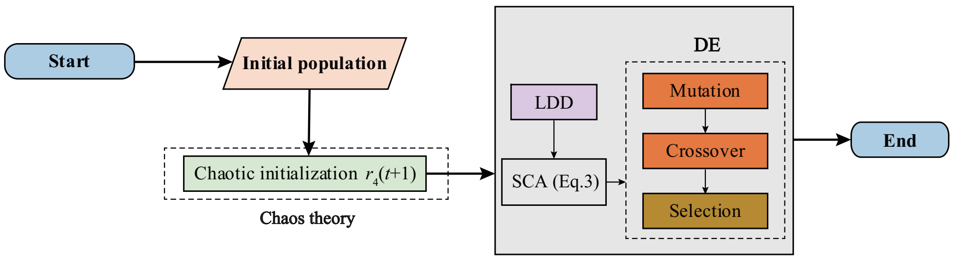


FIGURE 1 The whole process of LCD-SCA.

controller that enables it to maintain the desired geometric formation. The formation is maintained while path constraints on each UAV are considered through the cost function.

This paper focuses on metaheuristic-based approaches for solving trajectory planning problems in mobile robots. Recently, researchers have introduced novel methods by adapting or blending metaheuristic approaches, applying these techniques to single or multiple robot systems' trajectory planning. The Sine-Cosine Algorithm (SCA), developed by Mirjalili in 2016, is inspired by the triangular sine-cosine principle[9]. This algorithm takes a holistic approach, starting with a random solution and progressively refining it using sine and cosine functions toward an optimal solution[10,11]. Notably, it employs both random and adaptive variables to balance exploration and exploitation trade-offs.[12] proposed a hybrid method that combines improved cuckoo search, PSO, and SCA algorithms, applied to trajectory planning for a multi-robot system carrying sticks. The problem models involve multiple pairs of robots carrying sticks, emphasizing cooperation among robots. Another hybrid method introduced by[13] combines an enhanced GWO algorithm with the SCA algorithm. The improvement in the GWO algorithm involves integrating a democratic rule concept inspired by societal structures. Additionally,[14] presented a hybrid method merging the kidney-inspired algorithm with the SCA algorithm to hasten the convergence of the kidney-inspired algorithm (KA). However, due to the SCA's inherent structure and characteristics, it faces limitations in exploration, leading to near-optimal solutions. It encounters issues such as premature convergence and lower accuracy for certain optimization problems. In recent years, the Linear Differential Decrement strategy (LDD) has emerged as a mathematical tool utilized for exploring new algorithms[15]. LDD facilitates parameter selection within SCA, allowing for curvature adjustments during initial and terminal phases, thereby inducing significant alterations in future update positions. Chaos, characterized as a dynamic and deterministic system, exhibits extreme sensitivity to initial conditions and parameters. While inherently random and unpredictable, chaos also reveals patterns or regularities. Studies have demonstrated the applicability of chaos theory in assisting meta-heuristics like PSO in determining optimal global coverage values[16]. Meanwhile, the Differential Evolution (DE) method leverages both global and local search strategies[17,18]. Consequently, each search agent's final position is determined by the amalgamation of outcomes from both local and global search processes. To overcome these limitations, this paper proposes an improved SCA, called **L**inear differential decrement, **C**haos and **D**ifferential evolution **SCA** (LCD-SCA). It is notable that our method taken considers the synergistic constraints on the optimal paths of each UAV in a formation to cover all requirements for formation, flexibility and safety. Unlike other methods to formation control, the paths generated in our method are self-adaptive through the intervention of the proposed optimization algorithm, which enables the entire group to reconfigure itself in order to better adapt to changes in the complex environment.

A flow chart in Fig. 1 shows the complete implementation process of the proposed algorithm. The contributions and innovations of our work are as follows:

(i) We transform the problem of cooperative path planning for multiple UAVs into minimising a multi-objective cost function in a Stackelberg-Nash game.

(ii) We improve the SCA optimization algorithm by introducing linear differential decrement, chaos theory and differential evolution. Compared with the original SCA, LCD-SCA has a stronger ability to find the optimum.

(iii) The proposed LCD-SCA is employed to identify the optimal strategy of the Stackelberg-Nash game, with the introduction of obstacle avoidance potential function. The efficacy of our method in addressing the cooperative path planning problem is validated through simulations conducted in complex environments.

The subsequent section of the paper is shown below. In Sec. 2, the original SCA, the UAV model and the cooperative path planning problem are given. In Sec. 3 the process of the Stackelberg-Nash game is introduced with the obstacle avoidance potential function. The proposed LCD-SCA algorithm is proposed in Sec. 4. The method is demonstrated and verified through simulation results shown in Sec. 5. Finally, in Sec. 6, conclusions and future work are briefly introduced.

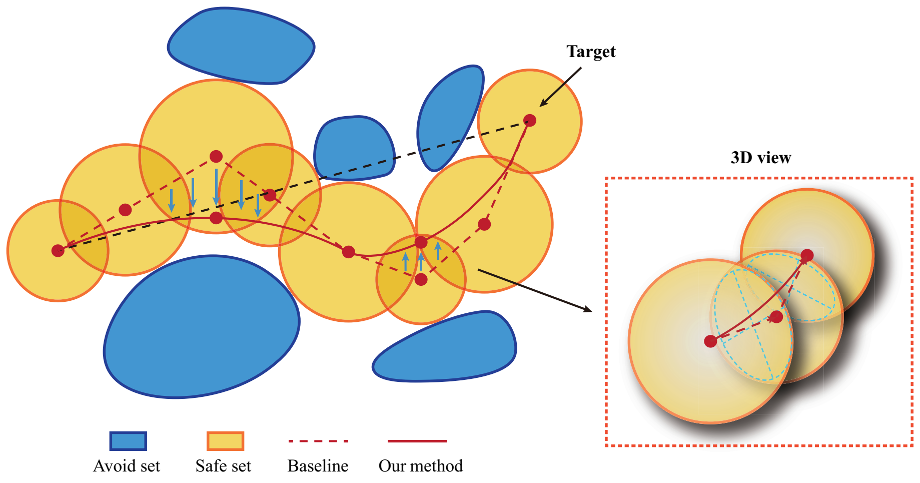


FIGURE 2 Safe sets and trajectories generated based on our baseline method and the method in this paper. The 3D view of the intersection of adjacent safe set surfaces is on the right side of the figure.

2 | PRELIMINARIES AND PROBLEM DESCRIPTION

2.1 | Problem description

A cooperative path planning problem can be defined as a situation in which the aim is to find optimal paths for all UAVs from their respective starting points to target locations. This problem can be approached by formulating an optimization problem that incorporates cost functions for each UAV in the team. The schematic framework for path planning optimization is shown in Fig. 2. The safe sets and trajectories generated by the baseline method are computed through a game-theoretic cooperative path planning. The optimized trajectories of our method are obtained through the optimization algorithm further proposed in this paper. The cost function associated with n UAVs can thus be expressed as:

$$J(P_n) = \sum_{i=1}^{\eta} \omega_i J_i(P_n) \tag{1}$$

where P_n represents the path of n UAVs, $J_i(P_n)$ is the cost corresponding to constraint i, ω_i is a weighting factor, and η is the number of constraints. The path P_n is defined by a set of k nodes, represented by waypoints $\mathbf{g}_n(k) = (x_n(k), y_n(k), z_n(k))^T$, $k = 1, 2, \ldots, K$, which connect the flight path of n UAVs. The cost function $J_i(P_n)$, $i = 1, 2, \ldots, \eta$, for each constraint is determined as follows.

2.1.1 | Formation constraint

The formation constraints are determined by the desired structure of the geometric shape and the interactions among UAVs. We define the n-th UAV as a vertex v_n and its interconnection as an edge $\epsilon_s = (v_n, v_{n'})$. The interaction topology of the UAVs is represented by a graph, denoted by $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. Here, $\mathcal{V}$ represents the node set, and $\mathcal{E}$ denotes the edge set, which is a subset of $\mathcal{V}$. The symbol $\times\mathcal{V}$ denotes the edge set. In order to establish the formation, it is necessary that the graph be connected, that is to say that for any two vertices $(v_n, v_{n'}) \in \mathcal{V}$ there exists an interconnection between them, or an edge in $\mathcal{E}$. The incidence matrix of the graph, denoted by $\mathcal{L}$, has a dimension of $n \times s$, where n is the number of vertices and s is the number of edges. The element of the matrix is equal to 1 if the UAV is the head of an edge, -1 if the UAV is the tail of an edge, and 0 otherwise. This interconnection is weighted by $\gamma_{nn'}$ as our graph is an edge-weighted graph. In this context, R^o and R^t represent the distances perceived by the UAV as obstacles or threats (other UAVs whose distance from the UAV is below the minimum safety distance). Additionally, d_{safe} signifies the minimum distance deemed safe for UAVs to navigate between themselves or obstacles. This distance is defined by the user.

The formation error for edge $(v_n, v_{n'})$ is computed from $\mathbf{g}_n - \mathbf{g}_{n'} - \mathbf{g}_{n_r n'_r}$. The total formation error can be expressed via the incidence matrix as

$$\begin{aligned} E &= \sum_{n,n' \in \mathcal{E}} \gamma_{nn'} \|\mathbf{g}_n - \mathbf{g}_{n'} - \mathbf{g}_{n_r n'_r}\|_2 \\ &= (\mathbf{g} - \mathbf{g}_r)^T \hat{\mathcal{L}} \hat{\mathcal{D}} \hat{\mathcal{L}}^T (\mathbf{g} - \mathbf{g}_r) \\ &= \|\mathbf{g} - \mathbf{g}_r\|_{\hat{\mathcal{L}} \hat{\mathcal{D}} \hat{\mathcal{L}}^T} \end{aligned} \tag{2}$$

where $\hat{\mathcal{D}} = \mathcal{D} \otimes I_3$ and $\mathcal{D} = diag(\gamma_{nn'})$ is a diagonal weight matrix of dimension $s \times s$, $\hat{\mathcal{L}} = \mathcal{L} \otimes I_3$, where the operator $\otimes$ is the Kronecker product.

The Laplacian of the graph, denoted by W, is defined as the matrix $\mathcal{L}\mathcal{D}\mathcal{L}^T$. It is symmetric and positive semi-definite. The Laplacian of the graph, denoted by $\hat{W}$, is defined as the matrix $\hat{\mathcal{L}}\hat{\mathcal{D}}\hat{\mathcal{L}}^T$. It is also symmetric and positive semi-definite. This result is cited in Ref.[19]. The formation error can be expressed as $E = \|\mathbf{g} - \mathbf{g}_r\|_{\hat{W}}$. The cost function associated with the formation constraint for UAVs is defined as

$$J_1(P_n) = \sum_{k=1}^{K} \|\mathbf{g}(k) - \mathbf{g}_r\|_{\hat{W}_n} \tag{3}$$

2.1.2 | Path cost

In the planning of a path, it is necessary to minimize the length of the path in order to save time and energy, particularly when the objective is to utilize a low-cost UAV. In autonomous operations, a path typically comprises a list of waypoints, which are uploaded to the UAV as references for the flight controller to track. This is in accordance with the findings of[20], which states that a path is a series of waypoints that are uploaded to the UAV as references for the flight controller to track. A path with K waypoints can be represented by a set of $K - 1$ line segments connecting the waypoints. The path length is then simply the sum of the aforementioned segments. The cost representing the length of path n is then computed as follows: Denoting $\mathbf{g}_n(k)$ as waypoint k of path n, the cost is computed as:

$$J_2(P_n) = \sum_{k=1}^{K-1} \|\mathbf{g}_n(k+1) - \mathbf{g}_n(k)\|_2 \tag{4}$$

2.2 | Model of UAV

$\mathbb{R}^n$ with some obstacles $\mathbf{O} = \{O_1, O_2, \ldots, O_{n_o}\}$ denotes the Euclidean space n. The Euclidean space of $n \times m$ real matrices is denoted as $\mathbb{R}^{n \times m}$. Each UAV in the group is represented by the following dynamics

$$\begin{cases} \dot{x}_i = v_i \cos\gamma_i \cos\psi_i \\ \dot{y}_i = v_i \cos\gamma_i \sin\psi_i \\ \dot{z}_i = -v_i \sin\gamma_i \\ \dot{v}_i = \dfrac{T_i - D_i}{m_i} - g \sin\gamma_i \\ \dot{\gamma}_i = \dfrac{L_i \cos\mu_i}{m_i v_i} - \dfrac{g \cos\gamma_i}{v_i} \\ \dot{\psi}_i = \dfrac{L_i \sin\mu_i}{m_i v_i \cos\gamma_i} \end{cases} \tag{5}$$

where $\mathbf{g}_i = [x_i, y_i, z_i]^T$ and $\mathbf{v}_i = [\dot{x}_i, \dot{y}_i, \dot{z}_i]^T$ represent the position and velocity vectors of the i-th UAV in the inertial frame, respectively. The location of the formation is defined by $\mathbf{g} = \left[\mathbf{g}_1^T, \mathbf{g}_2^T, \ldots, \mathbf{g}_N^T\right]^T$, where N is the total number of UAVs in the team. To define the formation shape, a set of reference positions $\mathbf{g}_r = \left[\mathbf{g}_{1_r}^T, \mathbf{g}_{2_r}^T, \ldots, \mathbf{g}_{N_r}^T\right]^T$ is given, where $\mathbf{g}_{n_r} = \left(x_{n_r}, y_{n_r}, z_{n_r}\right)^T$ is the reference position of the n-th UAV. The desired vector between two neighbors n and n' is computed as $\mathbf{g}_{n_r n'_r} = \mathbf{g}_{n_r} - \mathbf{g}_{n'_r}$. The subscript i is used to differentiate between various UAVs for each scalar or vector. $\boldsymbol{\delta}_i = [v_i, \gamma_i, \psi_i]^T$ is a vector comprising the airspeed v_i, flight path angle γ_i, and heading angle ψ_i. Within this context, m_i, g, and μ_i denote the mass, gravity acceleration, and bank angle for each UAV, respectively. Additionally, T_i, L_i, and D_i represent the thrust force, lift force, and drag force, respectively.

3 | GAME THEORY FOR COOPERATIVE PATH PLANNING

Given the cost function $J(P_n)$ defined for each UAV, the cooperative path planning problem can be stated as finding paths P_n that simultaneously minimize $J(P_n)$. Since the value of $J(P_n)$ depends on the path P_n generated for UAVs itself as well as other paths of the remaining UAVs in the team, finding optimal solutions remains a challenging problem. To address the challenge of cooperative path planning for UAVs, we propose a game-theoretic enhanced SCA algorithm comprising two stages. Initially, a Stackelberg-Nash game is formulated from the cooperative path planning problem. Thereafter, an enhanced SCA algorithm is developed to resolve the Stackelberg-Nash game, thereby identifying optimal paths.

3.1 | Stackelberg-Nash game for cooperative path planning

We first introduce some assumptions.

Assumption 1. The continuous solution space of infinite states maps to a finite discrete set.

Assumption 2. All UAV models are considered as particles.

Assumption 3. Each UAV within the group is identified as either a decision-maker or player. Consequently, the UAVs that assume leading and following roles are respectively regarded as the primary decision-makers and those that follow.

Assumption 4. The strategy of the n-th UAV is defined by its path, denoted by P_n.

Assumption 5. The optimal payoff for the n-th UAV is its cost function, denoted by $J(P_n)$.

The Stackelberg game is a theoretical model designed to address the issue of asymmetric competition between a dominant decision-maker and their subordinates. In this game, the leader initiates the movement. Subsequently, the followers must determine their respective strategies in response to the leader's decision, as outlined in Ref.[21]. Consequently, the Stackelberg game can be employed to model the interactions among the UAVs.

Now, the Stackelberg game for UAVs can be described as $G_S = \big((l,f), (\alpha_l, \alpha_f), (J_l, J_f)\big)$, where (l,f) is a set of players with the leader l and followers f defined as a subset $f = (f_1, f_2, \ldots, f_N)$. The pair (α_l, α_f) stands for the strategy sets of the leader α_l and the followers α_f. They are defined respectively as $\alpha_l = \big(P_{l_1}, P_{l_2}, \ldots, P_{l_{\sum}}\big)$, where P_{l_n} is the decision strategy made by the leader ($\varrho = 1, 2, \ldots, \sum$), and $\alpha_f = \big(P_{f_1}, P_{f_2}, \ldots, P_{l_N}\big)$, where $\alpha_{f_n} = \big(P_{n_1}, P_{n_2}, \ldots, P_{n_{\sum}}\big)$ represents all $\sum$ decision strategies made by the n-th follower. The set (J_l, J_f) is the payoffs.

Let P_l^* and $P_f^* = \big(P_{f_1}^*, P_{f_2}^*, \ldots, P_{f_N}^*\big)$ be respectively the best strategy of the leader and of the followers, the Stackelberg strategy is defined as $\alpha_S^* = (P_l^*, P_f^*)$, which satisfies correspondingly

$$\begin{aligned} J_f\left(P_l, P_f^*(P_l)\right) &\leq J_f\left(P_l, P_f(P_l)\right) \\ J_l\left(P_l^*, P_f^*(P_l^*)\right) &\leq J_l\left(P_l, P_f^*(P_l)\right) \end{aligned} \tag{6}$$

The relation $P_f(P_l)$ represents the strategy P_f of the followers as a function of the leader's strategy P_l. From Eq.(6), $\alpha_S^* = (P_l^*, P_f^*)$ can be obtained as

$$\begin{aligned} P_f^* &= \arg\min_{P_f \in \alpha_f} J_f \\ P_l^* &= \arg\min_{P_l \in \alpha_l} J_l \end{aligned} \tag{7}$$

In addition to interactions between the leader and followers, it is also necessary to consider those among the followers themselves. The Nash game is employed for the purpose of modelling, utilising the symmetry inherent in the roles of the players, as evidenced in Ref.[22]. In a Nash game, each player is assumed to be aware of the optimal strategies employed by their rivals, and no player can achieve a higher payoff by modifying only their own plan. The Nash game thus provides an approach to obtain optimal results for all symmetric players, which represent the following UAVs. The Nash game can be expressed as

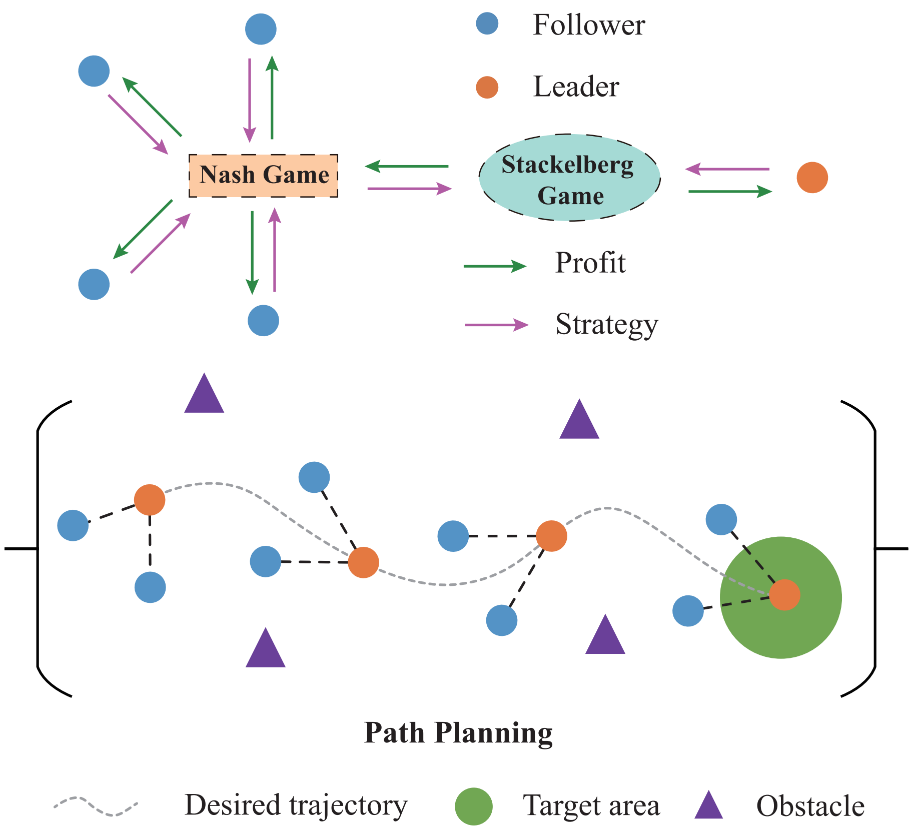


FIGURE 3 Overview of Stackelberg-Nash game and path planning.

$G_N = (f, \alpha_f, J_f)$. The Nash game is defined as the vector $\alpha_f^* = (P_{f_1}^*, P_{f_2}^*, \ldots, P_{f_N}^*)$, which satisfies the following condition:

$$\forall P_{n_\varrho} \in \alpha_{f_n},\ J_{f_n}(P_{f_n}^*, \bar{P}_{f_n}^*) \leq J_{f_n}(P_{n_\varrho}, \bar{P}_{f_n}^*) \tag{8}$$

where $\bar{P}_{f_n}^* = (P_{f_1}^*, \ldots, P_{f_{n-1}}^*, P_{f_{n+1}}^*, \ldots, P_{f_N}^*)$ is the optimal strategy set of P_n's rivals. The Nash strategy is obtained as

$$P_{f_n}^* = \arg\min_{P_{f_n}} J_{f_n}(P_{f_n}, \bar{P}_{f_n}^*) \tag{9}$$

By combining the two above models, the cooperative path planning problem for multiple UAVs can be represented by a Stackelberg-Nash game illustrated in Fig. 3. The game is expressed as

$$G = \big((l, f), (\alpha_l, \alpha_f), (J_l, J_f), (f, \alpha_f, J_f)\big) \tag{10}$$

The Stackelberg-Nash strategy, $\alpha^* = (P_l^*, P_{f_1}^*, P_{f_2}^*, \ldots, P_{f_N}^*)$, is defined to meet the conditions:

$$\begin{gathered} \forall P_{n_\varrho} \in \alpha_{f_n},\ J_{f_n}(P_l, P_{f_n}^*(P_l), \bar{P}_{f_n}^*(P_l)) \leq J_{f_n}(P_l, P_{n_\varrho}(P_l), \bar{P}_{f_n}^*(P_l)) \\ J_l(P_l^*, P_f^*(P_l^*)) \leq J_l(P_l, P_f^*(P_l)) \end{gathered} \tag{11}$$

3.2 | Obstacle avoidance potential functions

To ensure the flying safety of each UAV i, it is necessary to assume that the UAV is capable of promptly detecting impediments upon entering its sensory domain. Following this, the collection of neighbouring obstacles for UAV i is ascertained by

$$\mathbf{N}_i^o = \left\{k \in \{0, 1, \ldots, n_o\} : \|\mathbf{g}_i - \mathbf{g}_i^k\| < R^o\right\} \tag{12}$$

where $\mathbf{g}_i^k$ represents the virtual obstacle agent's position, generated by projecting UAV i onto the boundary of the nearby obstacle $O_k \in \mathbf{O}$. To provide further clarification, the position of this virtual obstacle agent, denoted as $\mathbf{g}_i^k$, can be calculated by

$$\mathbf{g}_i^k = \arg\min_{\mathbf{p}_{O_k} \in O_k} \|\mathbf{p}_{O_k} - \mathbf{p}_i\|, \quad k \in \{1, 2, \ldots, n_o\} \tag{13}$$

where $\mathbf{p}_{O_k}$ is the position of any point on the boundary of the obstacle O_k.

This study investigates protruding obstacles, assuming an obstacle O_k with its center at $\mathbf{c}_k$ and a radius of influence denoted as r_k. The position $\mathbf{g}_i^k$ of the virtual obstacle agent, along with its derivative $\dot{\mathbf{g}}_i^k$, can be represented as

$$\begin{aligned} \mathbf{g}_i^k &= \frac{r_k \mathbf{g}_i}{\|\mathbf{g}_i - \mathbf{c}_k\|} + \left(1 - \frac{r_k}{\|\mathbf{g}_i - \mathbf{c}_k\|}\right) \mathbf{c}_k \\ \dot{\mathbf{g}}_i^k &= \frac{r_k}{\|\mathbf{g}_i - \mathbf{c}_k\|} \left[\mathbf{I}_3 - \frac{(\mathbf{g}_i - \mathbf{c}_k)^T (\mathbf{g}_i - \mathbf{c}_k)}{\|\mathbf{g}_i - \mathbf{c}_k\|^2}\right] \dot{\mathbf{g}}_i \end{aligned} \tag{14}$$

Besides the obstacles within the flight space, it is important to note that when the relative distance between two unmanned aerial vehicles (UAVs) falls below the minimal safety distance, one UAV may present a security risk to the other. Consequently, UAV i views UAV j as a danger requiring obstacle avoidance if $\|\mathbf{g}_i - \mathbf{g}_j\| < R^t$. Likewise, the collection of adjacent UAVs for UAV i can be obtained as

$$\mathbf{N}_i^t = \left\{ j \in \mathcal{V} \backslash \{0\} : \|\mathbf{g}_i - \mathbf{g}_j\| < R^t \right\} \tag{15}$$

This study develops obstacle avoidance potential functions and obstacle avoidance potential functions to assure the safety of UAV flight, after determining the sets of adjacent UAVs and adjacent obstacles for each UAV. Notably, prior potential functions like those presented in Ref.[23,24], constructed using the 2-norm, lack differentiability when two UAVs coincide. Drawing inspiration from[25,26], we propose the implementation of a smooth pairwise potential in order to enhance obstacle avoidance. Leveraging the adjacent sets of UAVs and obstacles for UAV i, two distinct potential functions are formulated

$$\begin{aligned} \Gamma_i^o &= \frac{1}{2} \sum_{k \in \mathbf{N}_i^o} \Psi \left(\|\mathbf{g}_i^k - \mathbf{g}_i\|_\sigma \right) = \frac{1}{2} \sum_{k \in \mathbf{N}_i^o} \int_{\|d_{safe}\|_\sigma}^{\|\mathbf{g}_i^k - \mathbf{g}_i\|_\sigma} \Phi(s)\, ds \\ \Gamma_i^t &= \frac{1}{2} \sum_{j \in \mathbf{N}_i^t} \Psi \left(\|\mathbf{g}_j - \mathbf{g}_i\|_\sigma \right) = \frac{1}{2} \sum_{j \in \mathbf{N}_i^t} \int_{\|d_{safe}\|_\sigma}^{\|\mathbf{g}_j - \mathbf{g}_i\|_\sigma} \Phi(s)\, ds \end{aligned} \tag{16}$$

where

$$\begin{aligned} \Phi(s) &= \rho \left(\frac{s}{\|d_{safe}\|_\sigma} \right) \left(\frac{s - \|d_{safe}\|_\sigma}{\sqrt{1 + \left(s - \|d_{safe}\|_\sigma\right)^2}} - 1 \right) \\ \rho(s) &= \begin{cases} 1, & s \in [0, h) \\ \frac{1}{2} \left[1 + \cos \left(\pi \frac{s - h}{1 - h} \right) \right], & s \in [h, 1] \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{17}$$

where $\rho(\cdot)$ represents an activation function smoothly transitioning between 0 and 1, determined by a parameter $0 < h < 1$. Consequently, $\Phi(s)$ smoothly diminishes to zero at $s = \|d_{safe}\|\sigma$ and maintains a value of zero for all $s > \|d_{safe}\|_\sigma$. Introducing a flag $\aleph_i$ serves to indicate the presence of obstacles or threats within the perception area of UAV i. Specifically, $\aleph_i$ equals 0 if $\mathbf{N}_i^o \cup \mathbf{N}_i^t = \emptyset$, and $\aleph_i$ equals 1 otherwise.

To find α^*, an improved SCA algorithm is developed as described in the following.

4 | AN IMPROVED SCA ALGORITHM

From the above, the UAV cooperative path planning is reduced to finding the strategy α^* of a Stackelberg-Nash game that fulfills all requirements of conditions Eq.(11) to bring the game to its equilibrium. However, simultaneously solving inequalities (11) is challenging and even impractical for analytical methods as involving non-differentiable cost functions $J(P_n)$ and dependent variables $P_f(P_l)$. Instead, heuristic optimization techniques based on swarm intelligence are more viable and computationally efficient for this problem. Among heuristic optimization techniques, SCA has been widely used for path planning problems as being effective in the optimal search. In this paper, an improved optimization algorithm based on SCA is developed to find the best strategies for the Stackelberg-Nash game. Those strategies represent the optimal paths of the UAVs.

The original SCA has limitations in the exploration phase, making it difficult to explore efficiently and converge prematurely. In this paper, we propose a combination of Linear differential decrement strategy, Chaotic and Differential evolution SCA (LCD-SCA) based on the original SCA strategy. In the proposed algorithm, the linear differential decrement strategy improves the search efficiency, chaotic mapping averages the randomness of the search, and differential evolution enhances the diversity of the exploration. This enables the proposed algorithm to reach every corner of the search space in a shorter time.

4.1 | Model of the Sine-Cosine Algorithm

The SCA utilizes fewer operators than its peers, effectively improving the balance between exploring new options and exploiting known ones. To outline the algorithm's process, an initial aggregate is generated randomly through the use of Eq.(18).

$$X_i = X_{\min} + (X_{\max} - X_{\min})\, rand\,(1, D), \quad i \in \{1, 2, \ldots, P\} \tag{18}$$

where X_i represents candidate solution i, while $X_{\min}$ and $X_{\max}$ denote the minimum and maximum boundaries for solutions. The variables D and P stand for the dimensions of the problem and the population size, respectively. Once the population undergoes evaluation through a fitness function, an iterative process commences. Within this process, solutions are updated using Eq.(19)

$$X_i(t+1) = \begin{cases} X_i(t) + r_1 \sin(r_2) \left| r_3 X_*(t) - X_i(t) \right|, & r_4 < 0.5 \\ X_i(t) + r_1 \cos(r_2) \left| r_3 X_*(t) - X_i(t) \right|, & r_4 \geq 0.5 \end{cases} \tag{19}$$

where $X_i(t+1)$ denotes the possible solution i at iteration $t+1$ and the symbol $X_i(t)$ represents the candidate solution i at iteration t, while $X_*(t)$ denotes the optimal solution at iteration t within the population. The calculation of r_1 is derived from Eq.(20)

$$r_1 = a - \frac{a}{t_{\max}} \cdot t \tag{20}$$

where t denotes the current iteration, $t_{\max}$ represents the maximum number of iterations, while a remains constant. The variables denoted as r_2, r_3, and r_4 are assigned random numbers that are limited to the intervals $[0, 2\pi]$, $[0, 2]$, and $[0, 1]$, respectively. The newly generated population is subjected to assessment within the fitness function, and this iterative procedure continues until the conclusion criterion is met. After the iterations are finished, the optimal solution is documented.

While the implementation of SCA is straightforward and necessitates a restricted set of parameters, it is not without its limitations. These limits encompass constraints during the exploration phase, early convergence, and suboptimal precision in some optimization issues. Furthermore, the SCA algorithm endeavors to identify an optimal and efficient solution within a suboptimal portion of the search space by employing a sophisticated computational approach.

4.2 | Linear differential decrement strategy

As the SCA iterates, the parameter r_1 gradually decreases, leading to a slowdown in search velocity. This slowdown enhances the algorithm's ability to explore locally while diminishing its capacity for global exploration. In Eq.(20), the slope remains constant, ensuring a consistent change in search speed. However, when initial iterations don't yield superior points, subsequent iterations combined with the rapid decrease in search velocity often cause convergence toward a local optimum by the algorithm's conclusion. To refine the classical linear decrement strategy for r_1, we introduce differential equations to formulate a novel decrement strategy, which brings us to:

$$\frac{dr_1}{dt} = \frac{a}{t_{\max}^2} \cdot t \tag{21}$$

$$\int_{r_1}^{a} dr_1 = \frac{a}{t_{\max}^2} \int_0^t \tau \, d\tau \tag{22}$$

$$r_1 = a - \frac{a}{t_{\max}^2} \cdot t^2 \tag{23}$$

From Eq.(23), it's evident that there exists a negative correlation between r_1 and t, where r_1 follows a quadratic pattern with respect to t. During the initial iterations, r_1 undergoes gradual changes, aiding in the identification of a local optimum meeting specific conditions. However, as the iterations progress towards the maximum limit, r_1 accelerates its rate of change, swiftly converging towards the global optimum after identifying the local one. This acceleration greatly enhances operational efficiency once the local optimum is found.

4.3 | Chaos theory

As shown in the previous subsection, the values of $r_4(t+1)$ at the next iteration is determined by $r_4(t) \in [0,1]$ at the previous iteration, while in LCD-SCA, r_4 is substituted by two chaotic maps[27], Eqs.(24) and (25), which correspond to logical map and tent map, respectively.

$$r_4(t+1) = \mu \cdot r_4(t)[1 - r_4(t)], \quad r_4(t) \in [0,1] \tag{24}$$

$$r_4(t+1) = \begin{cases} \dfrac{r_4(t)}{\beta}, & r_4(t) < \beta \\ \dfrac{1 - r_4(t)}{1-\beta}, & r_4(t) \geq \beta \end{cases} \tag{25}$$

and according to[27], it is clear that tent map possesses better performance, so we use Eq.(25).

While the chaos theory contributes to achieving a broader spectrum of parameters and solutions, it's crucial to recognize that within the search process, X_* might not be updated a specific number of times. This situation could lead X_* to converge toward a local optimum. To enhance search efficiency and circumvent this local optimum scenario, this paper integrates the DE algorithm. This combination aims to mitigate these limitations and facilitate improved exploration of the solution space.

4.4 | Differential evolution

Eqs.(27)-(29) represent the mutation phase in the DE. For any search agent indexed as i, its neighbors are encompassed within the interval $[i-k, i+k]$, where k is a non-zero integer within the range $[1, (n-1)/2]$. When k is set to 1, the neighborhood comprises vectors $X_{i-1}(t)$, $X_i(t)$, and $X_{i+1}(t)$.

$X_{i,*}$ is the place with the highest fitness value in close proximity to the search agent i. Two matrices, γ_1 and γ_2, are employed to increase the randomness of the search procedure, and they are both calculated as:

$$\gamma = \lambda \cdot rand(\Lambda, \Psi) \tag{26}$$

where the constant λ has been established at a value of 0.0001. Λ denotes the size of the population, and Ψ reflects the dimension of each search agent. The calculation of the local donor vector for each search agent i at iteration t, denoted as $L_i(t)$, is performed in the following

$$L_i(t) = X_i(t) + \gamma_1 \cdot \left[X_{i,*} - X_i(t)\right] + \gamma_2 \cdot \left[X_{P_1} - X_{P_2}(t)\right] \tag{27}$$

where P_1 and P_2 are two random values chosen from $[i-k, i+k]$. $X_{i,*}$ is the best solution for the current global whale population, so the global donor vector for each search agent i at the iteration t, denoted by $G_i(t)$, is computed as:

$$G_i(t) = X_{i,*} + \mathcal{F} \cdot \left[X_{i,*} - X_i(t) + X_{R_1}(t) - X_{R_2}(t)\right] \tag{28}$$

where R_1 and R_2 are two random numbers chosen in the current whale population. $\mathcal{F}$ is a predefined scale factor. Combining $L_i(t)$ and $G_i(t)$ with Eq.(29) we have the final donor vector $V_i(t)$:

$$V_i(t) = \omega G_i(t) + (1-\omega) L_i(t) \tag{29}$$

where $\omega \in (0,1)$ is a predefined coefficient. The diversity of the population is further improved after the donor vector is computed in the mutation. $U_{i,j}(t)$ denotes the value of dimension j for each search agent i at the iteration t, which is updated from the following rule:

$$U_{i,j}(t) = \begin{cases} V_{i,j}(t), & rand(\Lambda, \Psi) \geq C_{rate} \text{ or } \mathcal{X} = j \\ X_{i,j}(t), & \text{otherwise} \end{cases} \tag{30}$$

Algorithm 1 LCD-SCA algorithm

```
Initialize the population of $X_i (i = 1, 2, \cdots, n)$
Initialize control parameters $C_{rate}$, $\mathcal{F}$, $t_{\max}$, $P$ and $a$
$X \leftarrow$ generate initial population using Eq.(18)
Calculate $r_4(t+1)$ with Eq.(25)
Initialize the fitness $F_{best} \leftarrow \infty$
for $i = 1 : P$ do
    if $f(X_i) < F_{best}$ then
        $X_* \leftarrow X_i$
        $F_{best} \leftarrow f(X_i)$
    end if
end for
for $t = 1 : t_{\max}$ do
    Update $r_1$ using Eq.(23)
    for $i = 1 : P$ do
        $r_2$, $r_3 \leftarrow$ random numbers
        $X_i \leftarrow$ update the solution $i$ using Eq.(19)
        Generate the final donor vector with Eq.(29)
        Generate $U_i(t)$ with Eq.(30)
        Update position of each search agent with Eq.(31)
        $X_i \leftarrow$ check if the population is within the bounds $[X_{\min}, X_{\max}]$
        if $f(X_i) < F_{best}$ then
            $X_* \leftarrow X_i$
            $F_{best} \leftarrow f(X_i)$
        end if
    end for
end forreturn $X_*$, $F_{best}$
```

where C_{rate} denotes the crossover rate. If a random number exceeds the value of C_{rate}, $U_{i,j}(t)$ is modified by the variable $V_{i,j}(t)$. Furthermore, when the random dimension $\mathcal{X}$ is equal to j, $U_{i,j}(t)$ is also modified by $V_{i,j}(t)$. The purpose of this is to prevent scenarios in which the initial value of C_{rate} is too small, which would invalidate the cross component. Alternatively, it is modified by $X_{i,j}(t)$. During the selection process, the search agent i is chosen based on the position with a higher fitness value, denoted as $X_i(t+1)$.

$$X_i(t+1) = \begin{cases} U_i(t), & f(U_i(t)) \leq f(X_i(t)) \\ X_i(t), & f(X_i(t)) < f(U_i(t)) \end{cases} \tag{31}$$

Algorithm 1 presents the overall process for LCD-SCA. Algorithm 2 presents game theory and LCD-SCA implementation for followers and leader path planning.

5 | SIMULATION

5.1 | Basic setting

To demonstrate the efficacy and superiority of the LCD-SCA approach, we integrate it with the classical artificial potential field (APF) algorithm, employing it for trajectory planning challenges. Notably, the APF algorithm, as cited in Ref.[28,29], frequently encounters issues related to local optima. Moreover, APF allows for 3D simulation scenarios, presenting a robust validation framework to mitigate the likelihood of solutions converging to local optima.

The configuration of the simulation experiments is described below: in a 100km $\times$ 100km $\times$ 20km $x-y-z$ space, the basic setup of the UAV and obstacle positions is shown in Fig. 6, where the obstacle influence radius $X \in [5, 20]$ and $Y \in [5, 15]$. The UAV model is used in this paper. The UAV has a minimum speed of 50 meters per second, a minimum turning radius of 2

Algorithm 2 LCD-SCA Implementation for Followers and Leader Path Planning

```
Input: Leader's strategy $P_l$, map, and initial path planning information
1: Initialize LCD-SCA parameters
2: Set the number of iteration $t_f = 0$ for followers and $t_l = 0$ for leader
3: Generate random follower's and leader's strategies
4: Obtain the initial optimal follower's strategies $P_f^*(t_f)$
5: for $t_f = 1 : t_{f,\max}$ do
6:    Recall $P_f^*(t_f - 1)$
7:    Calculate $J_{f_n}(P_l, P_{f_n}(t_f), \bar{P}_{f_n}^*(t_f - 1))$, for $n = \{1, 2, \ldots, N\}$
8:    Record $P_{f_n}^*(t_f)$
9:    Update $P_{f_n}(t_f)$
10: end for
11: Obtain $P_f^*(P_l)$
12: for $t_l = 0 : t_{l,\max}$ do
13:    Calculate the leader's profit $J_l(P_l, P_f^*(P_l))$
14:    Record $P_l^*(t_l)$
15:    Update $P_l(t_l)$
16: end forreturn $(P_l^*, P_f^*)$
```

kilometers and a minimum flight altitude of 1 kilometer. In the multi-UAV trajectory planning problem, the minimum safety distance between UAVs is $[5, 5, 2]^T$ km. Based on the simulation scenario, the initial position of UAVs, the location and range of influence of the obstacles, and the location and orientation of the targets are configured. Instead of considering a single state $X(t)$ in obstacle avoidance, we use the distance between the safety set of spheres and obstacles as a obstacle constraint.

To evaluate the performance of LCD-SCA comprehensively, a mixed static and dynamic obstacle scenario is created with several fitness functions, including unimodal functions (f_1–f_5) and multimodal functions (f_6–f_9)[18,30] in Table 1. The performance of the LCD-SCA method is evaluated using a set of 10 fitness benchmark functions. This evaluation is compared to the original SCA[9] as well as two other optimization algorithms. This incorporates both self-adaptive differential sine-cosine algorithm (sdSCA)[31] with a hybridization of whale optimization algorithm, sine-cosine algorithm, levy flight (WOASCALF)[32].

TABLE 1 Description of benchmark functions

Function	Dim	Iteration	Equation
Sphere	30	100	$f_1(X) = \sum_{i=1}^{n} X_i^2$
Sumsquares	30	100	$f_2(X) = \sum_{i=1}^{n} iX_i^2$
Step	30	100	$f_3(X) = \sum_{i=1}^{n} [X_i + 0.5]^2$
Quartic	30	100	$f_4(X) = \sum_{i=1}^{n} iX_i^4 + \text{random}[0, 1)$
Rosenbrock	30	100	$f_5(X) = \sum_{i=1}^{n-1} [100(X_{i+1} - X_i^2) + (X_i - 1)^2]$
Schwefel 2.26	30	100	$f_6(X) = \sum_{i=1}^{n} -X_i \sin\left(\sqrt{\lvert X_i \rvert}\right)$
Rastrigin	30	100	$f_7(X) = \sum_{i=1}^{n} [X_i^2 - 10\cos(2\pi X_i) + 10]$
Griewank	30	100	$f_8(X) = \sum_{i=1}^{n} \frac{X_i^2}{4000} - \prod_{i=1}^{n} \cos(\frac{X_i}{\sqrt{i}}) + 1$
Ackley	30	100	$f_9(X) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} X_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi X_i)\right) + 20 + e$

Each algorithm has 100 iterations and a dimension of 30. The calculation of average errors is presented in Table 2.

TABLE 2 Comparison LCD-SCA with SCA, sdSCA and WOASCALF for unimodal and multimodal benchmark functions (These results are average errors of 100 iterations).

Function	SCA	sdSCA	WOASCALF	LCD-SCA
f_1	9.23E+08	1.74E+05	9.31E+06	**3.14E+04**
f_2	5.02E+10	3.45E+00	4.98E+05	**5.44E–01**
f_3	5.53E+01	3.41E+00	5.25E+00	**2.11E–01**
f_4	2.07E–01	8.31E+01	**4.48E–02**	6.84E–02
f_5	4.25E+02	1.15E+02	5.13E+01	**3.39E+01**
f_6	4.78E+04	2.21E+04	3.73E+03	**1.56E+01**
f_7	1.21E+02	**4.01E+00**	2.58E+01	1.00E+01
f_8	2.71E–01	8.43E+01	**4.43E–02**	6.97E–02
f_9	1.27E+03	**4.82E+02**	1.68E+03	9.73E+02

5.2 | Convergence analysis

Table 2 demonstrates that the algorithm suggested in this study outperformed the original SCA in all benchmark functions. The enhancement has notable efficacy in both unimodal and multimodal functions, as well as in the case of f_6. The improved effect was comparatively less pronounced in other functions. Furthermore, in comparison to other algorithms documented in the literature, it can be asserted that this particular algorithm exhibits a high level of efficiency and competitiveness. Two functions were selected from the unimodal and multimodal functions, respectively, for the mean fitness box plots, in order to provide a meaningful comparison. Figure 4 displays box plots of SCA and LCD-SCA for the functions f_3, f_5, f_7, and f_9.

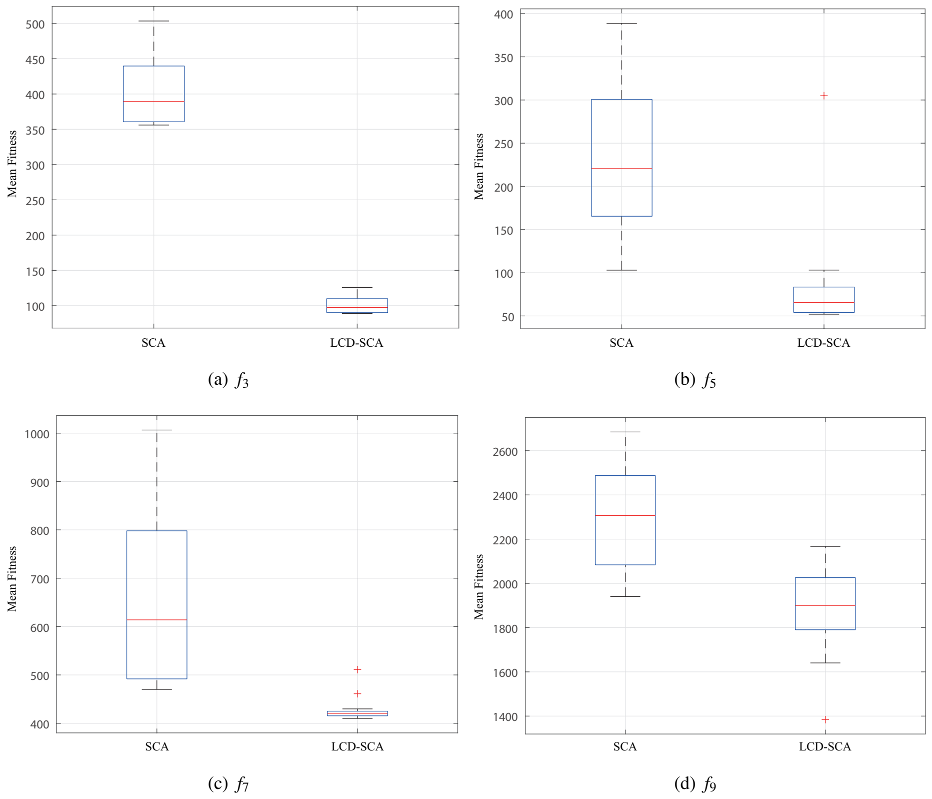


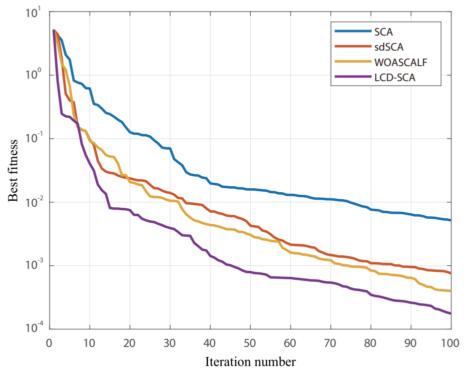
FIGURE 4 Box plots of SCA and LCD-SCA for the function f_3, f_5, f_7 and f_9.

Based on the box plots, it is evident that LCD-SCA outperforms the original approach in terms of low variance and median across most functions, and it also exhibits greater stability. The results are further supported by the maximum and minimum data obtained from both techniques.

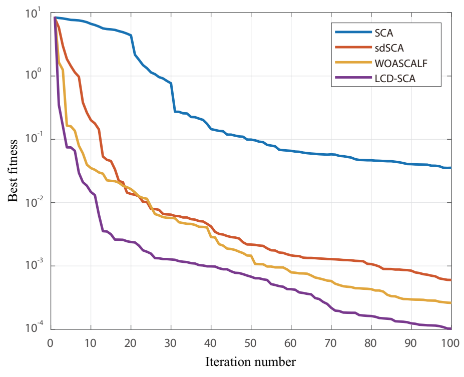
TABLE 3 Comparison of the average iteration running time for various algorithms.

Dimension	SCA	sdSCA	WOASCALF	LCD-SCA
20	0.8931	0.8817	**0.8793**	0.8803
30	2.5423	**2.3258**	2.4921	2.3941
50	14.291	16.236	13.114	**12.231**
100	43.241	40.926	41.984	**40.884**

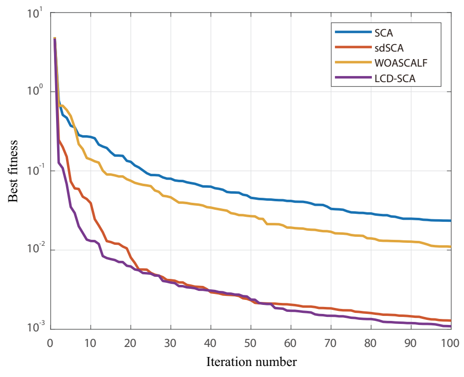
Furthermore, Fig. 5 displays the convergence curves for the four methods applied to the benchmark functions f_3, f_5, f_7, and f_9. Upon examining the optimal fitness curves, it becomes evident that the original SCA has significantly enhanced its optimization capability and convergence speed. The findings indicate that the LCD-SCA algorithm exhibits superior performance compared to the other optimization techniques.



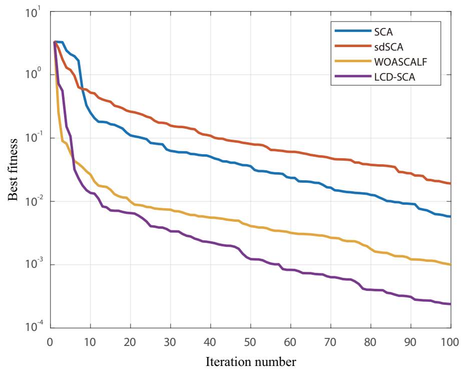
(a) f_3



(b) f_5



(c) f_7



(d) f_9

FIGURE 5 Comparison of convergence curves of LCD-SCA and other four algorithms obtained in some benchmark functions.

5.3 | Running time comparison

Additionally, as part of evaluating algorithm performance, we conducted comparative analyses of the execution times between LCD-SCA and other optimization algorithms. Table 3 presents the average time required for each iteration across the various algorithms studied. Each algorithm underwent 100 iterations in dimensions 20, 30, 50, and 100. We repeated each experiment 10 times to derive the average iteration runtime, as detailed in Table 3. The outcomes indicate that for dimensions 20 and 30, the average runtime of LCD-SCA aligns closely with the original SCA. Notably, as the dimensionality increases, LCD-SCA demonstrates superior performance compared to other optimization algorithms, showcasing notably improved average iteration runtimes.

5.4 | Cooperative path planning

To show the merit of the proposed framework using the Stackelberg-Nash game for UAV cooperative path planning, we compare it with our previously proposed path planning method[33]. The paths of the four UAVs are shown in Fig. 6 (trapezium formation). As shown in Fig. 6(a), despite the path planning and obstacle avoidance accomplished by the baseline approach, it is unable to maintain a relatively stable formation during the movement. As shown in Fig. 6(b), it can be seen that the Stackelberg-Nash game maintains a trapezoidal-like formation throughout the flight, while both intervehicle collisions and obstacle collisions can be avoided. In addition, we adjusted the formation formation to a diamond formation as shown in Fig. 7. Further, we adjusted the position of the obstacles in the scene and the formation (rectangular formation) as shown in Fig. 8.

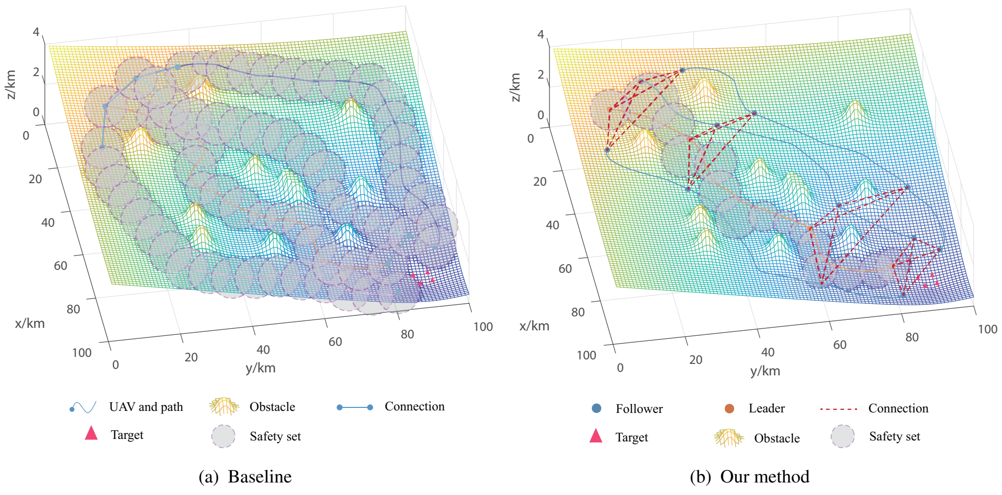


FIGURE 6 Scenario 1 (Trapezium formation): Schematic diagram of the trajectory simulation of UAVs.

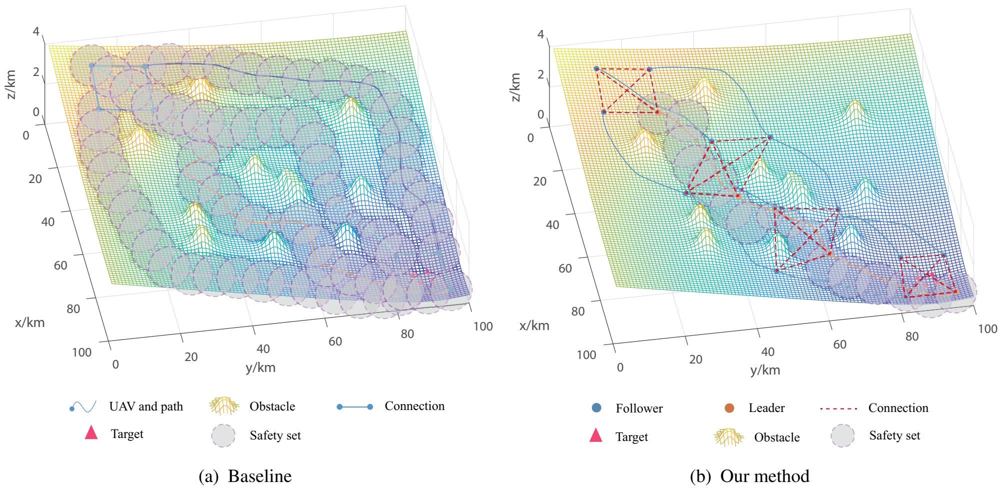
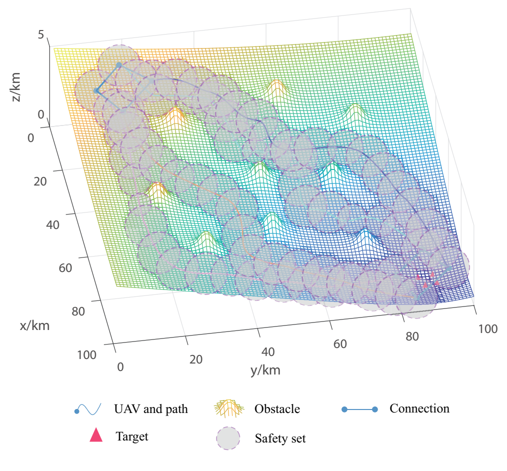
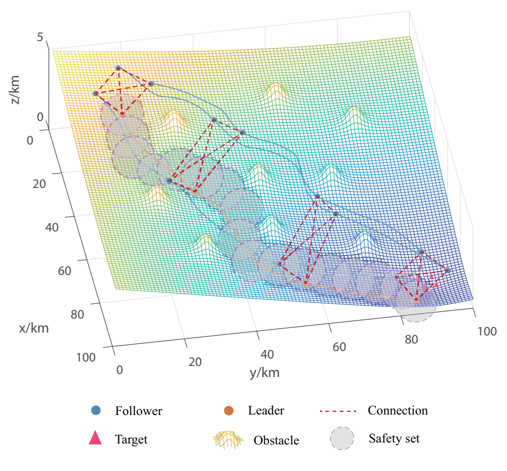


FIGURE 7 Scenario 2 (Diamond formation): Schematic diagram of the trajectory simulation of UAVs.



(a) Baseline



(b) Our method

FIGURE 8 Scenario 3 (Rectangular formation): Schematic diagram of the trajectory simulation of UAVs.

6 | CONCLUSION

In this paper, a novel method based on game theory and LCD-SCA optimization algorithm is proposed for solving the cooperative path planning challenge for multiple UAVs in a desired formation configuration. The UAV cooperative path planning problem is solved by identifying the optimal strategy for the Stackelberg-Nash game. The conventional SCA method is enhanced by incorporating linear differential decrement, chaos theory, and differential evolution, and the proposed LCD-SCA method is integrated into the path planning problem. An optimal strategy for finding the game by minimising the global cost function via the LCD-SCA method is integrated. The proposed method can generate path with safety guaranteed for flexible UAVs formation. In this paper, the proposed LCD-SCA is compared with the recently proposed optimization algorithms in terms of convergence, running time through several benchmark functions. Three cases of path planning are compared with our previous work as a validation of the method. Our future work will focus on solving the Nash equilibrium pursuit-evasion game problem through a probabilistic approach.

ACKNOWLEDGMENTS

This work was supported by the Guangdong Basic and Applied Basic Research Foundation under the Grant No. 2021A1515110569, and the National Natural Science Foundation of China under the Grant No. 52202502, and the Practice and Innovation Funds for Graduate Students of Northwestern Polytechnical University under the Grant No. PF2024040.

CONFLICT OF INTEREST

The authors declare no potential conflict of interests.

REFERENCES

1. Sujit P, Saripalli S, Sousa JB. Unmanned aerial vehicle path following: A survey and analysis of algorithms for fixed-wing unmanned aerial vehicless. *IEEE Control Systems Magazine.* 2014;34(1):42–59.
2. Motlagh NH, Taleb T, Arouk O. Low-altitude unmanned aerial vehicles-based internet of things services: Comprehensive survey and future perspectives. *IEEE Internet of Things Journal.* 2016;3(6):899–922.
3. Roger BM, others . Game theory: analysis of conflict. *The President and Fellows of Harvard College, USA.* 1991;66.
4. Moura J, Hutchison D. Game theory for multi-access edge computing: Survey, use cases, and future trends. *IEEE Communications Surveys & Tutorials.* 2018;21(1):260–288.
5. Han Z. *Game theory in wireless and communication networks: theory, models, and applications.* Cambridge university press, 2012.
6. Li C, Zhao H, Zhen S, Chen YH. Control design with optimization for fuzzy steering-by-wire system based on Nash game theory. *IEEE transactions on cybernetics.* 2021;52(8):7694–7703.
7. Zeng X, Chen J, Liang S, Hong Y. Generalized Nash equilibrium seeking strategy for distributed nonsmooth multi-cluster game. *Automatica.* 2019;103:20–26.
8. Li Y, Shi D, Chen T. False data injection attacks on networked control systems: A Stackelberg game analysis. *IEEE Transactions on Automatic Control.* 2018;63(10):3503–3509.
9. Mirjalili S. SCA: a sine cosine algorithm for solving optimization problems. *Knowledge-based systems.* 2016;96:120–133.
10. Abualigah L, Diabat A. Advances in sine cosine algorithm: a comprehensive survey. *Artificial Intelligence Review.* 2021;54(4):2567–2608.

11. Rizk-Allah RM, Hassanien AE. A comprehensive survey on the sine–cosine optimization algorithm. *Artificial Intelligence Review.* 2023;56(6):4801–4858.
12. Sahu B, Das PK, Kumar R. A modified cuckoo search algorithm implemented with SCA and PSO for multi-robot cooperation and path planning. *Cognitive Systems Research.* 2023;79:24–42.
13. Zhao M, Hou R, Li H, Ren M. A hybrid grey wolf optimizer using opposition-based learning, sine cosine algorithm and reinforcement learning for reliable scheduling and resource allocation. *Journal of Systems and Software.* 2023;205:111801.
14. Das PK. Hybridization of Kidney-Inspired and sine–cosine algorithm for multi-robot path planning. *Arabian Journal for Science and Engineering.* 2020;45(4):2883–2900.
15. Li J, Dong X, Ruan S, Shi L. A parallel integrated learning technique of improved particle swarm optimization and BP neural network and its application. *Scientific Reports.* 2022;12(1):19325.
16. Chuang LY, Yang CH, Li JC. Chaotic maps based on binary particle swarm optimization for feature selection. *Applied Soft Computing.* 2011;11(1):239–248.
17. Tirronen V, Neri F, Karkkainen T, Majava K, Rossi T. A memetic differential evolution in filter design for defect detection in paper production. In: Springer. 2007:320–329.
18. Luo J, Shi B. A hybrid whale optimization algorithm based on modified differential evolution for global optimization problems. *Applied Intelligence.* 2019;49:1982–2000.
19. Gu D. A differential game approach to formation control. *IEEE Transactions on Control Systems Technology.* 2007;16(1):85–93.
20. Phung MD, Ha QP. Safety-enhanced UAV path planning with spherical vector-based particle swarm optimization. *Applied Soft Computing.* 2021;107:107376.
21. Shang L, Abdel Aziz AM. Stackelberg game theory-based optimization model for design of payment mechanism in performance-based PPPs. *Journal of Construction Engineering and Management.* 2020;146(4):04020029.
22. Hou F, Zhai Y, You X. An equilibrium in group decision and its association with the Nash equilibrium in game theory. *Computers & Industrial Engineering.* 2020;139:106138.
23. Santos MCP, Rosales CD, Sarcinelli-Filho M, Carelli R. A novel null-space-based UAV trajectory tracking controller with collision avoidance. *IEEE/ASME Transactions on Mechatronics.* 2017;22(6):2543–2553.
24. Huber L, Billard A, Slotine JJ. Avoidance of convex and concave obstacles with convergence ensured through contraction. *IEEE Robotics and Automation Letters.* 2019;4(2):1462–1469.
25. Olfati-Saber R. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on automatic control.* 2006;51(3):401–420.
26. Chen L, Duan H. Collision-free formation-containment control for a group of UAVs with unknown disturbances. *Aerospace Science and Technology.* 2022;126:107618.
27. Abdullah AH, Enayatifar R, Lee M. A hybrid genetic algorithm and chaotic function model for image encryption. *AEU-International Journal of Electronics and Communications.* 2012;66(10):806–816.
28. Wu J, Wang H, Li N, Yao P, Huang Y, Yang H. Path planning for solar-powered UAV in urban environment. *Neurocomputing.* 2018;275:2055–2065.
29. YongBo C, YueSong M, JianQiao Y, XiaoLong S, Nuo X. Three-dimensional unmanned aerial vehicle path planning using modified wolf pack search algorithm. *Neurocomputing.* 2017;266:445–457.
30. Bi J, Gu W, Yuan H. Hybrid whale optimization algorithm with differential evolution and chaotic map operations. In: . 1. IEEE. 2021:1–6.
31. Akay R, Yildirim MY. Multi-strategy and self-adaptive differential sine-cosine algorithm for multi-robot path planning. *Expert Systems with Applications.* 2023:120849.
32. Seyyedabbasi A. WOASCALF: A new hybrid whale optimization algorithm based on sine cosine algorithm and levy flight to solve global optimization problems. *Advances in Engineering Software.* 2022;173:103272.
33. Zhang Y, Zhu Y, Li H, Wang J. A hybrid optimization algorithm for multi-agent dynamic planning with guaranteed convergence in probability. *Neurocomputing.* 2024:127764.