An improved constrained differential evolution algorithm for unmanned aerial vehicle global route planning

Xiangyin Zhang\textsuperscript{a,b}, Haibin Duan\textsuperscript{a,b,∗}

\textsuperscript{a} State Key Laboratory of Virtual Reality Technology and Systems, Beihang University (BUAA), Beijing 100191, PR China
\textsuperscript{b} Science and Technology on Aircraft Control Laboratory, School of Automation Science and Electrical Engineering, Beihang University (BUAA), Beijing 100191, PR China

\begin{abstract}
This paper formulates the global route planning problem for the unmanned aerial vehicles (UAVs) as a constrained optimization problem in the three-dimensional environment and proposes an improved constrained differential evolution (DE) algorithm to generate an optimal feasible route. The flight route is designed to have a short length and a low flight altitude. The multiple constraints based on the realistic scenarios are taken into account, including maximum turning angle, maximum climbing/gliding slope, terrain, forbidden flying areas, map and threat area constraints. The proposed DE-based route planning algorithm combines the standard DE with the level comparison method and an improved strategy is proposed to control the satisfactory level. To show the high performance of the proposed method, we compare the proposed algorithm with six existing constrained optimization algorithms and five penalty function based methods. Numerical experiments in two test cases are carried out. Our proposed algorithm demonstrates a good performance in terms of the solution quality, robustness, and the constraint-handling ability.
\end{abstract}

1. Introduction

Unmanned aerial vehicles (UAVs) can greatly increase the capability of high-risk targets penetration, suppressing enemy air defense, deep target attacking and dominating the battle space. Route planning and optimization is one of the most important problems in the autonomous navigation process of UAVs [1,2]. It allows the UAV to autonomously compute the best path from a start point to an end point [3]. For both the civil and military tasks of the UAV, the route planning problem is often formulated as an optimization problem, where the feasibility of the candidate route depends on the mission, environment and UAV physical constraints. Meanwhile, the optimality of the candidate route is evaluated according to the planning criteria, such as minimal path length or destruction risk [4]. Previous research has proposed series of algorithms to solve the UAV route planning problem, for example, the graph-based methods, such as the Voronoi diagram search method [5]; the grid-based methods, such as the mathematical programming methods, the A* searching algorithm [6] and the bi-level programming based method [7]; the nature inspired methods, such as the artificial physics algorithm [8] and the evolutionary computation technique [9–11]; and other methods, such as optimal control based methods [12] and sampling based methods.

The global and integrative optimization of the flight route is important content of the modern flight vehicle design, which can be in favor of improving flight qualities to meet the technical requirements for all determined missions [13]. It has been proven that finding the optimal route is an NP-hard problem, and the problem complexity increases very quickly as the size of the problem grows. To reduce the complexity, many researchers have applied the population-based algorithms to solve the route planning problem, including genetic algorithm (GA) [3–5], particle swarm optimization (PSO) [14,15], ant colony optimization (ACO) [16–18], artificial bee colony (ABC) [19], differential evolution (DE) [20–22], gravitational search algorithm (GSA) [23], intelligent water drops optimization (IWD) [24], and memetic computing method [25]. Zheng et al. [9] adopted the evolutionary computation to design a real-time route planner for UAV. Besada-Portas et al. [4,10] presented a UAV path planner based on evolutionary algorithms for realistic scenarios. The planner can return the paths that fulfill and optimize the multiple criteria, with the properties of real UAVs, terrains, radars, and missiles taken into consideration.
Roberge et al. [3] compared the performance of GA and PSO in computing feasible and quasi-optimal paths for fixed wing UAVs in a complex 3D environment, and a multiobjective cost function is developed to evaluate the characteristics of the optimal path. Moreover, by using a parallel implementation on standard multicore CPUs, they drew the conclusion that real-time path planning for UAVs is possible. Fu et al. [14] proposed the phase angle-encoded and quantum-behaved particle swarm optimization (θ-QPSO) to generate a safe and flyable path for the UAV in the presence of different threat environments.

Recently, differential evolution (DE) [26] has been successfully applied in many research and application areas. The recent two reviews on DE can be seen in [27] and [28]. Compared with many other population-based algorithms, DE is much simpler and straightforward to implement, which only takes several lines to code the core part of the algorithm in any programming language [28]. Despite its simplicity, DE exhibits the outstanding performance on a wide variety of problems including unimodal, multimodal, separable, non-separable and so on, and many modified versions of DE has been proposed for both unconstrained and constrained optimization problems [27–29]. Several existing researches have studied the application of the DE to the UAV route planning problem. Brintaki et al. [20] adopted the DE algorithm to design a 2D offline path planner for multi-UAVs coordinated navigation in known static maritime environments. Zhou et al. [21] presented a three-dimensional (3-D) trajectory planning algorithm for UAV based on an improved DE algorithm. Moreover, considering that the route planning problem is truly a multi-objective optimization problem in which conflicting goals of minimizing the length of the route and maximizing the margin of safety can be simultaneously important, Mittal et al. [22] used a hybrid multi-objective evolutionary algorithm to optimize the flight distance and risk factor simultaneously, and thus generated a set of Pareto-optimal routes. However, the majority of existing studies mainly employs the penalty function based methods to handle the constraints and cannot always achieve the feasible route during every computation. Different from these existing works, we formulate the UAV route planning as the constrained single objective optimization problem in the 3-D environment, and generate the feasible route by using a variant of constrained DE algorithm.

During the last few years, several DE-based methods were proposed for the constrained optimization problems [30–38]. These methods adopt various constraint-handling strategies and can be grouped into the following categories [30]: (1) methods based on preserving feasibility of solutions, (2) methods based on penalty functions, (3) methods which make a clear distinction between feasible and infeasible solutions, and (4) other hybrid methods. Inspired by the fuzzy control theory, Takahana et al. [35,38] proposed the $\alpha$ constrained method and the $\varepsilon$ constrained differential evolution algorithm ($\varepsilon$DE). Due to the excellent performance of DE on unconstrained optimization problems and the superior constraint-handling ability of $\alpha$ constrained method, Wang et al. [31,32] designed a hybrid DE algorithm with level comparison (DELCS) for the constrained optimization by incorporating the $\alpha$ constrained method into DE. In this paper, a novel UAV route planning method is proposed by combining the DE algorithm and an improved level comparison strategy. Our proposed route planning algorithm is easy to implement and numerical experiments demonstrate its effectiveness, efficiency and robustness.

The rest of this paper is organized as follows. Section 2 describes the UAV route planning problem, including the route representation, the objective function and multiple constraints in the realistic scenarios. In Section 3, the improved DE algorithm is described in detail, and a detailed implementation procedure to solve the route planning problem is also presented. Experiments and comparisons with some existing constrained algorithms are provided in Section 4. Further discussion and comparisons are presented in Section 5, including the influence of some parameters as well as the comparison with the penalty function based methods and other DE variants. Finally, Section 6 concludes with a brief summary of this paper.

## 2. Problem description

### 2.1. Route representation

As to the problem of UAV global route planning considered in this paper, we assume that the flight environment is fixed and all obstacles and threat areas are known a priori. The mission of the UAV is to fly through a high-threat region as safely as possible, and meanwhile, with the least time consumption.

In the global earth-surface inertial reference frame $S_{0}$–OXYZ, which puts the origin of coordinates $O$ on a certain point on the ground and uses three orthogonal directions as the $X$, $Y$, $Z$ axes, where the $X$, $Y$ axes are in the horizontal plane and the $Z$ axis is in the vertical direction, the labels $S : (x_{S}, y_{S}, z_{S})^{T}$ and $T : (x_{T}, y_{T}, z_{T})^{T}$ identify the starting and target points, respectively, as is shown in Fig. 1, and various danger zones, forbidden flying areas and terrain obstacles exist in the mission region. The offline route planning is to generate a short and safe route from $S$ to $T$, while satisfying the constraints. The UAV route can be described by a point set consisting of N waypoints besides $S$ and $T$ as $\mathbb{PUAV} = \{ p_{1}, p_{2}, \ldots, p_{n}, T \}$, which can be determined by $n$ control points $w_{i} : (x_{i}, y_{i}, z_{i})^{T}$ and a predefined trajectory smooth strategy. It plays a key role of the route parameterization in the global route planning problem. Bezier curves have been widely adopted when computing smooth, dynamically feasible trajectories for the UAV [4,10,11,14,15]. Employing Bezier curve, the route can be represented using a relatively smaller number of parameters than using a complete geometric description of the route. However, the construction of Bezier curves needs all the coordinates of the corresponding control points, including horizontal ordinates and vertical coordinates.

In order to reduce the dimension of the problem, a new rotated coordinate frame $S_{R}$–SX,SY,ZR shown in Fig. 1 is established using $ST$ as the new X-axis [19,19,21]. Given the straight distance from $S$ to $T$ is $L$, coordinates of the points $S$ and $T$ are $(0, 0, z_{S})^{T}$ and $(0, L, z_{T})^{T}$ in the coordinate frame $S_{R}$–SX,SY,ZR. The coordinate transformation between the two reference systems is performed according to:

$$
\begin{bmatrix}
    x' \\
    y' \\
    z'
\end{bmatrix}
= \begin{bmatrix}
    \cos \theta & -\sin \theta & 0 \\
    \sin \theta & \cos \theta & 0 \\
    0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    x \\
    y \\
    z
\end{bmatrix}
+ \begin{bmatrix}
    x_{S} \\
    y_{S} \\
    z_{S}
\end{bmatrix}
\tag{1}
$$

where $(x', y', z')^{T}$ and $(x', y', z')^{T}$ are coordinates of the same point in the original global reference frame and the rotated frame,

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{Fig_1.png}
\caption{Schematic diagram of UAV route.}
\end{figure}
respectively, and the angle $\theta$ is measured counterclockwise from the positive $OX$ axis. Divide $ST$ into $(n+1)$ equal parts with the length of $\Delta l$ by $n$ vertical lines, denoted as $l_1, l_2, \ldots, l_n$, where $\Delta l = |ST|/(n+1)$. Thus, the point in the vertical line $l_k$, $k = 1, 2, \ldots, n$, has the same horizontal coordinate $x'_i = k \Delta l$ in the coordinate frame $S_l - SX, Y_l Z_l$. If the control waypoint $w_k$ is located on the vertical line $l_k$, its location can be uniquely determined by the vertical coordinates $y'_i$ in the coordinate frame $S_l - SX, Y_l Z_l$. Therefore, the UAV route can be determined by the 2n-dimensional vector $[y'_1, y'_2, \ldots, y'_n, z'_1, z'_2, \ldots, z'_n]$, which can greatly reduce the computational cost.

In order to make the search process more efficient, we constrain the route search space between the start and target points as shown in Fig. 2, where two dash lines limit the upper and lower bounds. The search bounds are obtained by extending outward a certain distance $\Delta d$ of the points that have the maximum distance from the line $ST$. In addition, the search area should also take into account the special condition that all threat areas are distributed on one side of the line $ST$ as shown in Fig. 2(b). Based on the rotated coordinate frame $S_r - SX, Y_r Z_r$, the vertical coordinate of each waypoint is restricted in $[S_{min}, S_{max}]$, of which the maximum and minimum bounds are computed as follows:

$$ S_{min} = \min\{\min_j y'_{\text{threat}_j} - R_j, 0\} - \Delta d $$

$$ S_{max} = \max\{\max_j y'_{\text{threat}_j} + R_j, 0\} + \Delta d $$

(2)

where $y'_{\text{threat}_j}$ is the vertical coordinate of $j$th threat and $R_j$ is the radius of its threat range.

To ensure the generated route smooth and flyable, we adopt a class of dynamically feasible trajectory smooth strategy [39] to make the curve continuous. As is shown in Fig. 3, consider the sub-route defined by three control points $w_{i-1}, w_i$ and $w_{i+1}$, and let $\vec{q}_i$ be the unit vector in the direction from $w_{i-1}$ to $w_i$, and $\vec{q}_{i+1}$ be the unit vector in the direction from $w_i$ to $w_{i+1}$, as follows:

$$ \vec{q}_i = \frac{(w_i - w_{i-1})}{||w_i - w_{i-1}||} $$

$$ \vec{q}_{i+1} = \frac{(w_{i+1} - w_i)}{||w_{i+1} - w_i||} $$

(3)

Let $\varphi$ represent the angle between $\vec{q}_i$ and $\vec{q}_{i+1}$, and $\psi = \arccos(-\vec{q}_{i+1} \cdot \vec{q}_i)$. Meanwhile, let $C_i$ be a circular arc with the radius

$$ R = 0.5 \min\{|w_i - w_{i-1}|, |w_{i+1} - w_i|\} \tan \frac{\varphi}{2} $$

(4)

where the center $C_i$ lies on the bisector of the angle formed by the three control points, given by

$$ C_i = w_i + \left(\frac{R}{\tan \frac{\varphi}{2}}\right) \frac{\vec{q}_{i+1} - \vec{q}_i}{||\vec{q}_{i+1} - \vec{q}_i||} $$

(5)

The circular arc intersects the two lines $w_{i-1}w_i$ and $w_{i+1}w_i$, and the original route $w_{i-1}w_i + w_{i+1}w_i$ can be replaced by $AA + \overline{CC} + BB$. Using this strategy, the routes produced are composed of line segments and circular arcs and can be smoothed for feasible flying. As is shown in Fig. 3, the endpoints and the midpoints of each line segment together with the four equal division points of each circular arc make up the smoothed route for UAVs.

Compared with Bezier curves, the route representation method is more simple and easier to implement. One limitation of this method is that the route must be monotone along the $X$-axis, which means the UAV cannot fly around obstacles through moving backward. However, for an amount of realistic scenarios, the UAV absolutely can bypass obstacles without flying backward.

![Fig. 3. Route smooth strategy.](image)

### 2.2. Objective function and performance constraints

The objective function that evaluates a candidate route should take account of the cost of the route and the performance and mission constraints, which is expressed as follows:

$$ \min \quad J = J_L + J_H $$

s.t. \quad $g_i \leq 0$, $i = 1, 2, 3, 4$

$$ h_j = 0$, $j = 1, 2, 3$,  

where $J_L$ and $J_H$ represent the cost functions that are introduced to evaluate the performance of the candidate routes in terms of flight length and altitude, respectively. The inequality constraints $g_i$, $i = 1, 2, 3, 4$ require the candidate routes to satisfy the maximum turning angle, the climbing/gliding slope and the terrain constraints, while the equality constraints $h_j$, $j = 1, 2, 3$ can avoid the UAV flying inside the threaten and forbidden flying areas, and outside the map range.

Assume that the UAV route is represented by a collection of discrete waypoints $\{p_0, p_1, p_2, \ldots, p_N, p_{N+1}\}$ and the coordinate of $p_k$ is $(x_k, y_k, z_k)$, where the first and last waypoints are the given start and target points, respectively, and thus the cost functions and constraints of the UAV route are presented as follows:

1. **Length cost**
The cost function $J_L$ is the length of the UAV route, which is defined to provide the shorter route and calculated with the following expression:

$$ J_L = \sum_{k=0}^{N} l_k $$

with

$$ l_k = \sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2 + (z_{k+1} - z_k)^2} $$

(2) Flight altitude cost

When flying at low altitude, the UAV can benefit from the terrain mask effect, which will help itself to avoid unknown radars [4]. Therefore, the cost function $J_H$ integrates the flight altitude along the route to make the UAV tend to search the low-altitude routes, which is calculated as follows:

$$ J_H = \int_{\rho_{UAV}} H_p dl $$

with

$$ H_p = \begin{cases} 0, & \text{if } z_p < 0 \\ z_p, & \text{otherwise} \end{cases} $$

where $z_p$ is the altitude of the route at the waypoint $p$.

(3) Turning angle constraint

In order to obtain the smoother routes, the maximum turning angle $\psi_k^{\text{max}}$ is introduced at each waypoint $p_k$, which depends on the maximum lateral overload $n_{\text{max}}$ and the flight velocity $V$, given as

$$ \psi_k^{\text{max}} = \frac{n_{\text{max}} g}{V} \sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2} $$

where $g$ is the gravitational acceleration. The turning angle constraint function $g_1$ is the maximum value that the turning angle violates its upper bound, which is written as

$$ g_1 = \max(\psi_k - \psi_k^{\text{max}}) \leq 0 $$

for all $k$ in 1, 2, ..., $N$, where $\psi_k$ denotes the turning angle of the path at the waypoint $p_k$.

(4) Climbing/gliding constraint

The UAV slope $s_k$ at the waypoint $p_k$ is constrained by the maximum climbing slope $\alpha_k$ and the minimum gliding slope $\beta_k$, which both depend on the altitude. The constraint functions $g_2$ and $g_3$ can be calculated as follows [10]

$$ g_2 = \max(s_k - \alpha_k) \leq 0 $$

$$ g_3 = \max(\beta_k - s_k) \leq 0 $$

for all $k$ in 1, 2, ..., $N$, where

$$ \alpha_k = -1.5377 \times 10^{-10} z_k^2 - 26997 \times 10^{-5} z_k + 0.4211 $$

$$ \beta_k = 2.5063 \times 10^{-9} z_k^2 - 63014 \times 10^{-6} z_k - 0.3257 $$

$$ s_k = \frac{z_{k+1} - z_k}{\sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2}} $$

(5) Terrain constraint

The minimum altitude of the UAV route above the ground should be larger than a predefined safe flight height. The terrain constraint function $g_4$ is written as

$$ g_4 = H_{\text{safe}} - \min(z_k - H_{\text{terr}}(x_k, y_k)) \leq 0 $$

for all $k$ in 1, 2, ..., $N$, where $H_{\text{safe}}$ denotes the minimal safe flight height, and $H_{\text{terr}}(x_k, y_k)$ is the height of the ground at $(x_k, y_k)$.

(6) Forbidden flying area constraint

The constraint function $h_1$ is introduced to avoid the UAV getting into the user-defined forbidden flying areas (i.e., NFZs) [10], such as high-risk zones, harsh climate zones, unknown zones, large buildings [40] and so on. The UAV route must satisfy the following equality constraint:

$$ h_1 = L_{\text{inNFZ}} = 0 $$

where $L_{\text{inNFZ}}$ is the length of the UAV route that is inside the NFZs.

(7) Map constraint

In order to limit the UAV to completely fly within the map range, the constraint function $h_2$ is introduced as follows:

$$ h_2 = L_{\text{OutMap}} = 0 $$

where $L_{\text{OutMap}}$ denotes the length of the UAV route that is outside the map.

(8) Threat area constraint

The threat area constraint $h_3$ is designed to avoid the UAV flying into the scope of the ground radars and weapons. The threat function depends on the length and the threat probability of each route segment, given as

$$ h_3 = \sum_{k=0}^{N} \left( l_k \cdot \sum_{j=1}^{n_T} P_{j,k} \right) = 0 $$

where $P_{j,k}$ is the threat probability from the $j$th threat at the midpoint on the path segment $p_k p_{k+1}$. In this paper, three kinds of ground weapon units are considered, i.e., radars, missiles, and anti-air guns, and the corresponding probability-of-detection and probability-of-kill functions are calculated as follows.

(1) Radar

The probability of radar detection $P_R$ depends on the radar cross-section $RCS$ of the UAV and the distance $d$ between the UAV and the radar [10], which is calculated with the following expression:

$$ P_R = \begin{cases} 0 & \text{if } d > R_{\text{Rmax}} \\ \frac{1}{1 + \zeta_2(d^4/RCS)^{\zeta_1}} & \text{otherwise} \end{cases} $$

where $\zeta_1$ and $\zeta_2$ are the inherent parameters of the radar, $R_{\text{Rmax}}$ denotes the maximum detection distance. RCS depends on the orientation of the UAV with respect to the radar and can be calculated as follows:

$$ RCS = \frac{\pi a^2 b^2 c^2}{\sqrt{(a \alpha_2 \beta_2)^2 + (a \alpha_2 \phi_2)^2 + (c \beta_2)^2}} $$

with $a = \sin \psi$, $b = \cos \psi$, $\alpha_0 = \sin \phi$, $\beta_0 = \cos \phi$, $\phi_2 = \phi - \arctan(tan(\theta)/\sin \psi)$ with $\phi$, $\theta$ and $\psi$ denote the roll, elevation and azimuth between the positions of the UAV and the radar.

(2) Missile

Neglecting the influence of the altitude of the UAV, the simplified kill model of ground antiaircraft missile is utilized in this paper. Let $R_{\text{Mmax}}$ be the maximum hit radius of antiaircraft missile at a certain height and $d$ be the distance between UAV and the missile center, the probability-of-kill $P_M$ can be expressed by

$$ P_M = \begin{cases} 0 & \text{if } d > R_{\text{Mmax}} \\ \frac{g_M}{R_{\text{Mmax}}^2 + d^2} & \text{otherwise} \end{cases} $$

(3) Anti-aircraft gun
The probability-of-kill $P_C$ of the anti-aircraft gun is expressed by

$$
P_C = \begin{cases} 
1 & d < R_C \\
\frac{1}{d} & R_C \leq d \leq R_{G_{\text{max}}} \\
0 & \text{otherwise}
\end{cases}
$$

where $R_C$ and $R_{G_{\text{max}}}$ denote the effective and maximum hit radius, respectively, and $d$ is the distance between the UAV and the anti-aircraft gun.

3. Constrained differential evolution for global route planning

3.1. Standard differential evolution

The DE algorithm is a population-based stochastic parallel direct search method originally applied to unconstrained problem, and it operates through three computational steps, namely mutation, crossover and selection. According to its original definition by Storn and Price [26], the general structure of DE algorithm can be described in Algorithm 1.

**Algorithm 1 (Standard DE algorithm).**

Set the generation counter $t=0$

// Initialization // generate $M_{\text{pop}}$ individuals $x^i_0$ of the initial population randomly and evaluate every individuals

while termination criteria are not satisfied do

for $i=1,M_{\text{pop}}$ do

// Mutation // generate the donor vector $v^i$ using different individuals randomly selected

// Crossover // generate the trial vector $u^i$ through the random crossover scheme

// Selection // evaluate $u^i$ and perform greedy selection

end for

end while

Suppose the objective function to be minimized is $f(x), x = [x_1, \ldots, x_D] \in \mathbb{R}^D$ and the optimization search space is defined by lower and upper bound values, i.e., $x_{\text{min}} = [x_{1, \text{min}}, \ldots, x_{D, \text{min}}]$ and, $x_{\text{max}} = [x_{1, \text{max}}, \ldots, x_{D, \text{max}}]$ and. In the computation process, each individual can be symbolized by $x^i = [x^i_1, \ldots, x^i_D]$, for $i = 1, 2, \ldots, M_{\text{pop}}$, where $t=0, 1, \ldots, G_{\text{max}}$ denotes the current generation and $G_{\text{max}}$ is the maximum number of generation.

At generation $t=0$, the initialization is implemented by using a random number distribution to generate the potential individuals in the optimization search space according to

$$
x^0_{i,j} = x_{j, \text{min}} + r_j \cdot (x_{j, \text{max}} - x_{j, \text{min}})
$$

where $r_j$ is a uniformly random number in $[0,1]$.

For each target vector $x^t_i$, the mutation operation is performed as follows:

$$
v^t_i = x^t_i + F \cdot (x^t_r_2 - x^t_r_3)
$$

where $v^t_i = [v^t_{i,1}, \ldots, v^t_{i,D}]$ is known as donor vector. $r_1$, $r_2$ and $r_3$ are randomly chosen indices from the population while $r_1 \neq r_2 \neq r_3 \neq i$, and $F$ is a real positive parameter, called mutation or scale factor [38].

(2) DE/best/1:

$$
v^t_i = x^t_{\text{best}} + F \cdot (x^t_r_1 - x^t_r_2)
$$

(3) DE/best/2:

$$
v^t_i = x^t_{\text{best}} + F \cdot (x^t_r_1 - x^t_r_2) + F \cdot (x^t_r_3 - x^t_r_4)
$$

(4) DE/current-to-best/2:

$$
v^t_i = x^t_i + F \cdot (x^t_{\text{best}} - x^t_i) + F \cdot (x^t_r_3 - x^t_r_4)
$$

(5) DE/rand-to-best/2:

$$
v^t_i = x^t_i + F \cdot (x^t_{\text{best}} - x^t_r_2) + F \cdot (x^t_r_3 - x^t_r_4)
$$

(6) DE/current-to-rand/2:

$$
v^t_i = x^t_i + F \cdot (x^t_r_1 - x^t_r_2) + F \cdot (x^t_r_3 - x^t_r_4)
$$

where $x^t_{\text{best}}$ is the vector of the lowest cost in the current population.

Then, the crossover operation constructs the trial vector $u^t_i = [u^t_{i,1}, \ldots, u^t_{i,D}]$ by mixing the components of the target and donor vectors according to:

$$
u^t_{i,j} = \begin{cases} 
u^t_{i,j} & \text{if } j \leq cr \text{ or } j = rn \\
x^t_{i,j} & \text{otherwise}
\end{cases}
$$

where $r_j$ denotes a random value generated for each $j$ in accordance with a uniform distribution over $[0,1]$, $cr$ is the crossover rate between 0 and 1, $rn$ is a randomly selected index from $\{1, 2, \ldots, D\}$, which ensures that at least one component is taken from the donor vector.

Finally, DE compares the objective function values of the target $x^t_i$ and trial $u^t_i$ individuals. If the new trial individual yields an equal or lower value of the objective function, it is accepted for the next generation; otherwise the target is retained in the population. The greedy selection operator can be described as:

$$
x^{t+1} = \begin{cases} 
u^t_i & \text{if } f(u^t_i) \leq f(x^t_i) \\
x^t_i & \text{if } f(u^t_i) > f(x^t_i)
\end{cases}
$$

3.2. The $\alpha$ level comparison

The $\alpha$ level comparison based constraint-handling technique introduces the $\alpha$ satisfaction level for the constraints to indicate how well the search solutions satisfy the constraints [35]. The constrained problem can be converted to the unconstrained problem by comparing the solutions based on a predefined satisfaction level.

A general objective constrained optimization problem can be described as follows:

$$
\min_x f(x)
$$

s.t. $g_i(x) \leq 0, i = 1, 2, \ldots, n$

$h_j(x) = 0, j = 1, 2, \ldots, p$

where $x = [x_1, x_2, \ldots, x_D]$ is a D-dimensional vector of the decision variable, $f(x)$ is an objective function, $g_i(x) \leq 0$ and $h_j(x) = 0$ denote the inequality and equality constraints. The satisfaction level $\mu(x)$ is defined in the following function:

$$
\mu(x) = \begin{cases} 
1 & \text{if } g_i(x) \leq 0 \text{ and } h_j(x) = 0 \text{ for all } i, j \\
0 & \text{otherwise}
\end{cases}
$$

$$
0 \leq \mu(x) \leq 1
$$

where $\alpha$ is a real positive parameter, called constraint level.
Obviously, the solution with satisfaction level less than 1 is not feasible. Each constraint in (28) can be transformed into one of the satisfaction levels defined by piecewise linear functions on \( g_i \) and \( h_j \) as follows:

\[
\mu_{g_i}(x) = \begin{cases} 
1 & \text{if } g_i(x) \leq 0 \\
1 - \frac{g_i(x)}{b_i} & \text{if } 0 \leq g_i(x) \leq b_i \\
0 & \text{otherwise}
\end{cases} \tag{30}
\]

\[
\mu_{h_j}(x) = \begin{cases} 
1 - \frac{h_j(x)}{b_j} & \text{if } |h_j(x)| \leq b_j \\
0 & \text{otherwise}
\end{cases} \tag{31}
\]

where \( b_i \) and \( b_j \) are two positive fixed numbers.

The satisfaction level of the solution is obtained by combining the satisfaction levels of all constraints according to:

\[
\mu(X) = \min_{i,j} (\mu_{g_i}(x), \mu_{h_j}(x))
\]

Note that the constraint satisfaction level depends on parameters \( b_i (i = 1, 2, \ldots n) \) and \( b_j (j = 1, 2, \ldots p) \). The proper setting of these parameters depends on the special problems. In this paper, the median values of the constraint violations in the initial population are employed, because the median values of the random initialization can show suitable magnitudes of constraint violation [31].

The \( \alpha \) level comparison is introduced to describe the order relation on the set of solution. In order to emphasize the feasibility, the \( \alpha \) level comparison is defined by a lexicographic order in which the comparison between two solutions will give priority to \( \mu(X) \) instead of \( f(x) \).

Suppose that \( f_1, f_2 \) and \( \mu_1, \mu_2 \) are the function values and the satisfaction levels of \( x_1 \) and \( x_2 \), respectively. For any \( \alpha \in [0,1] \), the \( \alpha \) level comparison \( \preceq_{\alpha} \) and \( \succeq_{\alpha} \) between \( (f_1, \mu_1) \) and \( (f_2, \mu_2) \) are defined as follows:

\[
(f_1, \mu_1) \preceq_{\alpha} (f_2, \mu_2) \Leftrightarrow \begin{cases} 
\alpha \leq \alpha_2 & \text{if } f_1 \leq f_2 \text{ and } \mu_1 \leq \mu_2 \\
\mu_1 \geq \mu_2 & \text{otherwise}
\end{cases} \tag{33}
\]

\[
(f_1, \mu_1) \succeq_{\alpha} (f_2, \mu_2) \Leftrightarrow \begin{cases} 
\alpha \leq \alpha_2 & \text{if } f_1 < f_2 \text{ and } \mu_1 = \mu_2 \\
\mu_1 > \mu_2 & \text{otherwise}
\end{cases} \tag{34}
\]

According to the \( \alpha \) level comparison, only if two solutions both have higher satisfaction level than the predefined value or they have the same satisfaction level, the comparison between them will be performed based on the objective function value. Obviously, the \( \alpha \) level comparison is equivalent to ordinal comparison of objective function value when \( \alpha = 0 \), while the satisfaction level \( \mu(X) \) will completely replace the function value \( f(x) \) during comparison in the case of \( \alpha = 1 \).

According to the \( \alpha \) constrained method, the ordinary comparisons are replaced by the \( \alpha \) level comparison, and thus, the general constrained minimization problem is transformed to:

\[
\min_{\alpha} f(x)
\]

where \( \min_{\alpha} \) denotes the minimization based on the \( \alpha \) level comparison \( \preceq_{\alpha} \) or \( \succeq_{\alpha} \).

### 3.3. Differential evolution with level comparison

The DELC algorithm has the similar procedure as the standard DE [30,31], in which the random population initialization, the differential mutation and crossover operators are the same as those of the standard DE without any modification. However, the selection operator of DELC is based on the \( \alpha \) level comparison to handle constraints as follows:

\[
x^{t+1} = \begin{cases} 
x^t & \text{if } f(u^t) \leq f(x^t) \\
x^t & \text{if } f(u^t) > f(x^t)
\end{cases}
\]

\[
\alpha^t = \begin{cases} 
\mu_0 & t = 0 \\
(1 - \beta) \cdot \alpha^{t-1} + \beta & t < G_{\text{max}}/2 \\
1 & t \geq G_{\text{max}}/2
\end{cases}
\]

where \( \beta \) is used to control the increasing rate of \( \alpha \). \( \mu_0 \) is set as the average value of the maximum and the median of the satisfaction levels in the initial population [31], and it is limited to 0.1 in order to pay more attention to the infeasible region at the beginning of the search process.

### 3.4. Application of DELC

This section discusses the application of DELC to the route planning in details. The route planning problem described in Section 2 is modeled as a constrained optimization problem with four inequality constraints and three equality constraints. According to Eqs. (30) and (31) as well as the definitions of the constraints in Section 2.2, \( \mu_{g_i}(P_{\text{UAV}}) \) and \( \mu_{h_j}(P_{\text{UAV}}) \) can be calculated to describe whether the UAV candidate route \( P_{\text{UAV}} \) meets the inequality and equality constraints or not. Thus, the UAV candidate route’s satisfaction level \( \mu(P_{\text{UAV}}) \) is obtained by using Eq. (32). The flowchart of the DELC-based UAV route planner is shown in Fig. 4.
3.5. Further discussion and modification

In this section, the improvement strategy is presented to improve the performance of the original DELC algorithm in solving the constrained route planning problem.

As noted previously, the α level is constantly 1 and does not need to be controlled. But for those problems in which the feasible region is very small, such as the problem with several strong equality constraints, the α level should be controlled properly to obtain high-quality solutions. In Takahama et al.'s [38] and Wang et al.'s [31] works, they designed the α level control method shown in Eq. (37). Fig. 5 shows the change curves between the α value and the computing iteration under different β values. When β is large, the α level comparison strategy makes the search region decrease rapidly, which leads to the bad quality of the solutions. When β is small, the algorithm escapes the infeasible region slowly and also fails to find the optimal solution.

Although the algorithm can control the curve shape of the α value via changing β value to solve different problems, it is still not satisfactory to solve the large dimensional problems with strong equality constraints. The α value determined by Eq. (37) increases too rapidly, at the earlier searching stage, to make full use of the information in the infeasible domain. Especially for the problems with a large number of variables and small feasible region, forcing the solution to jump out of the infeasible domain prematurely cannot help to solve the problem effectively.

In this paper, we design a modified strategy to update the α value by using the sigmoid function (see Fig. 6). The α value is dynamically updated as follows:

\[
\alpha' = \begin{cases} 
\frac{1}{1 + \exp(4 - t/G_{\text{max}})} & t < G_{\text{max}}/2 \\
1 & t \geq G_{\text{max}}/2
\end{cases}
\]

(38)

where the parameter ε is introduced to control the increasing rate of α. With the different values of ε, the changing curves of the α value are illustrated in Fig. 7. Compared with Fig. 5, the α update function here is S-shaped and the α value increases slowly at earlier searching stage. Thus, the population can avoid the prematurity and have adequate time to escape from the infeasible regions.

The detailed implementation procedure of our proposed improved algorithm, which is referred to as mDELC to distinguish from the original DELC, to the UAV route planning is illustrated in Algorithm 2.

Algorithm 2 (The mDELC algorithm for solving the UAV route planning problem (DE/rand/1)).

1/* Mission */ Mission information: start and target points \((x_0, y_0)^T, (x_T, y_T)^T\), performance parameters of the UAV, and the number of the control points \(n\); Threat area information: threat types (including radar, missile and anti-aircraft gun), threat centers \((x_{\text{threat_i}}, y_{\text{threat_i}})^T\) \(i\) and the maximum radius \(R_i\); NFZ information: the vertices of the polygonal NFZs.

2/* Modeling */ (1) Establish the rotated coordinate frame \(S_0 - SX, SY, SZ\); (2) Determine the upper and lower bounds \(S_{\text{max}}, S_{\text{min}}\); (3) Compute horizontal coordinate for each control point \(x'_i = k\Delta t\) where \(\Delta t = |ST|/\left(n + 1\right)\) and \(k\) is the number of control point required;

3/* Input for mDELC */ Maximum number of generation \(G_{\text{max}}\), the population size \(M_{\text{pop}}\), the scale factor \(F\), the crossover rate \(cr\), and the parameter \(\epsilon\).

4/* Initialization */ Set the generation number \(t = 0\) and random initialize \(M_{\text{pop}}\) individuals \(x_i = [x_i^1, \ldots, x_i^D]\) where \(D = 2n\) and each element uniformly distributed in the range \([S_{\text{max}}, S_{\text{min}}]\) with \(i = 1, 2, \ldots, M_{\text{pop}}\).

5/* Evaluation */ Route generation: Generate the control point sequence \(S' = [x'_1, x'_2, \ldots, x'_i]^{T}\) for each \(x'_i\) in the coordinate frame \(S_0 - SX, SY, SZ\), where \(S' = [\{0, z_0\}^T, \{0, ST_i, z_i\}^T, \{k\Delta t, x_i^0, y_i^0\}^T]^{\infty}\) with \(k = 1, 2, \ldots, n\); transform the point sequence into the inertial reference frame \(S_0 - OXZ\); and then, generate the smooth trajectory \(P_{\text{UAV}} = \{S_0, S_1, S_2, \ldots, S_{\text{pop}}\}\); Satisfaction level computation: Compute the cost and constraints of each candidate route, and then compute its satisfaction level \(z(P_{\text{UAV}})\).

6/* Iteration computation */

while \(t < G_{\text{max}}\), do

Generation counter \(t = t + 1\) and update the value of \(\alpha'\) by using

\[
\alpha' = \begin{cases} 
\frac{1}{1 + \exp(4 - t/G_{\text{max}})} & t < G_{\text{max}}/2 \\
1 & t \geq G_{\text{max}}/2
\end{cases}
\]

(38)

for \(i = 1:M_{\text{pop}}\), do

/* Mutation */ Choose \(r_1, r_2, r_3\) from \([1, 2, \ldots, M_{\text{pop}}]\) s.t. \(r_1 \neq r_2 \neq r_3 \neq i\), and generate the donor vector \(v_{\text{d}}\) via \(v_{\text{d}} = x_i + F \cdot (x_{r_2} - x_{r_3})\).

for \(j = 1:D\), if \(v_{\text{d}j} > S_{\text{max}}\) or \(v_{\text{d}j} < S_{\text{min}}\) then \(v_{\text{d}j} = S_{\text{max}} + r_3 \cdot (S_{\text{max}} - S_{\text{min}})\).

end for

/* Crossover */ Generate the trial vector \(v_{\text{t}} = [v_{\text{t}1}, \ldots, v_{\text{t}D}]^T\) according to

\[
v_{\text{t}j} = \begin{cases} 
v_{\text{d}j} & \text{if } j \leq cr \text{ or } j = mn \\
x_i^j & \text{otherwise}
\end{cases}
\]

(39)

end for

/* Selection */ Select a new individual \(x_i\) according to the selection function

\[
\begin{align*}
P_{\text{select}}(x_i) &= \begin{cases} 
1 & \text{if } f_{\text{test}}(x_i) > f_{\text{test}}(D_{\text{old}}) \\
\frac{f_{\text{test}}(D_{\text{old}})}{f_{\text{test}}(x_i)} & \text{otherwise}
\end{cases}
\end{align*}
\]

(40)

end while

Algorithm terminates when \(t = G_{\text{max}}\).

/* Output */ Output the best route found during the run.
/*α-Selection*/ Evaluation the trial vector \( \mathbf{w}_t \), compute the satisfaction level \( \mu(\mathbf{w}_t) \), and choose \( x_t^{i+1} \) from \( x_t^i \) and \( \mathbf{w}_t \) according to
\[
x_t^{i+1} = \begin{cases} 
\mathbf{w}_t, & \text{if } \mu(\mathbf{w}_t) \geq \mu(x_t^i) \\
\mathbf{w}_t, & \text{otherwise}
\end{cases}
\]
end for
end while
/*Output*/ Decode and output the best route for UAV

4. Experiment evaluation and comparison
To assess the performance of the mDELC algorithm to solve the constrained route planning problem, series of experiments are conducted. Each algorithm is executed 50 times independently and the statistical results are used for evaluation and comparison. Performance parameters of the UAV are set as: the maximum

### Table 1
Parameters of the two test cases.

<table>
<thead>
<tr>
<th>Case</th>
<th>Start point</th>
<th>Target point</th>
<th>Scope of threats and NFZs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.</td>
<td>[50,50]</td>
<td>[950,950]</td>
<td>Anti-aircraft gun (800,650), 70; 250,200] 90.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Missile [600,500], 80; 1350,450, 80.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Radar NFZ [100,600], [100,900], 400,600).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[450,50], [450,400], [550,400], [550,50].</td>
</tr>
<tr>
<td>II.</td>
<td>[50,900]</td>
<td>[900,100]</td>
<td>Anti-aircraft gun [550,800], 70; [700,200], 90.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Missile [375,550], 80; [700,700], 80.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Radar NFZ [50,200, 50,600], [250,200].</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[500,600], [800,600], [800,300].</td>
</tr>
</tbody>
</table>

Fig. 8. Optimal UAV routes obtained by the six constrained algorithms in a 3D environment for Case I. (a) The 3-D view of the UAV routes and the danger zones. The cylinders and cubes represent the danger zones of the anti-aircraft guns, missiles and radars, as well as the NFZs to be avoided, respectively. (b) The 2-D show of (a) on the contour map. (c)-(h) The vertical profiles of the UAV routes obtained by mDELC, DELC, CPSO, rDE, jDE-2 and SaDE, respectively.
flight altitude $H_{\text{max}} = 3 \text{ km}$, the minimum safety flight altitude $H_{\text{safe}} = 50 \text{ m}$, the maximum lateral overload $n_{\text{max}} = 5 \text{ g}$, and the flight velocity $V = 200 \text{ m/s}$. The number of control points is $n = 10$, and $\Delta d = 20 \text{ km}$. The UAV mission region is 1000 km long and 1000 km wide with several known threat areas and NFZs. Two test cases are designed to compare the performance of the proposed mDELC with the original DELC [31] and other state-of-the-art constrained versions of the population-based algorithms, including the constrained partial swarm optimization (CPSO) [42], the $e$ constrained differential evolution ($e$DE) [35] and two versions of the self-adaptive differential evolution, namely SaDE [37] and jDE-2 [34]. Because the PSO and DE are the two representative optimization algorithms with strong search ability, we mainly choose the PSO and DE based variants to make the comparison. The information of the two test cases is presented in Table 1. For the anti-aircraft guns, missiles and radars, the information contains the threat center and the radius. For the NFZs, the information is the coordinates of all vertices. Both the two test cases fulfill the hypothesis that there exist feasible routes with the monotonous increasing $x_i$ values in the straight line that joins the start and end point.

The main parameters of the mDELC are set as follows: $cr = 0.9$, mutation factor $F$ is uniformly distributed in $[0.2, 0.9]$ [31], the population size $M_{\text{pop}} = 50$, the maximum iteration $C_{\text{max}} = 400$, and the control parameter $e = 10$. To make a fair comparison, the same population size and the maximum iteration number are both used for all the algorithms. Other parameters of these tested algorithms are set according to the original literature: for DELC [31], $F$ is set as a random value in $[0.2, 0.9]$, $cr = 0.9$, $\beta = 0.01$; for CPSO [42], $c_0 = 0.9$, and $c_1 = c_2 = 0.8$; for jDE-2 [34], $F_1 = 0.9$, $F_2 = 0.1$, $\tau_1 = \tau_2 = 0.1$, $k = 15$, $l = 40$; for SaDE [37], $F$ takes random values in the range $(0, 2)$ with normal distributions of mean 0.5 and standard deviation 0.3, and the learning period is 20; for $e$DE [35], $c_p = 5$, $F = 0.7$, $cr = 0.9$, $P_a = 0.01$, and $R_k = 0.3$.

Fig. 8 shows the best UAV routes of Case I generated by the above six constrained algorithms during 50 independent runs. Part (a) of these figures displays the 3-D view of UAV routes in the digital terrain, where the cylinders represent the threat areas of missiles, radars and anti-aircraft guns, and the cubes represent the NFZs. Part (b) shows the 2-D view of (a) in the contour map, where the danger zones are labeled. Parts (c)--(h) are the corresponding vertical profiles of the six UAV routes in (a), where the $X$-axis is the length of the route and the $Y$-axis is the altitude of the corresponding waypoints. During 50 runs, all the six optimization algorithms with the customized constraint-handling strategies can successfully produce the feasible routes that avoid all danger zones. However, the route obtained by mDELC has the smallest length. The best cost value $J$ of the route generated by mDELC is 1441.8, which is much smaller than those of other five algorithms.

The statistical results of mDELC, DELC, CPSO, $e$DE, jDE-2 and SaDE during 50 runs on Case I are listed in Table 2, where the best, median, mean, worst and standard deviation of the cost value are recorded. The best results are marked with boldface in Tables 2–5. The better values are highlighted using boldface. The last column FR denotes the feasible rate of the 50 runs that find the feasible routes that satisfy all constraints. Note that these statistical results listed in columns 1–5 only count those successful runs that produce the feasible solutions. From the statistics given in Table 2, it is observed that the FR of mDELC, DELC and jDE-2 can achieve 100%, while those of CPSO, SaDE and $e$DE are only 54%, 98% and 88%, respectively. The mDELC obtains the minimum best, mean, median and worst cost among all algorithms, and even the worst result produced by mDELC is better than the best results of other five algorithms. Moreover, the standard deviation of mDELC is 24.41, also is smaller than others, which further demonstrates the high robustness of mDELC. So, the performance of mDELC is superior to that of the other five tested algorithms in terms of effectiveness and robustness.

The convergence curves of the average best cost values over 50 runs are shown in Fig. 10. The mDELC algorithm is originally designed to make full use of the infeasible information, and it allows the population to select the solutions that violate the constraints in the early stage. Thus mDELC achieves a faster convergence speed and the smaller cost when $t < 0.5G_{\text{max}}$, as is shown in Fig. 9. However, when $t > 0.5G_{\text{max}}$ and the $\alpha$ value gets 1, the $\alpha$ selection strategy of mDELC forces the population out of the infeasible regions gradually and make the population evolve into the feasible solution finally. It is noted that the convergence curve of mDELC fully accords with the design original intention. Compared with other five algorithms, although the cost value of mDELC gets larger in the middle of the evolution and converges a little slower at the later stage, the fastest convergence speed of mDELC at the early stage (mainly in the infeasible regions) can also guarantee that the mDELC can achieve the best solution in the feasible regions.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Best</th>
<th>Median</th>
<th>Mean</th>
<th>Worst</th>
<th>Std.</th>
<th>FR (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>mDELC</td>
<td>1441.8</td>
<td>1452.6</td>
<td>1462.4</td>
<td>1542.5</td>
<td>24.41</td>
<td>100</td>
</tr>
<tr>
<td>DELC</td>
<td>1575.0</td>
<td>1598.7</td>
<td>1622.3</td>
<td>1755.9</td>
<td>49.60</td>
<td>100</td>
</tr>
<tr>
<td>CPPO</td>
<td>1651.9</td>
<td>1717.5</td>
<td>1731.7</td>
<td>1867.0</td>
<td>81.02</td>
<td>54</td>
</tr>
<tr>
<td>jDE-2</td>
<td>1684.7</td>
<td>1868.7</td>
<td>1899.4</td>
<td>2256.4</td>
<td>154.57</td>
<td>100</td>
</tr>
<tr>
<td>SaDE</td>
<td>1598.8</td>
<td>1683.7</td>
<td>1752.7</td>
<td>2425.3</td>
<td>187.25</td>
<td>98</td>
</tr>
<tr>
<td>$e$DE</td>
<td>1555.3</td>
<td>1608.7</td>
<td>1620.9</td>
<td>1816.6</td>
<td>50.25</td>
<td>88</td>
</tr>
<tr>
<td>PSO</td>
<td>1607.6</td>
<td>1739.4</td>
<td>1779.2</td>
<td>2335.2</td>
<td>205.59</td>
<td>54</td>
</tr>
<tr>
<td>DE</td>
<td>1570.6</td>
<td>1643.3</td>
<td>1631.9</td>
<td>1676.7</td>
<td>31.72</td>
<td>76</td>
</tr>
<tr>
<td>ABC</td>
<td>1870.7</td>
<td>2045.2</td>
<td>2055.4</td>
<td>2341.5</td>
<td>168.50</td>
<td>20</td>
</tr>
<tr>
<td>$\delta$-QPSO</td>
<td>1655.7</td>
<td>1740.6</td>
<td>1776.3</td>
<td>1991.0</td>
<td>92.22</td>
<td>84</td>
</tr>
<tr>
<td>$\theta$-QPSO</td>
<td>1634.8</td>
<td>1878.3</td>
<td>1935.8</td>
<td>2171.1</td>
<td>258.93</td>
<td>78</td>
</tr>
</tbody>
</table>

Case II
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Best</th>
<th>Median</th>
<th>Mean</th>
<th>Worst</th>
<th>Std.</th>
<th>FR (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>mDELC</td>
<td>1533.6</td>
<td>1581.7</td>
<td>1597.4</td>
<td>1762.7</td>
<td>58.42</td>
<td>100</td>
</tr>
<tr>
<td>DELC</td>
<td>1775.4</td>
<td>1788.8</td>
<td>1919.5</td>
<td>3127.0</td>
<td>282.50</td>
<td>96</td>
</tr>
<tr>
<td>CPPO</td>
<td>1620.8</td>
<td>2348.8</td>
<td>2412.3</td>
<td>3593.5</td>
<td>546.30</td>
<td>50</td>
</tr>
<tr>
<td>jDE-2</td>
<td>1844.4</td>
<td>2126.6</td>
<td>2233.2</td>
<td>2867.2</td>
<td>324.75</td>
<td>100</td>
</tr>
<tr>
<td>SaDE</td>
<td>1754.1</td>
<td>1924.0</td>
<td>2120.4</td>
<td>2963.6</td>
<td>420.65</td>
<td>88</td>
</tr>
<tr>
<td>$e$DE</td>
<td>1518.7</td>
<td>1572.0</td>
<td>1648.7</td>
<td>1907.8</td>
<td>123.60</td>
<td>94</td>
</tr>
<tr>
<td>PSO</td>
<td>2072.7</td>
<td>2334.4</td>
<td>2367.0</td>
<td>2899.6</td>
<td>194.17</td>
<td>70</td>
</tr>
<tr>
<td>DE</td>
<td>1522.0</td>
<td>1768.8</td>
<td>1730.2</td>
<td>1817.1</td>
<td>77.57</td>
<td>64</td>
</tr>
<tr>
<td>ABC</td>
<td>1790.6</td>
<td>1973.9</td>
<td>1941.9</td>
<td>2053.2</td>
<td>90.01</td>
<td>28</td>
</tr>
<tr>
<td>$\delta$-QPSO</td>
<td>1810.7</td>
<td>1833.5</td>
<td>1843.3</td>
<td>1913.6</td>
<td>33.57</td>
<td>78</td>
</tr>
<tr>
<td>$\theta$-QPSO</td>
<td>1778.1</td>
<td>1860.0</td>
<td>1945.3</td>
<td>2496.1</td>
<td>234.86</td>
<td>70</td>
</tr>
</tbody>
</table>

The best results are marked with bold.

### Table 2

Performance comparison of various algorithms on the two test cases.
Fig. 9. Convergence curves of average best fitness values in Case I.

Fig. 10 displays the experimental results of Case II during 50 independent runs. The corresponding statistical results are illustrated in Table 2 and the convergence curves of average best fitness values are displayed in Fig. 11. It can be observed that all the six tested algorithms still can generate the feasible routes that avoid all threat areas and NFZs. The best routes generated by mDELC, CPSO, εDE and SaDE obviously are shorter than those generated by jDE-2 and DELC, while jDE-2 and DELC generate the routes with lower
altitude than mDELC, CPSO, εDE and SaDE. The minimum cost value obtained in Case II is 1518.7, which is generated by εDE. The FR values of the six tested algorithms are 100%, 96%, 50%, 100%, 88% and 94%, respectively, and only mDELC and jDE-2 can obtain the feasible route successfully every run. The mDELC obtains the minimum value in the mean cost, the worst cost and the standard deviation among the six tested algorithms. Meanwhile, the mDELC achieves the fastest convergence speed. Therefore, it is concluded that the mDELC has the more powerful robustness and constraint-handling ability than other five tested algorithms. In both the two test cases, the εDE also shows its powerful search ability and robustness to solve the UAV route planning problem, but the feasible rate of εDE is always lower than that of mDELC, DELC and jDE-2. In a word, the mDELC is superior to the five constrained optimization algorithms in solving the UAV route planning problem.

5. Further comparisons and discussion

From the above comparisons in Section 4, it can be seen that the mDELC is superior to the state-of-the-art constrained optimization algorithms in solving the UAV route planning problem. In this section, we further compare the mDELC with the penalty function based methods, and then discuss the influence of ε and n. In addition, we make a further comparison between the mDELC and the original DELC as well as other DE versions.

5.1. Comparing with the penalty function methods

In the penalty function methods, the extended cost function is defined by adding the constraint violation to the cost function as a penalty. The optimization of the cost function and the constraint violation is realized by the optimization of the extended cost function. In this paper, we define the extended cost function as follows:

$$f_p = f + 10^4 \left( \sum_{i} \max(0, g_i) + \sum_{j} |h_j| \right)$$

where the later term is the penalty term to penalize the infeasible solutions.

Fig. 12. Optimal UAV routes comparison between mDELC and the penalty function-based algorithms, including PSO, DE, ABC, jDE and 8-QPSO, in a 3D environment for Case I. (a) The 3-D view. (b) The 2-D show of (a) on the contour map. (c)–(g) The vertical profiles of the UAV routes obtained by PSO, DE, ABC, jDE and 8-QPSO, respectively.
In this section, we compare the mDELC with five population-based optimization algorithms in solving the constrained route planning problem, including PSO, DE [21], ABC [19], the self-adaptive differential evolution (jDE) [43], and the phase angle-encoded and quantum-behaved particle swarm optimization (θ-QPSO) [14]. The five tested algorithms use the penalty function method to handle the inequality and equality constraints, and the extended cost function (39) is treated as the cost function to be optimized. The population size and maximum iteration are set as $M_{pop} = 50$, $G_{max} = 400$ for all algorithms. Each of the algorithms is executed 50 times independently. Parameters of these algorithms are set according to the original literature: for PSO [3], $w_{max} = 0.9$, $w_{min} = 0.4$, $c_1 = c_2 = 1.49$; for ABC [19], $N_c = 25$, $N_x = 25$, $\text{Limit} = 30$; for DE [21,31,41,44], $F$ is set as a random value in $[0.2, 0.9]$, $cr = 0.9$; for jDE [43], $F_b = 0.9$, $F_1 = 0.1$, $\tau_1 = \tau_2 = 0.1$; for θ-QPSO [14], $b$ is decreased linearly from 1.0 to 0.5. Theses algorithms are tested in both Cases I and II and the experimental statistical results are recorded in Table 2. The optimal route comparisons of the mDELC and the five penalty function based algorithms for Cases I and II are shown in Figs. 12 and 14, respectively. The corresponding convergence curves of the average best cost value are shown in Figs. 13 and 15.

![Image](image-url)

**Fig. 13.** Convergence curve comparison of the average best cost value between mDELC and the penalty function based algorithms in Case I.

![Image](image-url)

**Fig. 14.** Optimal UAV routes comparison between mDELC and the five penalty function-based algorithms for Case II. (a) The 3-D view. (b) The 2-D show of (a) on the contour map. (c)-(g) The vertical profiles of the UAV routes obtained by PSO, DE, ABC, jDE and θ-QPSO, respectively.
Fig. 15. Convergence curve comparison of the average best cost value between mDELC and the penalty function based algorithms in Case II.

From Table 2, it can be seen that the penalty function based algorithms also can successfully generate the feasible routes in both two test cases. The jDE gains the larger FR than PSO, DE, ABC and θ-QPSO, and the standard deviation of jDE is even better than mDELC in Case II. The smallest best cost values obtained by the five penalty function based algorithms are 1570.6 for Case I and 1522.0 for Case II, which are both obtained by DE. All the five penalty function based algorithms have the lower FR than mDELC, DElC, jDE-2 and 8-DE in both Case I and Case II. Moreover, in Case II, the best cost value obtained by DE is even smaller than that of mDELC, and a little larger than 8-DE, but the mean cost, standard deviation and FR of DE are still inferior to those of mDELC. Comparing the experimental results between DE and mDELC, it can conclude that the level comparison strategy of mDELC has more powerful constraint-handling ability than the penalty function method. These experimental results demonstrated that the mDELC still maintains its superiority to the penalty function based algorithms in solving the UAV route planning problem.

5.2. Influence of n

In the UAV route planning problem, n represents the number of control points. Note that the dimension of the problem $D = 2n$. Generally, the selection of n is a problem dependent parameter determined by user to make a tradeoff between accuracy of the route and computational efficiency. In order to study the influence of the number of control points on the generated routes, we test the mDELC with $n = 5, 8, 10, 12, 15$ in the above two test cases. We execute the mDELC 50 times independently with each n, and the statistical results are listed in Table 3. It can be seen that, for both Case I and Case II, when $n \leq 10$, the mDELC can find feasible routes consistently and the standard deviation are rather small. When $n = 5$, the best cost value obtained by mDELC is larger than those obtained with $n = 8–10$. When $n = 12$ and 15, the FR cannot achieve 100%, and meanwhile, the standard deviation shows the significant increase. Therefore, as the number of the control points increases, the route planning problem becomes more complex and it is more difficult and unstable for the mDELC to handle the constraints and search the optimal solution. However, when the number of the control points is too small, it becomes difficult to describe the route accurately enough, and the optimal solution is not so satisfactory. So, $n = 10$ is the most appropriate value in the test case.

5.3. Influence of the $\alpha$ update strategy

As is mentioned above, the $\alpha$ level should be controlled properly to generate high quality solutions when the feasible region is rather small. Figs. 5 and 7 show the various $\alpha$ update strategies with different $\beta$ and $\epsilon$ values, respectively. In order to study the influence of the $\alpha$ update strategy, we test the performance of DELC (with $\beta = 0, 0.01, 0.05, 0.09, 0.2$) and mDELC (with $\epsilon = 5, 10, 15, 20, 25$) in solving the UAV route planning problem. The environment information of Case I and Case II is used here and each algorithm is run 50 times independently. The statistical results are listed in Table 4.

From Table 4, it can be seen that the mDELC is generally superior to the DELC for various $\beta$ and $\epsilon$. For DELC, the best cost value obtained with $\beta = 0$ is much smaller than those obtained with $\beta = 0.01, 0.05, 0.09$ and 0.2, but the FR of $\beta = 0$ is the smallest. For mDELC, all the best cost values obtained by each $\epsilon$ are generally smaller than those obtained by DELC, and the FR values are greater than 95%. So, the mDELC shows the significant superiority to the DELC, and the proposed modified $\alpha$ update strategy greatly improves the searching ability and robustness of the original algorithm. Meanwhile, as the value of $\epsilon$ increases, the best cost value changes rather little, but the standard deviation becomes large. For the UAV route planning problem in this paper, $\epsilon = 10$ is a recommended value.

5.4. Comparison of different DE variants

In this section, we compare the performance of mDELC in different DE variants, including DE/rand/1, DE/rand/2, DE/best/1, DE/best/2, DE/current-to-best/2, DE/rand-to-best/2 and DE/current-to-rand/2, detailed description of which are presented in Section 3.1. We run the mDELC with each variant 50 times independently for the two above test cases. The statistical results are recorded in Table 5. It can be seen that only the DE/rand/1 has the FR of 100% in finding the feasible route for both the two test cases. DE/best/1 and DE/best/2 are the worst two variants that cannot find the feasible route at all. The DE/current-to-rand/2 can obtain the second best route only after
Table 4
Comparison with the original DELC for different $\beta$ and mDELC for different $\epsilon$.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Parameter</th>
<th>Case I Best Median Mean Worst Std.</th>
<th>FR (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELC</td>
<td>$\beta = 0$</td>
<td>1468.8 1547.3 1565.8 1835.2 84.00 94</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$0.01$</td>
<td>1575.0 1598.7 1622.3 1755.9 49.60 100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$0.05$</td>
<td>1580.6 1661.9 1693.3 1993.8 117.94 100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$0.09$</td>
<td>1559.6 1648.4 1667.7 2147.2 116.04 100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$0.2$</td>
<td>1571.5 1664.1 1719.3 2177.8 153.43 100</td>
<td></td>
</tr>
<tr>
<td>mDELC</td>
<td>$\epsilon = 5$</td>
<td>1489.9 1547.6 1575.8 1908.0 84.22 100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$10$</td>
<td>1441.8 1452.6 1464.2 1542.5 24.41 100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$15$</td>
<td>1443.7 1530.7 1543.4 1947.3 95.78 96</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$20$</td>
<td>1442.6 1564.0 1571.0 2172.2 136.54 100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$25$</td>
<td>1444.7 1612.0 1632.0 2027.5 107.42 98</td>
<td></td>
</tr>
</tbody>
</table>

Case II

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Parameter</th>
<th>Case II Best Median Mean Worst Std.</th>
<th>FR (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELC</td>
<td>$\beta = 0$</td>
<td>1553.4 1665.1 1685.7 1962.3 100 22 92</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$0.01$</td>
<td>1775.4 1788.8 1919.5 3127.0 282.50 96</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$0.05$</td>
<td>1752.6 1799.3 2004.9 3119.6 435.06 94</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$0.09$</td>
<td>1781.8 1796.0 2056.4 3156.3 439.61 94</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$0.2$</td>
<td>1634.6 1793.3 1956.9 3136.0 413.79 100</td>
<td></td>
</tr>
<tr>
<td>mDELC</td>
<td>$\epsilon = 5$</td>
<td>2394.1 2585.8 2620.1 3092.0 184.32 100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$10$</td>
<td>1535.6 1581.7 1597.4 1762.7 58.42 100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$15$</td>
<td>1686.0 2090.4 2024.8 2243.7 203.21 100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$20$</td>
<td>1668.2 2106.1 2055.5 3166.2 344.32 100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$25$</td>
<td>1688.5 1690.3 1874.0 2245.0 242.91 100</td>
<td></td>
</tr>
</tbody>
</table>

The best results are marked with bold.

Table 5
Statistical results of the mDELC in different DE variants.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Case I Best Median Mean Worst Std.</th>
<th>FR (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DE/rand/1</td>
<td>1441.8 1452.6 1462.4 1542.5 24.41 100</td>
<td></td>
</tr>
<tr>
<td>DE/rand/2</td>
<td>1735.9 2260.7 2464.0 4736.5 681.26 100</td>
<td></td>
</tr>
<tr>
<td>DE/best/1</td>
<td>1621.5 1949.3 1945.3 2242.1 203.72 40</td>
<td></td>
</tr>
<tr>
<td>DE/best/2</td>
<td>1643.5 1912.2 2152.1 4674.9 598.86 98</td>
<td></td>
</tr>
<tr>
<td>DE/current-to-best/1</td>
<td>1483.6 1635.1 1720.8 2152.7 207.33 44</td>
<td></td>
</tr>
<tr>
<td>DE/rand-to-best/1</td>
<td>1535.6 1581.7 1597.4 1762.7 58.42 100</td>
<td></td>
</tr>
<tr>
<td>DE/rand-to-best/2</td>
<td>1895.5 2164.9 2256.6 2914.3 329.39 86</td>
<td></td>
</tr>
<tr>
<td>DE/current-to-rand/1</td>
<td>1667.0 2041.3 2227.0 4076.9 671.81 64</td>
<td></td>
</tr>
<tr>
<td>DE/current-to-rand/2</td>
<td>2089.1 2704.0 2702.9 4376.0 550.03 96</td>
<td></td>
</tr>
</tbody>
</table>

The best results are marked with bold.

DE/rand/1. Comparing the seven DE variants, the DE/rand/1 variant significantly outperforms other six variants in terms of the global searching and constraint-handling abilities as well as robustness. Thus, DE/rand/1 is employed in the mDELC algorithm to solve the UAV route planning problem.

6. Conclusion

In this paper, an improved constrained DE algorithm is proposed and successfully applied to solve the UAV route planning problem in the 3-D environments. The route is designed to have the short length and low altitude. Several practical constraints are also taken into account. The proposed route planning algorithm combines the DE-based global search ability and the level comparison-based constraint-handling techniques. A novel satisfactory level update strategy is designed in this paper, which can improve the searching ability of the original DELC by making full use of the infeasible information in the solution space. The experimental results in two test cases show the proposed route planning algorithm can successfully obtain the optimal feasible route during every run. The proposed algorithm is superior to several existing state-of-the-art constrained optimization algorithms, such as DELC, CPSO, SaDE, jDE-2 and $\varepsilon$DE, in terms of the searching and constraint-handling abilities, as well as robustness and convergence speed. Compared with the penalty function based methods, the level comparison strategy shows the more powerful constraint-handling ability by using our proposed satisfactory level updating method. In addition, the influences of some key parameters are studied and the comparisons of different DE variants are also discussed. The route representation approach used in this paper can reduce the dimension of the decision variables, but it is just suitable for those scenarios that there exist feasible routes with the monotonous coordinate values. Therefore, further improvements should be made in the future work. Moreover, we will also focus on the internal relations between the satisfactory level update strategy and different constraint conditions. We will also extend our study to the multi-UAVs cooperation route planning problem with other more practical constraints in the following work.

Acknowledgments

This work was partially supported by the Natural Science Foundation of China (NSFC) under Grant #61333004 and #61273054, Top-Notch Young Talents Program of China, and Aeronautical Foundation of China under Grant #20135851042.
Astron. 2007, 3D combat optimization 1451–1465.


References