

Article

Quantum-Behaved Loser Reverse-Learning Differential Evolution Algorithm-Based Path Planning for Unmanned Aerial Vehicle

Zhuoyun Chen, Xiangyin Zhang *  and Yao Lu

School of Information Science and Technology, Beijing University of Technology, Beijing 100124, China; chenzy@emails.bjut.edu.cn (Z.C.); paocaijiaozi@emails.bjut.edu.cn (Y.L.)

* Correspondence: xy_zhang@bjut.edu.cn; Tel.: +86-010-67396763

Abstract

This paper proposes the Quantum-behaved Loser Reverse-learning Differential Evolution (QLRDE) algorithm to address the inherent limitations of the standard Differential Evolution (DE) algorithm, including slow convergence speed and the premature stagnation in local optima. QLRDE incorporates three innovations: quantum-behaved mutation strategies suppress premature convergence by leveraging quantum mechanics, the Loser Reverse-Learning Mechanism enhances diversity by reconstructing inferior individuals through opposition-based learning, and an adaptive parameter adjustment mechanism balances exploration and exploitation to improve robustness and convergence efficiency. Experimental evaluations on twelve benchmark functions confirm that QLRDE demonstrates better performance than existing algorithms in terms of search capability and convergence speed. Furthermore, QLRDE is employed for the 3D UAV path planning problem. QLRDE can generate B-Spline-based smooth flight paths and incorporate real-world constraints into the cost function. Simulation results confirm that QLRDE outperforms several competing algorithms with respect to path quality, computational efficiency, and robustness.

Keywords: unmanned aerial vehicle; quantum-behaved; differential evolution; path planning; opposition-based learning

1. Introduction

Unmanned aerial vehicles (UAVs) are increasingly utilized across various military and civilian applications, driven by rapid technological advancement [1]. Path planning is a critical technology for enabling autonomous UAV navigation [2,3]. In complex 3D environments, a UAV path must satisfy physical and dynamic constraints, such as maximum turning angle, maximum climb angle, path length, and flight cost, while simultaneously avoiding terrain obstacles and threat areas [1]. Currently, path planning algorithms can be generally divided into four classes: heuristic algorithms, sampling-based methods, learning-based methods, and swarm intelligence-based methods. Heuristic algorithms, such as A* [4], guide the search process using designed heuristic functions, though designing effective heuristics can be challenging. Sampling-based methods, e.g., the rapidly exploring random tree (RRT) [5], can ensure path feasibility but often cannot guarantee optimality. Learning-based methods, such as reinforcement learning (RL) [6], offer adaptive optimization and generalization capabilities but typically require substantial computational resources and data. Concurrently, the rise of large AI models for end-to-end navigation [7] presents a complementary frontier where efficient optimizers could be



Academic Editors: Guangtao Ran, Yanning Guo, Chuanjiang Li and Kai Zhang

Received: 16 December 2025

Revised: 20 January 2026

Accepted: 23 January 2026

Published: 26 January 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

integrated to refine AI-generated paths with guaranteed constraint satisfaction. Swarm intelligence-based algorithms usually possess strong global search capabilities and achieve significant improvements in path planning performance, such as the Differential Evolution (DE) algorithm [8], Ant Colony Optimization (ACO) algorithm [9], Grey Wolf Optimizer (GWO) algorithm [10], Fireworks Algorithm (FWA) [11], and Particle Swarm Optimization (PSO) algorithm [11].

DE, recognized for its structural simplicity and robustness [12], has been effectively applied in a large number of engineering fields [13,14], including UAV path planning and multi-UAVs coordination [15]. Researchers have conducted extensive studies and improvements on the standard DE to enhance its performance. The adaptive DE (JADE) [16] incorporates an optional external archive and adaptive mechanisms to improve the diversity and global search ability. The multiple strategies of adaptive DE (MLSHADE-SPA and SaUSDE) [17,18] dynamically adjust and select the suitable mutation strategy based on problem characteristics and dynamic changes during evolution, thereby enhancing the algorithm's performance and convergence speed. The iterative local search adaptive DE (SHADE-ILS) [19] combines adaptive mechanisms with an iterative local search. The Multiple-Objective DE (MODE) [20] incorporates multiple differential mutation strategies and crossover operators to handle multiple-objective optimization problems. The hybrid DE (HDE) [21] enhances the robustness against noise and disturbances. However, existing DE variants still suffer from premature convergence and insufficient search efficiency when dealing with complex problems.

In recent years, several existing studies have studied the application of the DE to the UAV path planning problem. Zhou et al. [22] presented a three-dimensional trajectory planning algorithm for UAV based on an improved DE algorithm. Moreover, considering that the path planning problem is truly a multi-objective optimization problem in which conflicting goals of minimizing the length of the path and maximizing the margin of safety can be simultaneously important, Mittal et al. [23] used a hybrid multi-objective evolutionary algorithm to optimize the flight distance and risk factor simultaneously, and thus generated a set of Pareto optimal paths. However, in complex 3D environments, these methods often converge slowly, become trapped in local optima, and struggle to produce smooth, feasible paths.

Opposite-Based Learning (OBL) [24] has recently been incorporated into population-based optimization techniques to enhance their search capabilities. OBL, as a relatively simple learning mechanism, can analyze known outputs, derive optimal input parameters, and make adjustments to enhance system performance or improve operational processes. OBL does not require complex data processing, making it simple to implement and computationally cost-effective. Meanwhile, quantum theory has provided new ideas for improving optimization algorithms [25]. When individuals follow the rules of quantum-behaved movement, their solutions exhibit higher diversity, thereby expanding the search range. Quantum-behaved Particle Swarm Optimization (QPSO) utilizes quantum mechanical rules for particle motion to help particles escape local optima [26,27]. Quantum Firefly Algorithm (QFA) employs quantum movement to maintain population diversity [28].

This paper integrates quantum theory and the OBL mechanism to overcome the limitations of standard DE, and proposes a novel Quantum-behaved Loser Reverse-learning Differential Evolution (QLRDE) algorithm. First, a Loser Reverse-Learning Mechanism (LRLM) is introduced to reconstruct the positions of inferior individuals, thereby increasing population diversity. Second, quantum-behaved mutation strategies are adopted, leveraging the probabilistic nature of wave functions to suppress premature convergence. Third, an adaptive parameter adjustment strategy based on the hyperbolic tangent function is designed to dynamically balance the exploration and exploitation of solution space. Simu-

lation results demonstrate that QLRDE outperforms other algorithms on benchmark test functions. Furthermore, this paper presents a significant application of QLRDE to 3D path planning for UAV. The QLRDE encodes B-Spline-based control points as optimization individuals and employs a comprehensive cost function of UAV path. Experimental evaluations confirm that QLRDE can generate shorter, lower-altitude, and smoother flight paths compared to other algorithms.

This paper makes three main contributions. First, we propose the Quantum-behaved Loser Reverse-learning Differential Evolution (QLRDE) algorithm, which incorporates three innovations: a quantum-behaved mutation strategy to suppress premature convergence, the Loser Reverse-Learning Mechanism (LRLM) to enhance population diversity, and an adaptive parameter adjustment mechanism to balance exploration and exploitation. Second, experimental evaluations on twelve benchmark functions confirm that QLRDE demonstrates better performance than existing algorithms in terms of search capability and convergence speed, achieving solutions closer to the true global optimum. Third, the proposed QLRDE is effectively applied to the 3D UAV path planning problem. By encoding B-Spline-based control points as optimization individuals and incorporating real-world constraints into a comprehensive cost function, QLRDE generates shorter, lower-altitude, and smoother flight paths, outperforming other algorithms with respect to path quality and robustness.

This paper proceeds as follows. Section 2 focuses on the DE algorithm. The detailed QLRDE algorithm is introduced in Section 3. Section 4 compares QLRDE and other algorithms on test functions. Section 5 shows the application of QLRDE to UAV path planning. Section 6 discusses experimental results of QLRDE-based UAV path planning. Section 7 summarizes this paper finally.

2. Standard DE Algorithm

Operating as a parallel direct search method, DE follows an iterative, population-based stochastic process consisting of mutation, crossover, and selection. Algorithm 1 presents the general structure of the DE algorithm [12].

Algorithm 1. Structure of the basic DE algorithm

1. Set the generation index $t = 0$
 2. /*Initialization*/
 3. An initial population of M_{pop} individuals x_i^t is randomly generated. The fitness of each individual is then evaluated
 4. **While** stopping criteria are unsatisfied, **do**
 5. generation index $t = t + 1$
 6. **For** $i = 1: M_{\text{pop}}$ **do**
 7. /*Mutation*/
 8. generate v_i^t via a random selection of distinct individuals
 9. /*Crossover*/
 10. apply the crossover scheme to v_i^t and x_i^t to obtain u_i^t
 11. /*Selection*/
 12. evaluate u_i^t , then perform greedy selection between x_i^t and u_i^t
 13. **End for**
 14. **End while**
-

The standard DE algorithm consists of the following four steps:

(1) Initialization: Generate the random initial solution population, in which the i -th individual is denoted as $\mathbf{x}_i = (x_{i,1}, x_{i,2}, \dots, x_{i,j}, \dots, x_{i,D})$, and

$$x_{i,j}(0) = x_{j,\min} + \text{rand} \cdot (x_{j,\max} - x_{j,\min}) \quad (1)$$

where D denotes the dimensionality of search space, $i = 1, 2, \dots, M_{\text{pop}}$, $x_{j,\min}$ and $x_{j,\max}$ are the predefined search bounds, and rand is a uniformly distributed random number within $[0, 1]$.

(2) Mutation: In the g -th generation, the mutation vector $\mathbf{v}_i$ is generated by linearly combining the components of the difference vector with the corresponding components of other individuals by

$$\mathbf{v}_i = \mathbf{x}_{r1} + F \cdot (\mathbf{x}_{r2} - \mathbf{x}_{r3}) \quad (2)$$

where $\mathbf{x}_{r1}$, $\mathbf{x}_{r2}$ and $\mathbf{x}_{r3}$ are three distinct individuals that are randomly selected from the population and $i \notin \{r1, r2, r3\}$. Amplification factor $F > 0$ can scale the difference vector.

The vector $\mathbf{v}_i = (v_{i,1}, v_{i,2}, \dots, v_{i,j}, \dots, v_{i,D})$ should also satisfy the constraints of the search space as follows:

$$v_{i,j} = \begin{cases} x_{j,\min}, & \text{if } v_{i,j} \leq x_{j,\min} \\ x_{j,\max}, & \text{if } v_{i,j} \geq x_{j,\max} \end{cases} \quad (3)$$

(3) Crossover: Perform a crossover operation using $\mathbf{v}_i$ and $\mathbf{x}_i$ to generate $\mathbf{u}_i = (u_{i,1}, u_{i,2}, \dots, u_{i,j}, \dots, u_{i,D})$ by

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } \text{rand} \leq CR_{\text{new}} \text{ or } j = j_{\text{rand}} \\ x_{i,j}, & \text{otherwise} \end{cases} \quad (4)$$

where CR is the crossover rate and j_{rand} is a randomly selected integer from 1 to D .

(4) Selection: By comparing $\mathbf{u}_i$ and $\mathbf{x}_i$, the vector with lower value of cost function is selected in the next generation, as follows:

$$\mathbf{x}_i(g+1) = \begin{cases} \mathbf{u}_i, & \text{if } f(\mathbf{u}_i) \leq f(\mathbf{x}_i) \\ \mathbf{x}_i, & \text{otherwise} \end{cases} \quad (5)$$

where the function $f(\cdot)$ represents the cost function.

3. QLRDE

In the standard DE algorithm, the mutation vector updates primarily rely on the differences between individuals, lacking clear directionality. Moreover, the DE algorithm tends to overfit the current population, further reducing diversity and weakening global search ability. As a result, DE is often stuck in local optima, especially when dealing with the high-dimensional complex problems. To address these issues, this paper proposes the QLRDE algorithm by combining quantum-behaved mutation strategies and the Loser Reverse-Learning Mechanism.

3.1. Quantum-Behaved Mutation Strategies

According to the principles of quantum mechanics [25], the population of QLRDE is viewed as a quantum system where each individual exhibits quantum behavior. Additionally, local attractors serve as centers of an attractive potential field, and individuals within this field are able to explore different positions in the feasible region.

Assuming a D -dimensional search space and a population of M_{pop} individuals, during the g -th iteration, if the position vector of individual i , $i = 1, 2, \dots, M_{\text{pop}}$, and the space is $\mathbf{x}_i = (x_{i,1}, x_{i,2}, \dots, x_{i,D})$, then the local attractor $\mathbf{q}_i$ and mutation vector $\mathbf{v}_i$ are generated by

$$\mathbf{q}_i = \mathbf{x}_{r2} - \mathbf{x}_{r3} \quad (6)$$

$$\mathbf{v}_i = \begin{cases} \mathbf{x}_{r1} + k \cdot F_{\text{new}} \cdot r_p \cdot \tanh(x \cdot \ln(1/r_x)) \cdot \mathbf{q}_i, & \text{if } \text{rand} > 0.5 \\ \mathbf{x}_{r1} - k \cdot F_{\text{new}} \cdot r_p \cdot \tanh(x \cdot \ln(1/r_x)) \cdot \mathbf{q}_i, & \text{otherwise} \end{cases} \quad (7)$$

where r_p , r_x , and rand are mutually independent random numbers from a uniform distribution in the range of $(0, 1)$. The amplification parameter k controls the intensity of quantum fluctuations acting on the mutation vector $\mathbf{v}_i$. A larger k results in a broader exploration size around the local attractor $\mathbf{q}_i$. The nonlinear parameter x controls the shape of the $\tanh(x \cdot \ln(1/r_x))$, controlling the transition from exploration to exploitation, thereby balancing exploration and exploitation capabilities.

3.2. Loser Reverse-Learning Mechanism

The LRLM allows individuals with poor performance in the search process to improve through reverse-learning, explore a wider solution space, enhance the global optimization ability, and effectively mitigate premature convergence. The implementation and application of this augmentation mechanism is detailed below.

3.2.1. Identifying the Losers

The improvement extent δ_i of $\mathbf{x}_i$ after mutation and crossover operations is defined by

$$\delta_i = f(\mathbf{x}_i) - f(\mathbf{u}_i) \quad (8)$$

where the large value of δ_i indicates that the population is improving rapidly.

Then, utilize the linear prediction [29] to estimate its costs at generation g_{max} by

$$\hat{f}(\mathbf{u}_i, g_{\text{max}}) = f(\mathbf{u}_i) - (g_{\text{max}} - g) \cdot \delta_i \quad (9)$$

If the predicted final cost $\hat{f}(\mathbf{u}_i, g_{\text{max}})$ is inferior to the current best cost $\min f(\mathbf{x}_i)$, this identifies individual $\mathbf{u}_i$ as a “loser”, named as $\mathbf{u}^L$. In this case, reverse-learning will be applied to the loser crossover vector.

3.2.2. Reverse-Learning

OBL expands the search space and increases population diversity by simultaneously exploring both self-solution and antithetical solutions, thereby preventing premature convergence to local optimum and enhancing global search capabilities [28–30].

Suppose a point in the D -dimensional space $\mathbf{X} = (x_1, x_2, \dots, x_D)$, its opposite point $\mathbf{X}' = (x'_1, x'_2, \dots, x'_D)$ can be represented as

$$x'_j = a_j + b_j - x_j \quad (10)$$

where the variable x_j is confined to the interval $[a_j, b_j]$, and $j = 1, 2, \dots, D$.

To enhance the population’s probability of locating the global optimum, the reverse-learning operation in this paper is modified to update the loser as follows:

$$\mathbf{u}'_j = \left(\mathbf{x}_{j,\text{min}} + \mathbf{x}_{j,\text{max}} - \mathbf{u}^L_j \right) \cdot (w \cdot p_1 \cdot \tanh(o \cdot \ln(1/p_2))) \quad (11)$$

where p_1 and p_2 are uniformly random numbers within $(0, 1)$. The reverse-learning intensity parameter o primarily influences the magnitude of the reverse-learning perturbation. A

larger o increases the adjustment strength applied to the loser individual u^L , encouraging its movement toward better regions of the solution space. The limitation parameter w acts as a weighting coefficient, jointly with the random number p_1 , determining the distance between the reverse-learning direction and the original opposite point. Its role is to maintain diversity and stability in the learning process. In practice, o is commonly chosen from $[0.4, 1.0]$ and w from $[0.1, 1.0]$. The selection criterion involves balancing exploration intensity (controlled by o) with convergence stability (influenced by w), often determined through sensitivity analysis as demonstrated in Section 6.2.

The procedure of the LRLM is specifically described in Algorithm 2.

Algorithm 2. Loser Reverse-Learning Mechanism

```

1.  Require: maximal generation number  $g_{\max}$ , population size  $M_{\text{pop}}$ 
2.  For  $i = 1: M_{\text{pop}}$  do
3.      If  $f(u_i) < f(x_i)$  then
4.           $\delta_i = f(x_i) - f(u_i)$ 
5.          If  $\delta_i \cdot (g_{\max} - g) < f(u_i) - \min f(x_i)$  then
6.              The crossover vector  $u_i$  is denoted as  $u^L$ .
7.              Set adjustment parameters  $o, w$ .
8.              For  $j = 1: D$ 
9.                   $u_{i,j} = (x_{j,\min} + x_{j,\max} - u_{i,j}^L) \cdot (w \cdot p_1 \cdot \tanh(o \cdot \ln(1/p_2)))$ 
10.             End for
11.             Re-evaluate and updated the cost  $f(u_i)$ 
12.         End if
13.     End if
14. End for

```

3.3. Adaptive Parameter Adjust Mechanism

In the standard DE algorithm, two adaptive parameters, the amplification factor F and crossover rate CR , are typically fixed throughout the entire evolutionary process. This static configuration tends to lead to an imbalance between search diversity and convergence precision across distinct phases, especially when tackling complex high-dimensional problems. To better balance search diversity and convergence precision throughout the search, the adaptive parameter adjust mechanism is proposed for F and CR . This mechanism dynamically adjusts these parameters for each individual based on successful mutation events, thereby enhancing the algorithm's robustness and convergence efficiency.

For each individual i in the population, the amplification factor F_i and crossover rate CR_i are updated as follows.

3.3.1. Parameter Initialization

Initially, F_i and CR_i for each individual are randomly initialized within predefined ranges:

$$\begin{cases} F_i = F_{\min} + \text{rand} \cdot (F_{\max} - F_{\min}) \\ CR_i = CR_{\min} + \text{rand} \cdot (CR_{\max} - CR_{\min}) \end{cases} \quad (12)$$

where $\text{rand} \in [0, 1]$ is a uniformly distributed random number.

3.3.2. Parameter Retention

If a trial vector u_i succeeds in replacing the target vector x_i (where $f(u_i) \leq f(x_i)$), the parameters that generated this successful mutation are considered favorable. The parameters for individual i in the next generation are updated as follows:

$$\begin{cases} F_i^{g+1} = F_i^g + rand \cdot (F_a^g - F_i^g) \\ CR_i^{g+1} = CR_i^g + rand \cdot (CR_a^g - CR_i^g) \end{cases} \quad (13)$$

where $rand \in [0, 1]$ is a uniformly distributed random number and $a \in \{1, 2, \dots, M_{pop}\}$ is a randomly chosen index distinct from i . This update introduces a stochastic element that helps preserve population diversity.

3.3.3. Parameter Updating

If the u_i fails to outperform the x_i , the parameters remain unchanged for the next generation, allowing the individual to maintain its current search direction temporarily:

$$\begin{cases} F_i^{g+1} = F_i^g \\ CR_i^{g+1} = CR_i^g \end{cases} \quad (14)$$

3.3.4. Boundary Violation Handling

After each update, the values of F_i and CR_i are constrained to their respective allowable ranges to ensure stability:

$$F_i = \begin{cases} F_{min}, & \text{if } F_i < F_{min} \\ F_{max}, & \text{if } F_i > F_{max} \\ F_i, & \text{otherwise} \end{cases} \quad (15)$$

$$CR_i = \begin{cases} CR_{min}, & \text{if } CR_i < CR_{min} \\ CR_{max}, & \text{if } CR_i > CR_{max} \\ CR_i, & \text{otherwise} \end{cases} \quad (16)$$

This adaptive mechanism enables the algorithm to automatically tailor its search parameters to the landscape of the optimization problem. By promoting effective parameter sets and discouraging ineffective ones, the mechanism contributes to a more efficient trade-off between search diversity and convergence precision. Consequently, the adaptive parameter adjust mechanism significantly enhances the convergence performance and solution optimality of the QLRDE.

During exploratory phases, the adaptive mechanism promotes higher F_i and CR_i , which enhances the mutation strength in quantum-behaved mutation for global exploration while allowing Loser Reverse-Learning Mechanism to actively reconstruct inferior individuals. During exploitation phases, adaptive mechanism reduces parameter variations through its success-based retention strategy, enabling quantum-behaved mutation to perform refinement while loser reverse-learning focuses on fine-tuning near optimal regions.

By integrating the adaptive mechanism with the quantum-behaved mutation strategies and the Loser Reverse-Learning Mechanism, we obtain a comprehensive and powerful optimization framework, which is detailed in the complete procedure of the QLRDE algorithm provided in Algorithm 3.

3.4. Process of the QLRDE Algorithm

The specific procedure of QLRDE is shown in Algorithm 3.

Algorithm 3. QLRDE

```

    /*Parameter setting*/
1.  Set algorithm parameters  $g_{\max}, M_{\text{pop}}, F_{\max}, F_{\min}, CR_{\max}, CR_{\min}$ .
    /*Initialization*/
2.  Randomly initialize  $x_i$  and calculate the costs  $J(x_i)$ .
3.  Initialize amplification factors  $F_i$  and  $CR_i$ .
4.  While  $g < g_{\max}$  do
5.      Randomly select  $a, b \in [1, M_{\text{pop}}]$ .
6.      If  $\text{rand} < 0.5$  then
7.          
$$\begin{cases} F_{\text{new}} = F_a + \text{rand} \cdot (F_b - F_a) \\ CR_{\text{new}} = CR_a + \text{rand} \cdot (1 - CR_a) \end{cases}$$

8.      Else
9.          
$$\begin{cases} F_{\text{new}} = F_a \\ CR_{\text{new}} = CR_a \end{cases}$$

10.     End if
11.     For  $i = 1: M_{\text{pop}}$  do
12.         /*Quantum behavior mutation*/
13.         Set adjustment parameters  $k, x$ .
14.         If  $\text{rand} < 0.5$  then
15.             
$$v_i = x_{r1} - k \cdot F_{\text{new}} \cdot r_p \cdot \tanh(x \cdot \ln(1/r_x)) \cdot (x_{r2} - x_{r3})$$

16.         Else
17.             
$$v_i = x_{r1} + k \cdot F_{\text{new}} \cdot r_p \cdot \tanh(x \cdot \ln(1/r_x)) \cdot (x_{r2} - x_{r3})$$

18.         End if
19.         /*Crossover*/
20.         For  $j = 1: D$  do
21.             
$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } \text{rand} \leq CR_{\text{new}} \text{ or } j = j_{\text{rand}} \\ x_{i,j}, & \text{otherwise} \end{cases}$$

22.         End for
23.     End for
24.     Apply the Loser Reverse-Learning Mechanism as described in Algorithm 2.
25.     For  $i = 1: M_{\text{pop}}$  do
26.         /*Selection*/
27.         If  $J(u_i) \leq J(x_i)$  then
28.              $x_i = u_i$ 
29.              $x_{\text{best}} = u_i$ 
30.              $F_i = F_{\text{new}}$ 
31.              $CR_i = CR_{\text{new}}$ 
32.         Else
33.              $x_i = x_i$ 
34.              $x_{\text{best}} = x_i$ 
35.              $F_i = F_i$ 
36.              $CR_i = CR_i$ 
37.         End if
38.     End for
39.      $g = g + 1$ 
40. End while
41. Return the optimal solution.

```

3.5. Computing Complexity

Let M_{pop} denote the population size, D the problem dimension, G_{max} the maximum number of generations, and C the cost function. The QLRDE algorithm consists of two main phases: initialization and iterative optimization. The computing complexity of each part is detailed below.

3.5.1. Initialization

The initialization is executed only once at the beginning of the algorithm. It involves generating M_{pop} random individuals in the D -dimensional search space and evaluating their costs, C . The computational complexity of this part is $O(M_{\text{pop}} \cdot D) + O(M_{\text{pop}} \cdot C)$.

3.5.2. Optimization

The optimization is executed in each generation and comprises four components: quantum-behaved mutation, crossover, loser reverse-learning, and selection with adaptive parameter adjust. The complexity of each component is analyzed below under worst-case assumptions.

(1) Quantum-behaved mutation: In each generation, all M_{pop} individuals undergo mutation according to Equation (7). This operation involves arithmetic computations for each of the D dimensions per individual. Therefore, the computational complexity of the mutation step is $O(M_{\text{pop}} \cdot D)$.

(2) Crossover: Following mutation, the binomial crossover in Equation (4) is applied to each individual. For each of the D dimensions, a random number is generated and compared with the crossover rate CR_{new} . The resulting complexity is $O(M_{\text{pop}} \cdot D)$.

(3) Loser reverse-learning: The LRLM (Algorithm 2) is invoked after crossover and involves three steps. First, loser identification computes the improvement δ_i (Equation (8)) and the predicted cost $\hat{f}$ (Equation (9)) for each trial vector, requiring $O(M_{\text{pop}})$ operations. Second, reverse-learning application, assuming in the worst case that all M_{pop} trial vectors are identified as losers, applies Equation (11) to every dimension of each loser, resulting in a complexity of $O(M_{\text{pop}} \cdot D)$. Third, re-evaluation re-computes the cost of the updated losers, contributing $O(M_{\text{pop}} \cdot C)$ operations. Thus, the overall worst-case complexity of LRLM is $O(M_{\text{pop}} \cdot D) + O(M_{\text{pop}} \cdot C)$.

(4) Selection with adaptive parameter adjust: First, greedy selection compares each trial vector with its corresponding target vector based on cost, requiring $O(M_{\text{pop}} \cdot C)$ operations. Second, adaptive parameter adjusts the parameters F_i and CR_i for every individual according to Equations (13) and (14), which contributes $O(M_{\text{pop}})$ operations.

Summing the dominant terms from the above components, the computational complexity per generation is $O(M_{\text{pop}} \cdot D) + O(M_{\text{pop}} \cdot C)$. The scalar operations $O(M_{\text{pop}})$ are omitted as they are dominated by the higher-order terms when D is large. Over G_{max} generations, the total computational complexity of QLRDE is $O(G_{\text{max}} \cdot M_{\text{pop}} \cdot (D + C))$.

The standard DE algorithm possesses the same asymptotic complexity of $O(G_{\text{max}} \cdot M_{\text{pop}} \cdot (D + C))$. Therefore, QLRDE maintains the identical asymptotic complexity class as standard DE, demonstrating that the introduced quantum-behaved mutation, Loser Reverse-Learning Mechanism, and adaptive parameter adjustment do not increase the algorithm's asymptotic computational burden.

In practical implementation, QLRDE introduces a modest constant-factor overhead due to the additional operations in LRLM. However, this overhead is justified by the algorithm's significantly enhanced search capability and convergence performance, as evidenced by the experimental results in the following sections. The improved convergence characteristics often enable QLRDE to attain high-quality solutions with fewer generations (G_{max}), thereby potentially reducing the total computational time in practice.

In summary, the complexity analysis confirms that QLRDE achieves substantial performance improvements while preserving the computational efficiency of the DE.

4. Test Results and Discussion on Benchmark Functions

This section presents a comparative analysis of QLRDE against standard DE, PSO, the Loser Reverse-learning Differential Evolution (LRDE), and the Global Optimal Learning Differential Evolution (GOBLDE) algorithm [13]. We conducted 50 independent tests on 12 benchmark functions for each algorithm. All comparisons were conducted on a computer with the Win-11 platform. The algorithm was written in Matlab-2024a, without the use of any commercial algorithmic tools.

4.1. Parameters and Test Functions Setting

All tested algorithms shared a common configuration, where $M_{\text{pop}} = 40$ and $g_{\text{max}} = 400$. Other parameters are set as follows: for LRDE, DE, and GOBLDE, F is a random number in $[0.1, 0.9]$, and $CR = 0.7$; for PSO, $c_1 = c_2 = 1.4$ and $w = 0.9$; for QLRDE, both the amplification factor $F \in [0.1, 0.9]$ and the crossover operator $CR \in [0.2, 0.9]$ are dynamically adjusted during the optimization process. Twelve representative benchmark functions were considered, including seven unimodal functions ($F_1, F_2, F_3, F_4, F_8, F_9, F_{11}$) to assess convergence speed and exploitation capability and five multimodal functions ($F_5, F_6, F_7, F_{10}, F_{12}$) to examine the algorithm's ability to avoid local optima and maintain population diversity, which are provided in Table 1.

Table 1. Test functions.

Function	Range
$F_1(\mathbf{X}) = \sum_{i=1}^D x_i^2$	$-100 \leq x_i \leq 100$
$F_2(\mathbf{X}) = \sum_{i=1}^D (\sum_{j=1}^i x_j)^2$	$-100 \leq x_i \leq 100$
$F_3(\mathbf{X}) = \sum_{i=1}^D (10^6)^{\frac{i-1}{D-1}} x_i^2$	$-100 \leq x_i \leq 100$
$F_4(\mathbf{X}) = \sum_{i=1}^D (x_i^2 + 0.5)^2$	$-100 \leq x_i \leq 100$
$F_5(\mathbf{X}) = -20 \exp(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D x_i^2}) - \exp(\frac{1}{D} \sum_{i=1}^D \cos(2\pi x_i)) + 20 + e$	$-32 \leq x_i \leq 32$
$F_6(\mathbf{X}) = \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i) + 10)$	$-5.12 \leq x_i \leq 5.12$
$F_7(\mathbf{X}) = \frac{1}{4000} \sum_{i=1}^D x_i^2 - \prod_{i=1}^D \cos(\frac{x_i}{\sqrt{i}}) + 1$	$-600 \leq x_i \leq 600$
$F_8(\mathbf{X}) = \sum_{i=1}^D i x_i^2$	$-5.12 \leq x_i \leq 5.12$
$F_9(\mathbf{X}) = \sum_{i=1}^D \sum_{j=1}^i x_j^2$	$-65.54 \leq x_i \leq 65.54$
$F_{10}(\mathbf{X}) = 0.5 + \frac{\sin^2(\sqrt{x_1^2 + x_2^2}) - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2}$	$-100 \leq x_i \leq 100$
$F_{11}(\mathbf{X}) = x_1^2 + x_2^2 + 25(\sin^2 x_1 + \sin^2 x_2)$	$-5 \leq x_i \leq 5$
$F_{12}(\mathbf{X}) = -\sum_{i=1}^D e^{-\frac{1}{\pi} \sum_{j=1}^D (x_j - \frac{j}{D})^2} I_0(\sqrt{\sum_{j=1}^D (x_j - \frac{j}{D})^2})$	$-5 \leq x_i \leq 5$

4.2. Results and Discussion

This section presents data analysis, iterative curve analysis, and one-way analysis of variance (ANOVA) for different algorithms on 12 benchmark functions.

4.2.1. Analysis of Statistical Results and Evolution Curves

In the cases of $D = 20$ and $D = 30$, the best, worst, median, mean, and standard deviation (std) values for each group of benchmark functions and testing algorithms are recorded in Tables 2 and 3, respectively, with the optimal values highlighted in bold. According to Table 2, in the $D = 20$, LRDE performs best on F_2 and F_9 . For the other ten functions, the best results were achieved with QLRDE. Except for F_1 , F_2 , F_3 , F_9 , and F_{12} , QLRDE has the smallest standard deviations. Table 3 indicates that, in the $D = 30$, LRDE performs best on functions F_1 , F_3 , and F_9 , whereas QLRDE performs outstandingly on the other nine functions, particularly converging to 0 on functions F_4 , F_6 , F_7 , and F_{10} . QLRDE also shows superior performance in average best function values, consistently achieving lower averages in both 20-dimensional and 30-dimensional cases, demonstrating stable performance in approaching optimal solutions. Analysis of the tables reveals that QLRDE finds solutions closer to the true global optimum across most functions and exhibits higher stability and reliability in different runs.

The evolution curves in Figures 1 and 2 show that QLRDE demonstrates rapid convergence in the early stages of the search and achieves the optimal solution in the later stages. Compared to other algorithms, QLRDE has the better search performance and faster convergence rate.

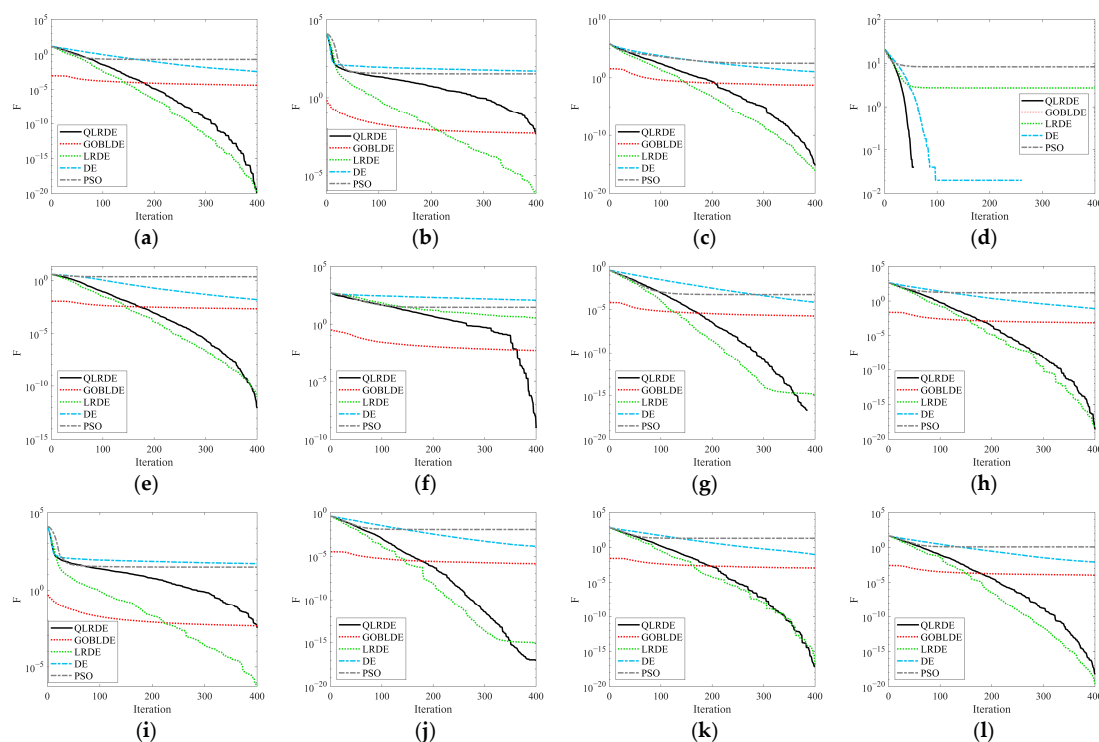


Figure 1. Comparison of evolution curves on test functions with $D = 20$. (a) Sphere function F_1 ; (b) Shifted Schwefel's problem F_2 ; (c) Shifted rotated high conditioned elliptic function F_3 ; (d) Shifted step function F_4 ; (e) Shifted rotated Ackley's function with global optimum on bounds F_5 ; (f) Shifted Rastrigin's function F_6 ; (g) Shifted rotated Griewank's function without bounds F_7 ; (h) Axis parallel hyper ellipsoid F_8 ; (i) Rotated hyper-ellipsoid function F_9 ; (j) Schaffer function F_{10} ; (k) Two-dimensional de Jong function F_{11} ; (l) Multimodal Rastrigin-Sankoff hybrid function F_{12} .

Table 2. Comparison results on test functions with $D = 20$.

F	Result	QLRDE	LRDE	GOBLDE	DE	PSO	F	Result	QLRDE	LRDE	GOBLDE	DE	PSO
F_1	best	4.69×10^{-30}	1.34×10^{-25}	3.66×10^{-10}	2.22×10^{-4}	3.72×10^{-6}	F_7	best	0	0	1.15×10^{-10}	6.32×10^{-6}	1.20×10^{-7}
	worst	5.69×10^{-19}	1.62×10^{-19}	3.82×10^{-4}	5.15×10^{-2}	1.00×10^0		worst	0	1.19×10^{-14}	3.09×10^{-5}	6.40×10^{-4}	2.79×10^{-2}
	median	3.34×10^{-23}	1.39×10^{-22}	7.13×10^{-6}	8.47×10^{-4}	1.26×10^{-4}		median	0	8.88×10^{-16}	2.02×10^{-7}	3.41×10^{-5}	1.42×10^{-6}
	mean	1.25×10^{-20}	1.13×10^{-20}	3.98×10^{-5}	3.59×10^{-3}	2.00×10^{-1}		mean	0	1.42×10^{-15}	1.88×10^{-6}	7.32×10^{-5}	5.62×10^{-4}
	std	8.03×10^{-20}	3.11×10^{-20}	7.72×10^{-5}	9.011×10^{-3}	4.04×10^{-1}		std	0	2.06×10^{-15}	5.06×10^{-6}	1.15×10^{-4}	3.95×10^{-3}
F_2	best	6.48×10^{-9}	5.83×10^{-10}	2.39×10^{-8}	1.40×10^1	1.99×10^{-1}	F_8	best	4.46×10^{-27}	5.06×10^{-24}	8.91×10^{-11}	3.25×10^{-3}	6.28×10^{-3}
	worst	1.28×10^{-1}	1.45×10^{-5}	4.13×10^{-2}	1.32×10^2	2.92×10^2		worst	1.23×10^{-17}	1.32×10^{-17}	7.98×10^{-3}	6.54×10^{-1}	5.50×10^1
	median	2.77×10^{-4}	6.18×10^{-8}	7.10×10^{-4}	4.98×10^1	1.51×10^1		median	1.26×10^{-21}	3.93×10^{-21}	8.47×10^{-5}	3.01×10^{-2}	1.36×10^1
	mean	4.33×10^{-3}	6.81×10^{-7}	4.96×10^{-3}	5.15×10^1	3.40×10^1		mean	2.73×10^{-19}	5.07×10^{-19}	7.37×10^{-4}	7.89×10^{-2}	1.57×10^1
	std	1.84×10^{-2}	2.27×10^{-6}	9.73×10^{-3}	2.12×10^1	5.91×10^1		std	1.73×10^{-18}	1.97×10^{-18}	1.47×10^{-3}	1.30×10^{-1}	1.16×10^1
F_3	best	1.25×10^{-25}	3.29×10^{-23}	3.22×10^{-7}	7.38×10^{-1}	1.03×10^2	F_9	best	2.73×10^{-8}	7.71×10^{-11}	4.66×10^{-6}	1.40×10^1	2.41×10^{-1}
	worst	2.76×10^{-14}	1.17×10^{-15}	4.64×10^{-1}	3.04×10^2	8.50×10^2		worst	1.03×10^{-1}	6.65×10^{-6}	4.35×10^{-2}	1.72×10^2	1.56×10^2
	median	3.28×10^{-19}	7.28×10^{-19}	1.52×10^{-2}	4.01×10^0	2.78×10^2		median	1.82×10^{-4}	3.67×10^{-8}	1.64×10^{-3}	4.50×10^1	1.93×10^1
	mean	6.32×10^{-16}	4.57×10^{-17}	5.27×10^{-2}	1.13×10^1	3.14×10^2		mean	3.22×10^{-3}	5.37×10^{-7}	4.43×10^{-3}	5.11×10^1	3.05×10^1
	std	3.91×10^{-15}	1.78×10^{-16}	9.36×10^{-2}	4.25×10^1	1.56×10^2		std	1.46×10^{-2}	1.39×10^{-6}	7.81×10^{-3}	2.47×10^1	3.73×10^1
F_4	best	0	0	0	0	4.00×10^0	F_{10}	best	0	0	1.95×10^{-11}	7.19×10^{-6}	4.63×10^{-7}
	worst	0	7.00×10^0	0	0	1.70×10^1		worst	1.67×10^{-16}	3.94×10^{-15}	1.31×10^{-5}	1.23×10^{-3}	4.83×10^{-2}
	median	0	3.00×10^0	0	0	8.00×10^0		median	0	5.83×10^{-16}	9.47×10^{-8}	3.61×10^{-5}	1.17×10^{-4}
	mean	0	2.72×10^0	0	0	8.30×10^0		mean	9.99×10^{-18}	8.63×10^{-16}	1.41×10^{-6}	1.37×10^{-4}	1.15×10^{-2}
	std	0	1.60×10^0	0	0	2.62×10^0		std	3.49×10^{-17}	9.49×10^{-16}	2.89×10^{-6}	2.15×10^{-4}	1.45×10^{-2}
F_5	best	4.44×10^{-16}	7.55×10^{-15}	3.27×10^{-5}	5.32×10^{-3}	1.60×10^0	F_{11}	best	9.42×10^{-26}	3.55×10^{-25}	2.13×10^{-11}	1.03×10^{-2}	6.11×10^{-4}
	worst	3.03×10^{-11}	2.64×10^{-10}	1.49×10^{-2}	6.71×10^{-2}	2.55×10^0		worst	3.20×10^{-16}	3.26×10^{-16}	1.85×10^{-2}	6.37×10^{-1}	1.12×10^2
	median	3.60×10^{-14}	9.65×10^{-13}	1.23×10^{-3}	1.13×10^{-2}	2.18×10^0		median	2.43×10^{-21}	2.75×10^{-20}	2.29×10^{-4}	4.03×10^{-2}	2.29×10^{-2}
	mean	8.33×10^{-13}	1.04×10^{-11}	2.04×10^{-3}	1.50×10^{-2}	2.15×10^0		mean	6.78×10^{-18}	1.35×10^{-17}	1.20×10^{-3}	1.00×10^{-1}	2.10×10^1
	std	4.30×10^{-12}	3.99×10^{-11}	2.71×10^{-3}	1.20×10^{-2}	1.90×10^{-1}		std	4.52×10^{-17}	5.97×10^{-17}	3.06×10^{-3}	1.47×10^{-1}	3.04×10^1
F_6	best	0	0	1.75×10^{-7}	3.98×10^1	2.20×10^1	F_{12}	best	2.03×10^{-27}	8.67×10^{-26}	1.47×10^{-9}	7.50×10^{-4}	6.04×10^{-5}
	worst	3.41×10^{-8}	5.98×10^1	4.93×10^{-2}	1.88×10^2	3.61×10^1		worst	1.37×10^{-17}	5.95×10^{-19}	2.30×10^{-3}	9.09×10^{-2}	7.35×10^0
	median	0	1.88×10^{-6}	9.41×10^{-4}	1.20×10^2	2.89×10^1		median	1.64×10^{-22}	4.92×10^{-22}	1.92×10^{-5}	3.56×10^{-3}	1.09×10^{-3}
	mean	9.62×10^{-10}	3.28×10^0	5.20×10^{-3}	1.18×10^2	2.88×10^1		mean	5.39×10^{-19}	2.29×10^{-20}	1.11×10^{-4}	8.29×10^{-3}	1.24×10^0
	std	5.17×10^{-9}	1.01×10^1	1.13×10^{-2}	3.36×10^1	3.93×10^0		std	2.60×10^{-18}	8.67×10^{-20}	3.50×10^{-4}	1.61×10^{-2}	1.82×10^0

Table 3. Comparison results on test functions with $D = 30$.

F	Result	QLRDE	LRDE	GOBLDE	DE	PSO	F	Result	QLRDE	LRDE	GOBLDE	DE	PSO
F_1	best	3.25×10^{-25}	1.90×10^{-25}	2.65×10^{-8}	1.33×10^{-2}	2.09×10^{-3}	F_7	best	0	0	1.23×10^{-11}	3.33×10^{-4}	2.33×10^{-5}
	worst	1.85×10^{-17}	3.10×10^{-17}	1.08×10^{-3}	2.18×10^{-1}	3.00×10^0		worst	1.11×10^{-16}	1.09×10^{-14}	2.16×10^{-5}	4.93×10^{-3}	5.88×10^{-2}
	median	6.51×10^{-22}	6.93×10^{-21}	4.26×10^{-5}	5.36×10^{-2}	7.80×10^{-3}		median	0	5.55×10^{-16}	3.48×10^{-7}	1.27×10^{-3}	1.98×10^{-4}
	mean	6.45×10^{-19}	8.36×10^{-19}	1.26×10^{-4}	7.27×10^{-2}	4.28×10^{-1}		mean	2.22×10^{-18}	1.37×10^{-15}	2.07×10^{-6}	1.56×10^{-3}	1.12×10^{-2}
	std	2.82×10^{-18}	4.44×10^{-18}	2.26×10^{-4}	4.97×10^{-2}	6.43×10^{-1}		std	1.57×10^{-17}	2.11×10^{-15}	3.99×10^{-6}	1.07×10^{-3}	1.59×10^{-2}
F_2	best	1.29×10^{-9}	4.11×10^{-9}	4.27×10^{-6}	8.29×10^1	2.55×10^0	F_8	best	3.94×10^{-25}	6.66×10^{-23}	1.96×10^{-7}	5.73×10^{-1}	2.00×10^0
	worst	2.34×10^0	2.14×10^{-4}	5.49×10^{-1}	4.42×10^2	1.26×10^3		worst	4.81×10^{-16}	7.52×10^{-17}	4.96×10^{-2}	1.31×10^1	1.33×10^2
	median	2.56×10^{-3}	9.09×10^{-7}	7.63×10^{-3}	1.76×10^2	1.22×10^2		median	8.82×10^{-20}	1.32×10^{-19}	4.85×10^{-4}	2.97×10^0	4.02×10^1
	mean	1.19×10^{-1}	1.15×10^{-5}	3.13×10^{-2}	1.97×10^2	1.92×10^2		mean	1.08×10^{-17}	4.88×10^{-18}	4.03×10^{-3}	3.49×10^0	4.77×10^1
	std	4.28×10^{-1}	3.46×10^{-5}	8.08×10^{-2}	7.54×10^1	2.39×10^2		std	6.80×10^{-17}	1.48×10^{-17}	8.48×10^{-3}	2.45×10^0	2.79×10^1
F_3	best	1.02×10^{-21}	1.78×10^{-22}	3.20×10^{-7}	6.41×10^1	6.84×10^2	F_9	best	4.57×10^{-7}	3.81×10^{-9}	9.35×10^{-6}	9.47×10^1	4.21×10^0
	worst	2.30×10^{-13}	2.66×10^{-14}	6.03×10^0	9.06×10^2	6.82×10^3		worst	1.73×10^0	1.63×10^{-5}	2.28×10^{-1}	3.76×10^2	5.23×10^2
	median	3.95×10^{-18}	6.41×10^{-18}	6.10×10^{-2}	1.26×10^2	1.69×10^3		median	1.44×10^{-3}	5.72×10^{-7}	4.32×10^{-3}	1.99×10^2	1.17×10^2
	mean	9.24×10^{-15}	1.15×10^{-15}	3.55×10^{-1}	1.72×10^2	1.96×10^3		mean	6.64×10^{-2}	2.30×10^{-6}	1.74×10^{-2}	2.12×10^2	1.63×10^2
	std	4.07×10^{-14}	4.42×10^{-15}	9.63×10^{-1}	1.43×10^2	1.09×10^3		std	2.65×10^{-1}	3.97×10^{-6}	4.38×10^{-2}	6.89×10^1	1.41×10^2
F_4	best	0	2.00×10^0	0	0	5.00×10^0	F_{10}	best	0	0	5.13×10^{-10}	2.76×10^{-4}	5.70×10^{-5}
	worst	0	1.20×10^1	0	1.00×10^0	2.60×10^1		worst	5.55×10^{-17}	4.50×10^{-15}	6.62×10^{-5}	8.44×10^{-3}	4.79×10^{-2}
	median	0	6.00×10^0	0	0	1.35×10^1		median	0	7.77×10^{-16}	8.20×10^{-7}	1.84×10^{-3}	1.60×10^{-2}
	mean	0	6.40×10^0	0	6.00×10^{-2}	1.42×10^1		mean	2.22×10^{-18}	9.80×10^{-16}	4.61×10^{-6}	1.90×10^{-3}	1.48×10^{-2}
	std	0	2.40×10^0	0	2.40×10^{-1}	3.91×10^0		std	1.10×10^{-17}	1.07×10^{-15}	1.16×10^{-5}	1.43×10^{-3}	1.49×10^{-2}
F_5	best	4.00×10^{-15}	2.89×10^{-14}	3.35×10^{-5}	4.11×10^{-2}	1.88×10^0	F_{11}	best	2.38×10^{-24}	3.00×10^{-23}	7.92×10^{-7}	7.04×10^{-1}	4.72×10^{-2}
	worst	4.31×10^{-11}	1.13×10^{-10}	2.10×10^{-2}	3.96×10^{-1}	2.54×10^0		worst	6.97×10^{-16}	1.03×10^{-15}	5.67×10^{-2}	9.90×10^0	1.13×10^2
	median	2.47×10^{-13}	7.43×10^{-12}	1.66×10^{-3}	8.31×10^{-2}	2.24×10^0		median	4.35×10^{-20}	1.87×10^{-19}	1.39×10^{-3}	2.69×10^0	3.77×10^1
	mean	2.67×10^{-12}	2.31×10^{-11}	3.48×10^{-3}	1.09×10^{-1}	2.23×10^0		mean	3.10×10^{-17}	3.94×10^{-17}	6.31×10^{-3}	3.18×10^0	4.12×10^1
	std	7.00×10^{-12}	2.94×10^{-11}	4.39×10^{-3}	8.40×10^{-2}	1.41×10^{-1}		std	1.36×10^{-16}	1.56×10^{-16}	1.07×10^{-2}	2.16×10^0	2.99×10^1
F_6	best	0	0	3.38×10^{-5}	8.52×10^1	3.84×10^1	F_{12}	best	1.78×10^{-26}	8.99×10^{-26}	1.62×10^{-10}	4.39×10^{-2}	8.96×10^{-3}
	worst	1.81×10^{-11}	9.33×10^1	7.41×10^{-1}	3.30×10^2	7.81×10^2		worst	5.16×10^{-17}	2.43×10^{-16}	2.67×10^{-2}	6.55×10^{-1}	8.48×10^0
	median	0	6.22×10^{-15}	8.01×10^{-3}	2.55×10^2	4.86×10^1		median	8.05×10^{-21}	5.82×10^{-21}	5.11×10^{-5}	1.40×10^{-1}	2.16×10^0
	mean	5.47×10^{-13}	3.70×10^0	5.85×10^{-2}	2.40×10^2	1.90×10^2		mean	3.68×10^{-18}	6.36×10^{-18}	1.06×10^{-3}	1.82×10^{-1}	2.90×10^0
	std	2.60×10^{-12}	1.46×10^1	1.37×10^{-1}	5.56×10^1	2.76×10^2		std	1.13×10^{-17}	3.46×10^{-17}	3.92×10^{-3}	1.25×10^{-1}	2.12×10^0

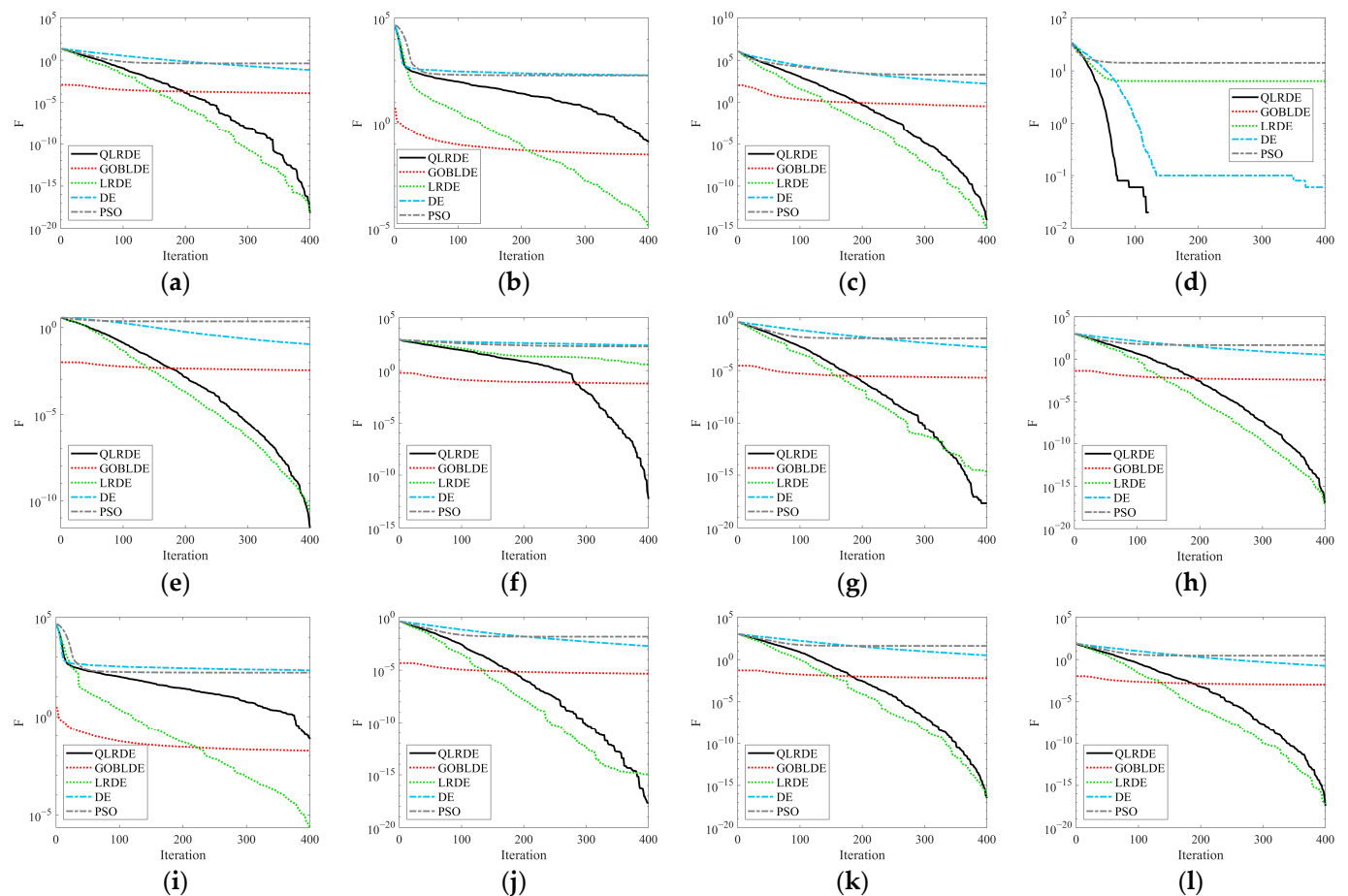


Figure 2. Comparison of evolution curves on test functions with $D = 30$. (a) Sphere function F_1 ; (b) Shifted Schwefel's problem F_2 ; (c) Shifted rotated high conditioned elliptic function F_3 ; (d) Shifted step function F_4 ; (e) Shifted rotated Ackley's function with global optimum on bounds F_5 ; (f) Shifted Rastrigin's function F_6 ; (g) Shifted rotated Griewank's function without bounds F_7 ; (h) Axis parallel hyper ellipsoid function F_8 ; (i) Rotated hyper-ellipsoid function F_9 ; (j) Schaffer function F_{10} ; (k) Two-dimensional de Jong function F_{11} ; (l) Multimodal Rastrigin-Sankoff hybrid function F_{12} .

The data from tables and iterative curves indicate that LRDE outperforms standard DE, PSO, and GOBLDE in all metrics, validating the effectiveness of the Loser Reverse-Learning Mechanism combined with DE. The LRLM significantly improves algorithm efficiency and the quality of optimal solutions, enabling LRDE to effectively escape local optima and approach the global optimum more closely. While GOBLDE performs well in some aspects, QLRDE generally exhibits superior global search ability and stability.

In conclusion, the analysis of the data presented in tables and figures indicates that QLRDE outperforms four other algorithms across twelve benchmark functions in both $D = 20$ and $D = 30$ dimensions. QLRDE demonstrates outstanding global search ability and consistently achieves excellent results in finding the minimum values. It effectively mitigates the issues of local optima, premature convergence, and slow convergence speed in solving continuous function optimization problems, thereby significantly improving algorithm performance. Moreover, QLRDE displays commendable convergence behavior, stability, and reliability throughout the optimization process. Therefore, QLRDE stands as a highly recommended choice for solving high-dimensional continuous optimization problems.

4.2.2. Analysis of ANOVA for Values

Table 4 shows the one-way ANOVA of the best values for five different algorithms across 12 test functions with $D = 20$. The results indicate that the between-group variance

SS is 672.04 with a degree of freedom (df) of 4, and the mean square (MS) is 168.0097. The within-group variance SS is 11,752.00 with a df of 55, and the MS is 213.6758. The total SS is 12,424.00 with a df of 59. The F -value is 0.7863 and the p -value is 0.5390, which is above the 0.05 significance level. This leads to the conclusion that there is no significant difference. Figure 3a illustrates the differences between the DE and PSO algorithms compared to others, showing considerable fluctuations in most cases, while other algorithms demonstrate more stability in certain test functions.

Table 4. The one-way ANOVA for the best values with $D = 20$.

Source	SS	df	MS	F	$P(F)$
Group	672.04	4	168.0097	0.7863	0.5390
Error	11,752.00	55	213.6758		
Total	12,424.00	59			

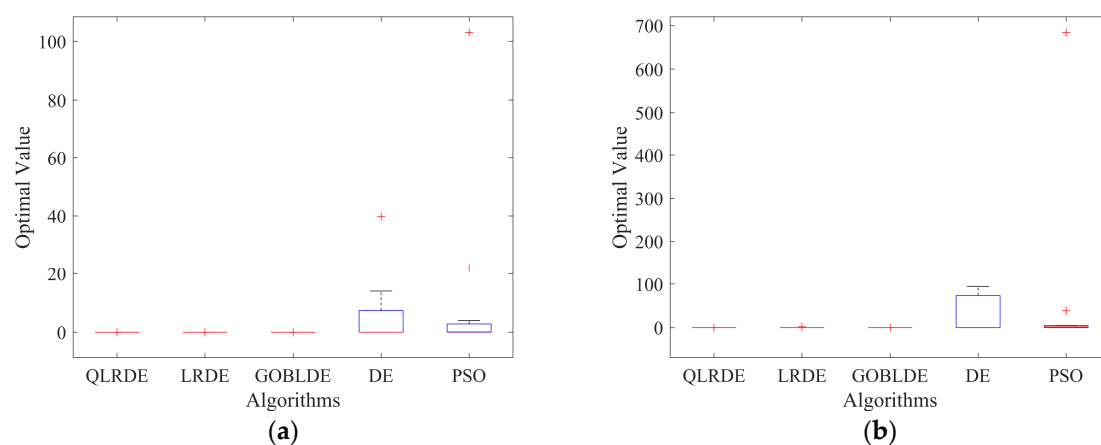


Figure 3. (a) One-way ANOVA box plots on the best values for different test functions with $D = 20$; (b) one-way ANOVA box plots on the best values for different test functions with $D = 30$.

Table 5 presents the one-way ANOVA of the mean values for eight different algorithms across 12 test functions with $D = 30$. The F -value is 0.9636 and the p -value is 0.4349, which is above the 0.05 significance level. Figure 3b shows that QLRDE is the most stable algorithm with the least fluctuations and the best performance, indicating greater reliability and effectiveness in most cases.

Table 5. The one-way ANOVA for the best values with $D = 30$.

Source	SS	df	MS	F	$P(F)$
Group	31,275.00	4	7818.80	0.9636	0.4349
Error	446,300.00	55	8114.60		
Total	477,580.00	59			

In summary, the analysis of these data concludes that QLRDE exhibits fast convergence, stability during operation, and a lower likelihood of becoming trapped in local optima. In most cases, it finds solutions that are closer to the true global optimum.

5. QLRDE for UAV Path Planning

This section presents on a path planning solution based on the QLRDE algorithm. The solution begins by employing B-Spline curves to transform the continuous path planning problem into an optimization problem with a finite number of control points, thereby reducing the dimensionality of the decision variables. A comprehensive cost function that

incorporates multiple objectives is established, considering path length, threat minimization, flying height, turning angle, and terrain constraints, thus modeling the path planning as an optimization problem. The QLRDE algorithm is then employed to optimize the path, ultimately generating a smooth flight path that satisfies dynamic constraints while minimizing the total cost.

5.1. Path Representation Based on B-Spline Curve

The UAV path comprises N discrete waypoints (excluding the fixed start and target), each waypoint p_k , $k = 1, \dots, N$, is defined by three coordinates (xp_k, yp_k, zp_k) , thereby resulting in a $3N$ -dimensional decision space. For dimensionality reduction and smooth path generation, B-Splined strategy [31] is employed to derive the flight trajectory from control points, ultimately represent the target UAV path.

UAV path defined by discrete waypoints $\{p_0, p_1, \dots, p_N, p_{N+1}\}$ with the coordinates (xp_k, yp_k, zp_k) for each waypoint, corresponds to a set of control points $\{w_0, w_1, \dots, w_n, w_{n+1}\}$ with the coordinates (xc_i, yc_i, zc_i) , $i = 1, \dots, n$. The start (p_0, w_0) and goal (p_{N+1}, w_{n+1}) are prescribed. Path planning is equivalent to solving for the n free control points that, along with the fixed endpoints, produce the desired B-Spline curve-based path.

5.2. Model of Cost Function

Path planning is formulated as an optimization task, and minimizes a comprehensive cost function J , which is the core criterion to evaluate path quality [27]. As detailed in Section 5.1, given the UAV path represented by waypoints $\{p_0, p_1, \dots, p_N, p_{N+1}\}$ with coordinates (xp_k, yp_k, zp_k) , $k = 0, \dots, N + 1$. The cost function is as follows:

$$\min J = f_{LC} + f_{TC} + f_{NFC} + f_{AC} + f_{LMC} + f_{VMC} \quad (17)$$

where total cost J aggregates five key terms: length cost f_{LC} , threat cost f_{TC} , no-fly zone cost f_{NFC} , altitude cost f_{AC} , lateral maneuvering cost f_{LMC} , and vertical maneuvering cost f_{VMC} .

The length cost f_{LC} is given by the accumulated sum of all consecutive segment lengths along the path

$$f_{LC} = \sum_{k=0}^N l_k \quad (18)$$

with

$$l_k = \sqrt{(xp_{i+1} - xp_i)^2 + (yp_{i+1} - yp_i)^2 + (zp_{i+1} - zp_i)^2}$$

Designed to penalize proximity to threats, the threat cost f_{TC} accounts for the cumulative threat along the path, factoring in segment lengths and localized threat probabilities as follows

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

with

$$P_{j,k} = \begin{cases} 0 & , \text{if } d_{j,k} > R_{\max,j} \\ \frac{R_{\max,j}^4}{d_{j,k}^4 + R_{\max,j}^4} & , \text{otherwise} \end{cases} \quad (19)$$

where $P_{j,k}$, $R_{\max,j}$, and $d_{j,k}$ denote, respectively, the threat probability for segment $p_k p_{k+1}$ from the j -th threat, the threat's maximum effective radius, and the segment's distance to the threat center.

The UAV should avoid entering some no-fly zones, such as harsh climate zones, unknown zones, and so on. The no-fly zones cost function f_{NFC} is calculated as follows:

$$f_{\text{NFC}} = \sum_{k=1}^N L_{\text{in},k} \quad (20)$$

where N is the number of no-fly zones and $L_{\text{in},k}$ is the length of the UAV path that is inside the k -th no-fly zones.

The altitude cost f_{AC} is designed to incentivize low-altitude flight, thereby exploiting terrain masking, and to penalize any path colliding with the terrain, as follows:

$$f_{\text{AC}} = \sum_{k=1}^N f_k^{\text{AC}} \quad (21)$$

with

$$f_k^{\text{AC}} = \begin{cases} C, & \text{if } zp_k < H_{\text{map}}(xp_k, yp_k) + H_{\text{min}} \\ zp_k, & \text{otherwise} \end{cases}$$

where $H_{\text{map}}(xp_k, yp_k)$ is the terrain elevation at coordinates (xp_k, yp_k) , H_{min} is the minimum allowable flight altitude, and C is the penalty coefficient.

The lateral maneuvering cost f_{LMC} penalizes excessive turning by summing penalties at waypoints where the turning angle exceeds the allowable limit, as follows:

$$f_{\text{LMC}} = \sum_{k=1}^N f_k^{\text{LMC}} \quad (22)$$

with

$$f_k^{\text{LMC}} = \begin{cases} 0, & \text{if } \phi_k \leq \phi_k^{\text{max}} \\ C, & \text{otherwise} \end{cases}$$

where turning angle ϕ_k at p_k and penalty C are as defined. The allowable turning is constrained by a maximum angle ϕ_k^{max} , which is determined by the UAV's dynamic limits

$$\phi_k^{\text{max}} = \frac{9.8n_{\text{max}}}{V^2} \cdot \sqrt{(xp_{k+1} - xp_k)^2 + (yp_{k+1} - yp_k)^2} \quad (23)$$

where V is the velocity and n_{max} is the maximum allowable lateral load.

The vertical maneuvering cost f_{VMC} can be computed as the sum of penalties imposed at waypoints that violate the allowable climb/glide slope, as follows:

$$f_{\text{VMC}} = \sum_{k=1}^N f_k^{\text{VMC}} \quad (24)$$

with

$$f_k^{\text{VMC}} = \begin{cases} 0, & \text{if } \beta_k \leq s_k \leq \alpha_k \\ C, & \text{otherwise} \end{cases}$$

where the maximum climbing slope α_k , minimum gliding slope β_k , and the instantaneous path slope s_k at waypoint p_k are calculated according to the aerodynamic model [27].

5.3. Path Planning Using QLRDE

The core of applying the QLRDE algorithm to UAV path planning lies in effectively mapping the path optimization problem onto the evolutionary search framework of QLRDE. This section details this mapping process, including the encoding of B-Spline control points into QLRDE individuals, the design of the cost function as the fitness evaluation criterion, and the specific procedural integration.

In the B-Spline-based path representation described in Section 5.1, a smooth flight path is determined by a sequence of n free-to-move control points, $W = \{w_1, w_2, \dots, w_n\}$, where each control point $w_i = (x_i, y_i, z_i)$ has 3D coordinates. Together with the fixed start point w_0 and target point w_{n+1} , these points fully define the path.

In the QLRDE algorithm, each individual in the population represents a candidate solution. Specifically, an individual x in the QLRDE population is encoded as the concatenation of all coordinates of the n free control points:

$$x = [x_1, y_1, z_1, x_2, y_2, z_2, \dots, x_n, y_n, z_n] \quad (25)$$

Thus, the problem dimension is $D = 3n$. Each individual x corresponds to a unique UAV path generated by the B-Spline curve construction formulas. The population of QLRDE, therefore, explores a space of potential paths by evolving these control points.

The quality of a path, represented by an individual x , is evaluated by the cost function $J(x)$ defined in Equation (17). This function J serves as the fitness function in the QLRDE algorithm, guiding the evolutionary search towards safer and more efficient paths.

During greedy selection, each trial path is directly compared with its parent based on the total cost $J(u_i)$ versus $J(x_i)$, which establishes a steady selection pressure toward lower-cost paths. The penalty mechanism embedded in the cost function plays a crucial role in steering the search direction: paths that violate hard constraints (e.g., exceedance of turning or climb-angle limits) receive a heavy penalty of order $C = 10^3$, raising their total cost far above that of any feasible path and ensuring their rapid elimination. For violations of soft constraints (e.g., proximity to threats), a moderate penalty on the order of 10^1 to 10^2 is applied, allowing some exploration near constraint boundaries while gradually guiding the population toward feasible regions. This differentiated penalty strategy enables the algorithm to explore a broad space that includes slightly infeasible solutions in early generations, yet converge strictly to fully feasible and optimized paths in later stages.

For the adaptive parameter adjust mechanism, the update of an individual's control parameters F_i and CR_i is directly coupled to cost improvement: only when $J(u_i) < J(x_i)$ are the parameters used in that trial considered successful and retained for the next generation via Equation (13); otherwise, the original parameters are kept. This mechanism ensures that search regions in the control-point space that consistently yield cost reductions acquire enhanced local-search capability, while regions that fail to improve are gradually assigned lower search intensity. In particular, individuals that successfully avoid threat zones or terrain obstacles tend to preserve and propagate their parameters, thereby establishing a search bias toward feasibility and low cost within the population.

The process of integrating the QLRDE algorithm for 3D UAV path planning follows a structured iterative procedure, as illustrated in Figure 4. Algorithm 4 summarizes the detailed procedure, highlighting the integration points between path planning and the QLRDE optimizer.

Algorithm 4. QLRDE path planning

- ```

/*Parameter setting*/
1. Set Start point w_0 , target point w_{n+1} , terrain data H_{map} , UAV parameters, n_{max} , V , H_{min} , QLRDE parameters D , g_{max} , M_{pop} , F_{max} , F_{min} , CR_{max} , CR_{min} .
/*Initialization*/
2. Generation $g = 0$
 Randomly initialize population of M_{pop} individuals, x_i , $i = 1, \dots, M_{\text{pop}}$, within
3. the mission boundaries. Each individual x_i represents n control points and
 calculate the costs $J(x_i)$.

```
-

**Algorithm 4.** *Cont.*


---

```

4. Initialize amplification factors F_i and CR_i .
5. While $g < g_{\max}$ do
6. Randomly select $a, b \in [1, M_{\text{pop}}]$.
7. If $\text{rand} < 0.5$ then
8.
$$\begin{cases} F_{\text{new}} = F_a + \text{rand} \cdot (F_b - F_a) \\ CR_{\text{new}} = CR_a + \text{rand} \cdot (1 - CR_a) \end{cases}$$

9. Else
10.
$$\begin{cases} F_{\text{new}} = F_a \\ CR_{\text{new}} = CR_a \end{cases}$$

11. End if
12. For $i = 1: M_{\text{pop}}$ do
13. /*Quantum behavior mutation*/
14. Set adjustment parameters k, x .
15. If $\text{rand} < 0.5$ then
16.
$$v_i = x_{r1} - k \cdot F_{\text{new}} \cdot r_p \cdot \tanh(x \cdot \ln(1/r_x)) \cdot (x_{r2} - x_{r3})$$

17. Else
18.
$$v_i = x_{r1} + k \cdot F_{\text{new}} \cdot r_p \cdot \tanh(x \cdot \ln(1/r_x)) \cdot (x_{r2} - x_{r3})$$

19. End if
20. /*Crossover*/
21. For $j = 1: D$ do
22.
$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } \text{rand} \leq CR_{\text{new}} \text{ or } j = j_{\text{rand}} \\ x_{i,j}, & \text{otherwise} \end{cases}$$

23. End for
24. End for
25. Apply the Loser Reverse-Learning Mechanism as described in Algorithm 2.
26. For $i = 1: M_{\text{pop}}$ do
27. /*Selection*/
28. If $J(u_i) \leq J(x_i)$ then
29. $x_i = u_i$
30. $x_{\text{best}} = u_i$
31. $J_{\text{best}} = J(u_i)$
32. $F_i = F_{\text{new}}$
33. $CR_i = CR_{\text{new}}$
34. Else
35. $x_i = x_i$
36. $x_{\text{best}} = x_i$
37. $J_{\text{best}} = J(x_i)$
38. $F_i = F_i$
39. $CR_i = CR_i$
40. End if
41. End for
42. $g = g + 1$
43. End while
44. Return the J_{best} and optimal path defined by the control points of x_{best} .

```

---

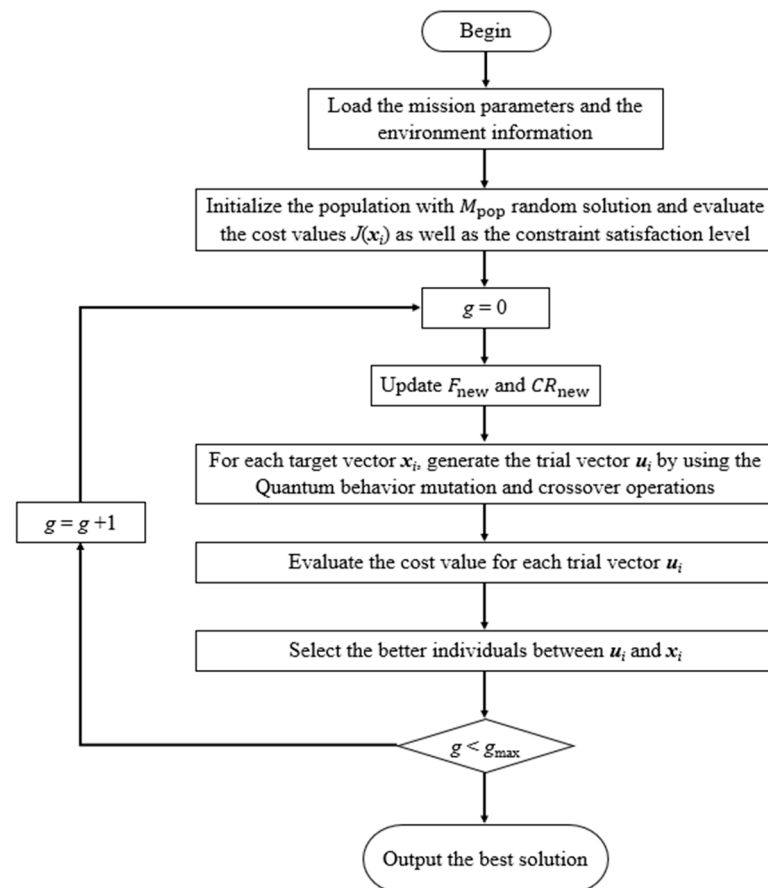


**Figure 4.** Flowchart of QLRDE-based UAV path planning.

#### 5.4. Synergistic Advantages of QLRDE in Path Planning

The unique mechanisms of the QLRDE algorithm confer distinct synergistic advantages for addressing the UAV path planning problem. Primarily, the quantum-behaved mutation strategy, by virtue of its inherent uncertainty, drives individuals to explore distant regions of the search space. This is crucial for escaping local optima frequently induced by complex obstacle fields, thereby enhancing the probability of discovering a globally competitive path. Furthermore, the LRLM actively revitalizes stagnant individuals, effectively preventing premature convergence of the population to a suboptimal path and promoting the exploration of diverse path alternatives around threats and terrain features, thus maintaining critical population diversity. Ultimately, the adaptive parameter adjustment mechanism dynamically fine-tunes the search intensity: encouraging broad exploration during the early evolutionary stages and subsequently focusing on the local refinement of promising paths, achieving a balance between convergence efficiency and robustness.

## 6. Evaluation and Comparison in Solving Path Planning

### 6.1. Simulation Results and Comparison

To evaluate experimental evaluation of the designed QLRDE-based path planner, several comparative simulation experiments are conducted. This section focuses on comparing the QLRDE algorithm with several other algorithms, including the LRDE algorithm, the GOBLDE algorithm, the standard DE algorithm, and the PSO algorithm under two mission cases. A fair comparison was ensured by adopting identical settings for all algorithms, with the maximum iteration count as the termination criterion and a population size of  $M_{pop} = 40$ . The simulation environment consisted of MATLAB-2024a running on a standard personal computer, on which all algorithms were implemented.

Given the inherent stochasticity of swarm intelligence algorithms, performance was evaluated statistically. For each test case, every algorithm was executed independently for 50 runs. The experiments were conducted in a known rectangular mission environment containing predefined terrain and threats. As shown in Table 6, the positions of the start point, target point, threats, and no-fly zones are all represented by 2D planar coordinates. In test cases, the UAV mission area is defined as a square airspace with a side length of 90 km. Threats from anti-aircraft gun, radars, and missiles are all modeled as cylinders of infinite height. For example,  $\{[40, 25], 11\}$  indicates that the center of a ground-based threat is located at coordinates  $[40, 25]$  km, with a threat radius of 11 km. No-fly zones are represented as rectangular cuboids of infinite height. For instance,  $\{[9, 43], [23, 43], [23, 27], [9, 27]\}$  denotes that the four vertices of the rectangular no-fly zone are situated at  $[9, 43]$  km,  $[23, 43]$  km,  $[23, 27]$  km, and  $[9, 27]$  km, respectively. The UAV path was parameterized using  $n = 5$  B-Spline control points, resulting in an optimization problem of dimension  $D = 3n = 15$ . The UAV was configured with the following operational parameters:  $n_{\max} = 5$ ,  $H_{\min} = 20$  m, and  $V = 200$  m/s. All tested algorithms shared a common configuration, where  $M_{\text{pop}} = 40$  and  $g_{\max} = 400$ . Other parameters are set as follows: for LRDE, DE, and GOBLDE,  $F$  is a random number in  $[0.1, 0.9]$ , and  $CR = 0.7$ ; for PSO,  $c_1 = c_2 = 1.4$ ,  $w = 0.9$ ; for QLRDE, both the amplification factor  $F \in [0.1, 0.9]$  and the crossover operator  $CR \in [0.2, 0.9]$  are dynamically adjusted during the optimization process.

**Table 6.** Parameters of the threats in test cases.

| Case   | Start Point | Target Point | Threats and No-Fly Zones | Range                                                                                       |
|--------|-------------|--------------|--------------------------|---------------------------------------------------------------------------------------------|
| Case 1 | [2, 65]     | [87, 21]     | Radar                    | $\{[40, 25], 11\}; \{[20, 57], 9\}$                                                         |
|        |             |              | Missile                  | $\{[42, 72], 12\}; \{[29, 8], 6\}$                                                          |
|        |             |              | Anti-aircraft gun        | $\{[73, 57], 9\}$                                                                           |
|        |             |              | No-fly zone              | $\{[9, 43], [23, 43], [23, 27], [9, 27]\};$<br>$\{[53, 42], [84, 42], [84, 21], [53, 21]\}$ |
|        |             |              |                          |                                                                                             |
| Case 2 | [5, 15]     | [65, 85]     | Radar                    | $\{[46, 23], 12\}; \{[69, 13], 7\}$                                                         |
|        |             |              | Missile                  | $\{[45, 70], 8\}; \{[25, 35], 8\}$                                                          |
|        |             |              | Anti-aircraft gun        | $\{[70, 45], 9\}$                                                                           |
|        |             |              | No-fly zone              | $\{[20, 60], [35, 60], [35, 75], [20, 75]\}$                                                |
|        |             |              |                          |                                                                                             |

Figure 5 shows the best 3D UAV paths in the digital terrain environment obtained by the five algorithms after 50 independent runs for Case 1 and Case 2, respectively, where white cylinders represent threat areas from missiles, radars, and anti-aircraft guns, blue cubes represent no-fly zones.

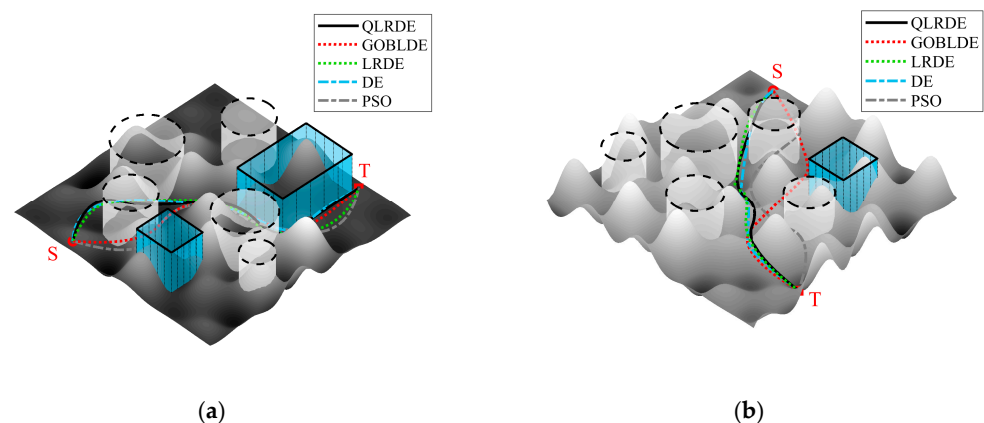


**Figure 5.** (a) Best UAV paths of the five algorithms in 3D environments in Case 1; (b) best UAV paths of the five algorithms in 3D environments in Case 2.

Figure 6 shows the horizontal projections of the best UAV paths from Figure 5 on the contour map, where circular areas represent threats and rectangular areas represent no-fly zones. All algorithms successfully planned safe paths that completely avoid threats.

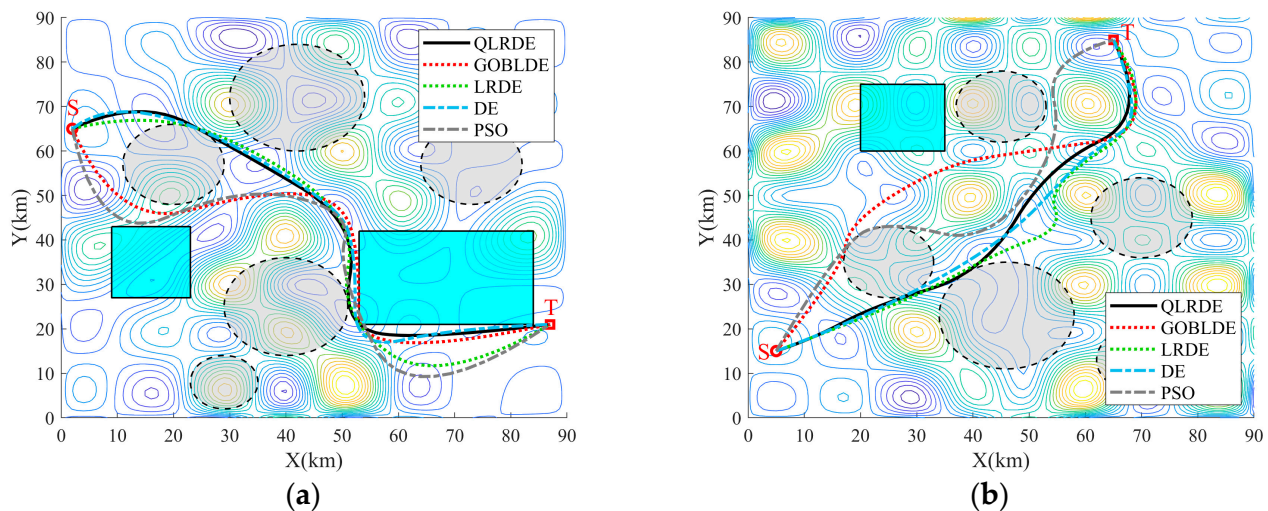


**Figure 6.** (a) Comparison of the horizontal projections in 3D environments for best UAV paths obtained by the five algorithms in Case 1; (b) comparison of the horizontal projections in 3D environments for best UAV paths obtained by the five algorithms in Case 2.

Figure 7 displays the altitude profiles of the best-performing paths for Case 1 and Case 2. The results show that after 50 runs, all five optimization algorithms successfully generated feasible paths avoiding all danger zones. However, the paths generated by the QLRDE algorithm have the shortest length while also exhibiting smaller variations in flight altitude. The QLRDE algorithm effectively combines threat avoidance and path optimization, significantly enhancing the survivability and mission effectiveness of the aircraft. Compared to the paths obtained by LRDE, GOBLDE, DE, and PSO, the QLRDE-generated paths yield superior cost metrics and smaller standard deviations, demonstrating its stronger search capability.

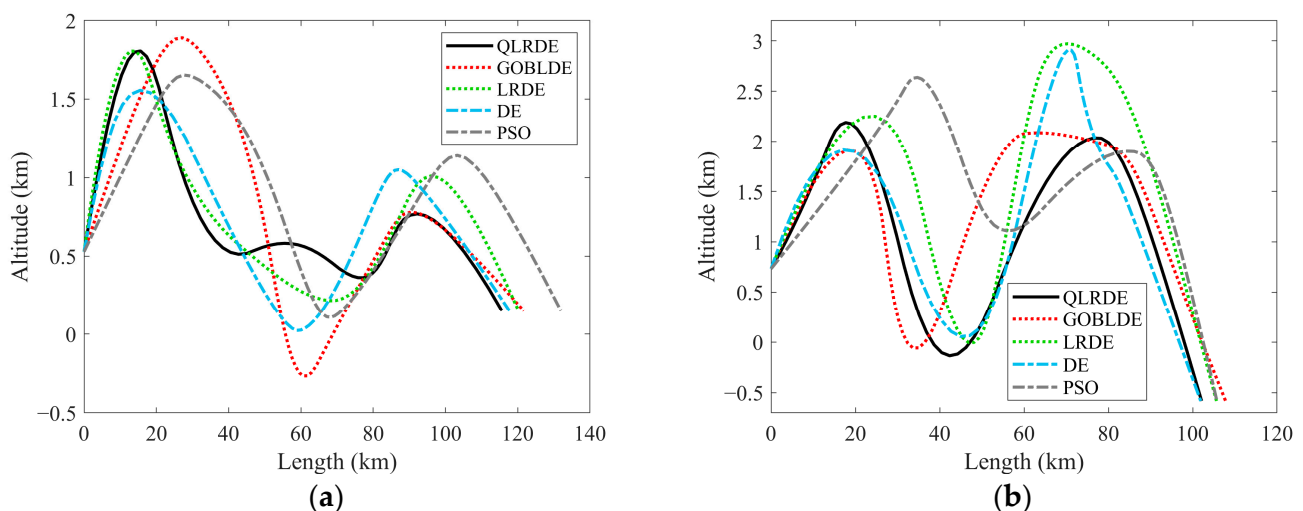


**Figure 7.** (a) Comparison of altitude curves for the best UAV paths of the five algorithms in Case 1; (b) comparison of altitude curves for the best UAV paths of the five algorithms in Case 2.

Algorithm performance can be evaluated using the mean cost and std, indicating its search capability and stability, respectively. The data in Table 7 indicate that QLRDE achieves the minimum values in all five metrics: best, mean, median, worst, and std,

with each optimal value highlighted in bold. This indicates that QLRDE possesses the strongest optimization capability in a statistical sense. Algorithm execution efficiency can be evaluated using the average time (AT, representing the average time the algorithm runs once). The data in Table 7 indicate that QLRDE achieves shorter average time in both test cases, outperforming LRDE and GOBLDE, while remaining competitive with DE and PSO. Figure 8 presents the convergence curves, showing that the QLRDE converges faster and achieves lower cost values than its counterparts.

**Table 7.** Statistical results of the tested algorithms.

| Case   | Result | QLRDE                                | DE                 | LRDE               | GOBLDE             | PSO                |
|--------|--------|--------------------------------------|--------------------|--------------------|--------------------|--------------------|
| Case 1 | best   | <b><math>2.69 \times 10^2</math></b> | $2.72 \times 10^2$ | $2.81 \times 10^2$ | $2.89 \times 10^2$ | $3.35 \times 10^2$ |
|        | worst  | <b><math>3.00 \times 10^3</math></b> | $3.17 \times 10^3$ | $4.60 \times 10^3$ | $3.80 \times 10^3$ | $1.49 \times 10^4$ |
|        | median | <b><math>8.30 \times 10^2</math></b> | $8.72 \times 10^2$ | $9.50 \times 10^2$ | $9.03 \times 10^2$ | $3.17 \times 10^3$ |
|        | mean   | <b><math>9.53 \times 10^2</math></b> | $1.17 \times 10^3$ | $1.12 \times 10^3$ | $1.36 \times 10^3$ | $3.81 \times 10^3$ |
|        | std    | <b><math>5.38 \times 10^2</math></b> | $7.89 \times 10^2$ | $7.62 \times 10^2$ | $1.04 \times 10^3$ | $2.70 \times 10^3$ |
|        | AT(s)  | 93.27                                | 93.63              | 98.91              | 93.61              | <b>92.32</b>       |
| Case 2 | best   | <b><math>2.53 \times 10^2</math></b> | $2.67 \times 10^2$ | $3.29 \times 10^2$ | $2.89 \times 10^2$ | $3.23 \times 10^2$ |
|        | worst  | <b><math>9.79 \times 10^2</math></b> | $4.25 \times 10^3$ | $2.48 \times 10^3$ | $3.82 \times 10^3$ | $8.62 \times 10^3$ |
|        | median | <b><math>5.97 \times 10^2</math></b> | $6.41 \times 10^2$ | $7.71 \times 10^2$ | $6.59 \times 10^2$ | $3.42 \times 10^3$ |
|        | mean   | <b><math>6.06 \times 10^2</math></b> | $8.46 \times 10^2$ | $9.62 \times 10^2$ | $8.92 \times 10^2$ | $3.31 \times 10^3$ |
|        | std    | <b><math>1.84 \times 10^2</math></b> | $7.28 \times 10^2$ | $5.06 \times 10^2$ | $6.54 \times 10^2$ | $1.93 \times 10^3$ |
|        | AT(s)  | 80.57                                | 79.92              | 85.20              | 83.90              | <b>79.05</b>       |

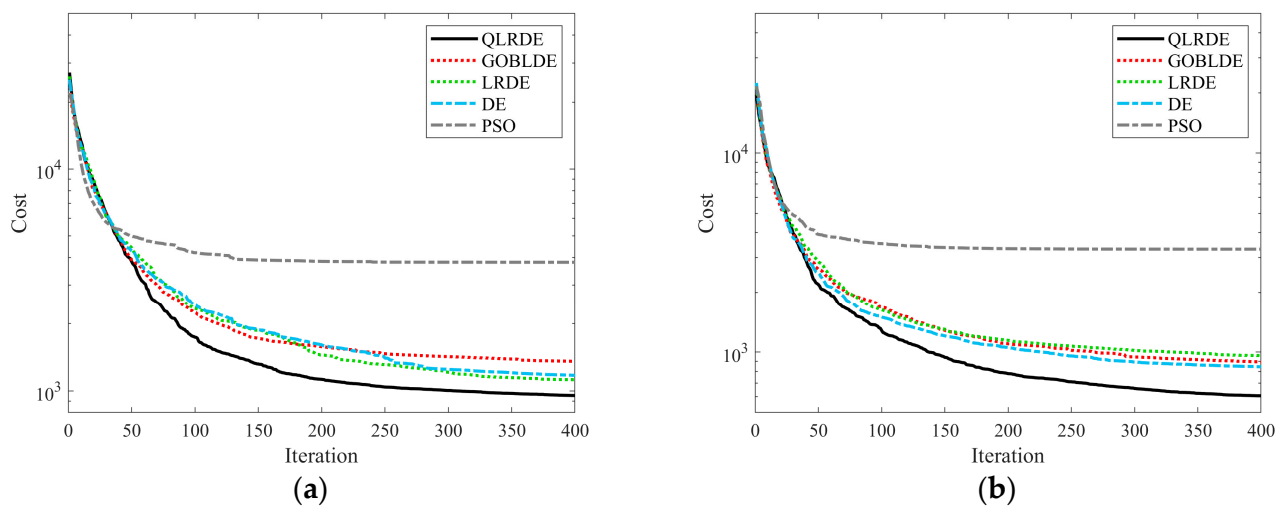


**Figure 8.** (a) Comparison of convergence curves of the average best cost values of the five algorithms in Case 1; (b) comparison of convergence curves of the average best cost values of the five algorithms in Case 2.

To study the distribution characteristics of the solution set, Figure 9 plots the cumulative frequency against the cost value  $J$  for the solution set obtained from independent runs, where the cumulative frequency at any cost threshold  $J$  is as follows:

$$P(J) = \frac{N(J_{\min} \leq J)}{N_{\text{total}}} \quad (26)$$

where the numerator  $N(J_{\min} < J)$  counts the runs where the obtained minimum cost is at most  $J$  and the denominator  $N_{\text{total}} = 50$  is the total number of independent runs. Figure 9 clearly shows that the QLRDE algorithm delivers better performance compared to comparison algorithms. Taking Case 1 as an example, 95% of the solutions found by

QLRDE have a cost below the threshold of 1500. In contrast, GOBLDE and LRDE reach this threshold in only about 75% and 80% of the trials, respectively. In the Case 2, 100% of the cost values obtained by QLRDE are less than 1000 iterations, whereas the GOBLDE, LRDE, and DE algorithms only achieve this in about 80% of the cases. These results show that QLRDE has higher capability compared to other algorithms in solving UAV path planning. In summary, the proposed QLRDE algorithm outperforms GOBLDE, LRDE, DE, and PSO.

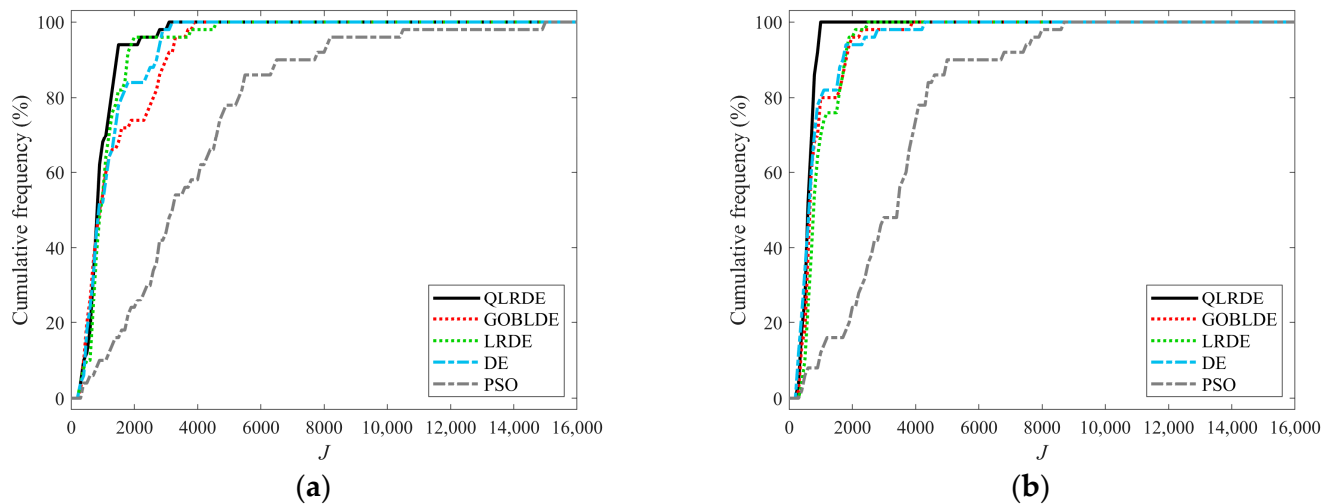


**Figure 9.** (a) Cumulative frequency curves of the five algorithms in Case 1; (b) cumulative frequency curves of the five algorithms in Case 2.

## 6.2. Further Discussion of Algorithm Parameters

In QLRDE algorithm, parameters  $k$ ,  $x$ ,  $o$ , and  $w$  significantly influence the search performance. To ensure the scientific rigor of the experiments, this study first established reasonable ranges for these parameters. Parameter  $k$  controls the amplification factor of differential mutation. If its value is too small, the population may lack exploration capability, whereas an excessively large value may cause oscillations in the solution space. Therefore, this study selected  $k = 3$  and  $k = 4$  within the commonly used range for comparison. Parameters  $x$  and  $o$  primarily affect the shape of the nonlinear adjustment function, influencing the step size distribution during the search process and the ability to escape local optima and their effective intervals are concentrated within  $[0.5, 0.9]$ . Thus, several representative points were selected for experimentation. The parameter  $w$  balances the ratio of mutation and retention. An overly small value may weaken exploration capability, while an overly large value may lead to algorithmic instability. Hence, typical values within  $[0.4, 0.8]$  were chosen for testing. Although this approach does not exhaustively traverse all possible combinations, it adequately covers the main effective intervals, ensuring the reliability of the analytical conclusions.

Under the above settings, this study designed 16 parameter combinations, each independently run 50 times in both Case 1 and Case 2, and recorded the best, worst, median, and mean values, as well as std in Tables 8 and 9, with the top four values per row marked in bold. The experimental results indicate that, regarding parameter  $k$ , the overall mean value for  $k = 4$  is significantly lower than that for  $k = 3$ , demonstrating stronger convergence performance. For the parameter  $w$ , although  $w = 0.4$  achieved favorable mean values in some combinations, nearly all the best and second-best results corresponded to  $w = 0.8$ , indicating that  $w = 0.8$  exhibits greater stability when combined with  $k = 4$ . In terms of the nonlinear coefficients, the combination  $x = 0.7$  and  $o = 0.9$  achieved the smallest mean value in both Case 1 and Case 2, representing the global optimum. It also demonstrated greater stability and better robustness across different experiments.

**Table 8.** The statistical results of different parameter combinations in Case 1.

|        |                    |                                      |                                      |                                      |                                      |                                      |                                      |                                      |
|--------|--------------------|--------------------------------------|--------------------------------------|--------------------------------------|--------------------------------------|--------------------------------------|--------------------------------------|--------------------------------------|
| $k$    | 4                  | 4                                    | 4                                    | 4                                    | 3                                    | 3                                    | 3                                    | 3                                    |
| $x$    | 0.6                | 0.6                                  | 0.6                                  | 0.6                                  | 0.5                                  | 0.5                                  | 0.5                                  | 0.5                                  |
| $o$    | 0.6                | 0.7                                  | 0.8                                  | 0.9                                  | 0.7                                  | 0.6                                  | 0.8                                  | 0.9                                  |
| $w$    | 0.8                | 0.8                                  | 0.4                                  | 0.8                                  | 0.8                                  | 0.8                                  | 0.4                                  | 0.8                                  |
| best   | $3.17 \times 10^2$ | $3.15 \times 10^2$                   | $3.19 \times 10^2$                   | <b><math>3.01 \times 10^2</math></b> | $3.03 \times 10^2$                   | $3.03 \times 10^2$                   | <b><math>2.53 \times 10^2</math></b> | $3.20 \times 10^2$                   |
| worst  | $2.16 \times 10^3$ | $1.73 \times 10^3$                   | <b><math>1.42 \times 10^3</math></b> | $3.18 \times 10^3$                   | <b><math>1.51 \times 10^3</math></b> | $3.22 \times 10^3$                   | $2.57 \times 10^3$                   | $2.91 \times 10^3$                   |
| median | $3.94 \times 10^2$ | <b><math>3.64 \times 10^2</math></b> | $3.71 \times 10^2$                   | <b><math>3.62 \times 10^2</math></b> | $3.84 \times 10^2$                   | $3.68 \times 10^2$                   | $3.97 \times 10^2$                   | $3.81 \times 10^2$                   |
| mean   | $5.08 \times 10^2$ | $4.55 \times 10^2$                   | <b><math>4.36 \times 10^2</math></b> | $5.09 \times 10^2$                   | $4.51 \times 10^2$                   | $5.42 \times 10^2$                   | $6.31 \times 10^2$                   | $4.77 \times 10^2$                   |
| std    | $3.37 \times 10^2$ | $2.83 \times 10^2$                   | <b><math>2.35 \times 10^2</math></b> | $4.74 \times 10^2$                   | <b><math>2.33 \times 10^2</math></b> | $5.10 \times 10^2$                   | $5.14 \times 10^2$                   | $3.87 \times 10^2$                   |
| $k$    | 4                  | 4                                    | 4                                    | 4                                    | 4                                    | 4                                    | 4                                    | 4                                    |
| $x$    | 0.7                | 0.7                                  | 0.7                                  | 0.7                                  | 0.8                                  | 0.8                                  | 0.8                                  | 0.8                                  |
| $o$    | 0.6                | 0.8                                  | 0.7                                  | 0.9                                  | 0.9                                  | 0.6                                  | 0.8                                  | 0.7                                  |
| $w$    | 0.8                | 0.4                                  | 0.8                                  | 0.8                                  | 0.8                                  | 0.8                                  | 0.4                                  | 0.8                                  |
| best   | $3.10 \times 10^2$ | $3.07 \times 10^2$                   | $3.13 \times 10^2$                   | <b><math>2.66 \times 10^2</math></b> | $3.10 \times 10^2$                   | $3.14 \times 10^2$                   | $3.36 \times 10^2$                   | <b><math>2.60 \times 10^2</math></b> |
| worst  | $3.27 \times 10^3$ | $1.95 \times 10^3$                   | $1.57 \times 10^3$                   | <b><math>1.15 \times 10^3</math></b> | $2.09 \times 10^3$                   | <b><math>1.22 \times 10^3</math></b> | $1.72 \times 10^3$                   | $1.62 \times 10^3$                   |
| median | $3.75 \times 10^2$ | $3.90 \times 10^2$                   | $3.74 \times 10^2$                   | <b><math>3.09 \times 10^2</math></b> | $3.67 \times 10^2$                   | $3.82 \times 10^2$                   | $3.83 \times 10^2$                   | <b><math>3.38 \times 10^2</math></b> |
| mean   | $5.20 \times 10^2$ | $4.50 \times 10^2$                   | $4.44 \times 10^2$                   | <b><math>4.27 \times 10^2</math></b> | <b><math>4.36 \times 10^2</math></b> | <b><math>4.37 \times 10^2</math></b> | $4.52 \times 10^2$                   | $4.81 \times 10^2$                   |
| std    | $4.71 \times 10^2$ | $2.51 \times 10^2$                   | $2.42 \times 10^2$                   | <b><math>2.16 \times 10^2</math></b> | $2.71 \times 10^2$                   | <b><math>1.95 \times 10^2</math></b> | $2.58 \times 10^2$                   | $3.19 \times 10^2$                   |

**Table 9.** The statistical results of different parameter combinations in Case 2.

|        |                    |                                      |                                      |                                      |                                      |                                      |                                      |                    |
|--------|--------------------|--------------------------------------|--------------------------------------|--------------------------------------|--------------------------------------|--------------------------------------|--------------------------------------|--------------------|
| $k$    | 4                  | 4                                    | 4                                    | 4                                    | 3                                    | 3                                    | 3                                    | 3                  |
| $x$    | 0.6                | 0.6                                  | 0.6                                  | 0.6                                  | 0.5                                  | 0.5                                  | 0.5                                  | 0.5                |
| $o$    | 0.6                | 0.7                                  | 0.8                                  | 0.9                                  | 0.7                                  | 0.6                                  | 0.8                                  | 0.9                |
| $w$    | 0.8                | 0.8                                  | 0.4                                  | 0.8                                  | 0.8                                  | 0.8                                  | 0.4                                  | 0.8                |
| best   | $2.66 \times 10^2$ | $3.20 \times 10^2$                   | $4.38 \times 10^2$                   | $2.88 \times 10^2$                   | $3.47 \times 10^2$                   | <b><math>2.43 \times 10^2</math></b> | $3.46 \times 10^2$                   | $3.61 \times 10^2$ |
| worst  | $1.83 \times 10^3$ | $2.09 \times 10^3$                   | $3.08 \times 10^3$                   | <b><math>1.61 \times 10^3</math></b> | $2.59 \times 10^3$                   | $2.67 \times 10^3$                   | $2.45 \times 10^3$                   | $2.82 \times 10^3$ |
| median | $7.63 \times 10^2$ | $7.24 \times 10^2$                   | $7.92 \times 10^2$                   | $7.71 \times 10^2$                   | $7.71 \times 10^2$                   | $7.70 \times 10^2$                   | $7.66 \times 10^2$                   | $7.40 \times 10^2$ |
| mean   | $8.52 \times 10^2$ | $7.52 \times 10^2$                   | $8.95 \times 10^2$                   | $7.43 \times 10^2$                   | $9.33 \times 10^2$                   | $8.47 \times 10^2$                   | $8.81 \times 10^2$                   | $8.15 \times 10^2$ |
| std    | $4.60 \times 10^2$ | $3.55 \times 10^2$                   | $4.77 \times 10^2$                   | <b><math>2.56 \times 10^2</math></b> | $5.33 \times 10^2$                   | $4.64 \times 10^2$                   | $4.35 \times 10^2$                   | $4.31 \times 10^2$ |
| $k$    | 4                  | 4                                    | 4                                    | 4                                    | 4                                    | 4                                    | 4                                    | 4                  |
| $x$    | 0.7                | 0.7                                  | 0.7                                  | 0.7                                  | 0.8                                  | 0.8                                  | 0.8                                  | 0.8                |
| $o$    | 0.6                | 0.8                                  | 0.7                                  | 0.9                                  | 0.9                                  | 0.6                                  | 0.8                                  | 0.7                |
| $w$    | 0.8                | 0.4                                  | 0.8                                  | 0.8                                  | 0.8                                  | 0.8                                  | 0.4                                  | 0.8                |
| best   | $2.55 \times 10^2$ | $2.62 \times 10^2$                   | $2.54 \times 10^2$                   | <b><math>2.43 \times 10^2</math></b> | <b><math>2.47 \times 10^2</math></b> | <b><math>2.53 \times 10^2</math></b> | $2.87 \times 10^2$                   | $2.95 \times 10^2$ |
| worst  | $2.29 \times 10^3$ | <b><math>1.77 \times 10^3</math></b> | <b><math>1.06 \times 10^3</math></b> | $1.87 \times 10^3$                   | <b><math>1.77 \times 10^3</math></b> | $1.84 \times 10^3$                   | $2.02 \times 10^3$                   | $2.29 \times 10^3$ |
| median | $6.96 \times 10^2$ | <b><math>6.78 \times 10^2</math></b> | $7.65 \times 10^2$                   | <b><math>6.33 \times 10^2</math></b> | $6.90 \times 10^2$                   | <b><math>6.83 \times 10^2</math></b> | <b><math>6.81 \times 10^2</math></b> | $7.55 \times 10^2$ |
| mean   | $7.40 \times 10^2$ | $7.03 \times 10^2$                   | <b><math>6.75 \times 10^2</math></b> | <b><math>6.67 \times 10^2</math></b> | <b><math>6.99 \times 10^2</math></b> | <b><math>6.96 \times 10^2</math></b> | $7.29 \times 10^2$                   | $7.78 \times 10^2$ |
| std    | $3.97 \times 10^2$ | $3.12 \times 10^2$                   | <b><math>2.05 \times 10^2</math></b> | <b><math>2.56 \times 10^2</math></b> | $3.11 \times 10^2$                   | <b><math>2.62 \times 10^2</math></b> | $3.07 \times 10^2$                   | $3.66 \times 10^2$ |

In Case 1 and Case 2, the performance metrics of each parameter combination are reported in Tables 8 and 9, with best values highlighted in bold. In Case 1 and Case 2,  $k = 4$ ,  $x = 0.7$ ,  $o = 0.9$ , and  $w = 0.8$  achieved the smallest best, median, mean, and standard values.

In summary, three main conclusions can be drawn: First,  $k = 4$  is superior to  $k = 3$ ; second,  $w = 0.8$  frequently appears in high-quality solutions and ensures stable performance; and third,  $x = 0.7$  and  $o = 0.9$  provides a good balance between mean performance and stability. Therefore, the optimal parameter combination is determined as  $k = 4$ ,  $x = 0.7$ ,  $o = 0.9$ , and  $w = 0.8$ , which achieves low mean cost while maintaining robust performance and general applicability, demonstrating the superiority of the selected parameters.

The experiments in the two test cases demonstrate the efficacy of the QLRDE algorithm for UAV path planning. In the two test cases, QLRDE achieved the shortest path lengths

with best costs of 269 (with an average execution time of 93.27 s per run) and 253 (with an average execution time of 80.57 s per run), respectively. The altitude profiles show that paths generated by QLRDE exhibit smaller variations in flight altitude compared to other algorithms. Moreover, cumulative frequency analysis exhibits that 95% of the solutions found by QLRDE have a cost below the threshold of 1500, demonstrating faster convergence than other algorithms. Additionally, the analysis of 16 parameter combinations identified the optimal set as  $k = 4$ ,  $x = 0.7$ ,  $o = 0.9$ , and  $w = 0.8$ , which achieved the lowest mean cost and the most robust performance across both test cases. Experimental results demonstrate that QLRDE outperforms other existing algorithms in terms of solution quality, convergence speed, and computational efficiency.

## 7. Conclusions

This paper contributes an improved QLRDE to overcome local optima and premature convergence in standard DE, with its application to the UAV path planning. The proposed QLRDE integrates three innovations: a quantum-behaved mutation strategy to suppress premature convergence, the LRLM to enhance population diversity, and an adaptive parameter adjust mechanism to enhance the algorithm's robustness and convergence efficiency. These improvements strengthen global exploration and convergence capabilities. Experimental results on twelve benchmark functions demonstrated that QLRDE demonstrates enhanced performance with respect to convergence speed, solution quality, and stability compared to several state-of-the-art algorithms. Applied to 3D UAV path planning, QLRDE generates short and low-altitude path while satisfying realistic constraints including maximum turning angle, terrain avoidance, and threat zones. Results from path planning experiments demonstrate QLRDE's advantages over other algorithms in achieving higher quality solutions, faster convergence, and more efficient computation.

QLRDE is suitable for medium-dimensional optimization problems (e.g., 20–30 dimensions), where it effectively escapes local optima. It also performs well in engineering applications such as UAV 3D path planning and robotic trajectory optimization, which require smooth and feasible solutions under multiple physical and environmental constraints. Furthermore, in scenarios demanding high robustness, QLRDE maintains consistent performance and stable solution quality across multiple independent runs. However, the current QLRDE is limited to single-UAV path planning and does not support cooperative multi-UAV coordination, which involves challenges such as inter-agent collision avoidance and communication constraints. Furthermore, the algorithm is an offline planning method assuming a static environment, and thus does not handle dynamic threats or moving obstacles that would require real-time replanning capabilities.

Future work will focus on two aspects. First, experimental validation will be conducted on real drone platforms or using public UAV path planning datasets to assess the physical feasibility and performance of the planned paths under real-world conditions. Second, we will extend the research to multi-UAVs cooperative path planning problem incorporating more practical constraints.

**Author Contributions:** Methodology, Z.C. and Y.L.; formal analysis, Y.L. and Z.C.; investigation, X.Z. and Y.L.; writing—original draft preparation, Z.C. and Y.L.; writing—review and editing, X.Z.; funding acquisition, X.Z. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work is supported by National Natural Science Foundation of China (No. 62373015).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Data are contained within the article.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Zhang, Z.; Jiang, J.; Ling, K.V.; Wang, X.; Zhang, W.A. Cooperative Path Planning for Heterogeneous UAV Swarms. *IEEE Trans. Autom. Sci. Eng.* **2025**, *22*, 18531–18548. [\[CrossRef\]](#)
2. Mac, T.T.; Copot, C.; Tran, D.T.; Keyser, R.D. Heuristic Approaches in Robot Path Planning: A Survey. *Rob. Auton. Syst.* **2016**, *86*, 13–28. [\[CrossRef\]](#)
3. Yang, P.; Tang, K.; Lozano, J.A.; Cao, X.B. Path Planning for Single Unmanned Aerial Vehicle by Separately Evolving Waypoints. *IEEE Trans. Robot.* **2015**, *31*, 1130–1146. [\[CrossRef\]](#)
4. Lin, Z.; Wu, K.; Shen, R.; Yu, X.; Huang, S. An Efficient and Accurate A-Star Algorithm for Autonomous Vehicle Path Planning. *IEEE Trans. Veh. Technol.* **2024**, *73*, 9003–9008. [\[CrossRef\]](#)
5. Zheng, Z.-A.; Xie, S.; Ye, Z.; Zheng, X.; Yu, Z. Research on Path Smoothing Optimization Based on Improved RRT-Connect Algorithm and Third-Order Bézier Curve. *Proc. Inst. Mech. Eng. Part C J. Mech. Eng. Sci.* **2025**, *239*, 2544–2561. [\[CrossRef\]](#)
6. Xu, F.-X.; Wang, Y.-C.; Cheng, D.-Q.; An, W.-G.; Zhou, C.; Kou, Q.-Q. Reinforcement Learning-Driven Heuristic Path Planning Method for Automated Special Vehicles in Unstructured Environment. *Robot. Auton. Syst.* **2026**, *195*, 105231. [\[CrossRef\]](#)
7. Lyu, Z.; Gao, Y.; Chen, J.; Du, H.; Xu, J.; Huang, K.; Kim, D.I. Empowering Intelligent Low-altitude Economy with Large AI Model Deployment. *arXiv* **2025**, arXiv:2505.22343. [\[CrossRef\]](#)
8. Zhu, C.; Bouteraa, Y.; Khishe, M.; Martin, D.; Hernando-Gallego, F.; Vaiyapuri, T. Enhancing Unmanned Marine Vehicle Path Planning: A Fractal-Enhanced Chaotic Grey Wolf and Differential Evolution Approach. *Knowl.-Based Syst.* **2025**, *317*, 113481. [\[CrossRef\]](#)
9. Wu, Y.; Low, K.H.; Pang, B.; Tan, Q. Swarm-Based 4D Path Planning for Drone Operations in Urban Environments. *IEEE Trans. Veh. Technol.* **2021**, *70*, 7464–7479. [\[CrossRef\]](#)
10. Yu, X.; Jiang, N.; Wang, X.; Li, M. A hybrid algorithm based on grey wolf optimizer and differential evolution for UAV path planning. *Expert Syst. Appl.* **2023**, *215*, 119327. [\[CrossRef\]](#)
11. Zhang, X.Y.; Xia, S.; Zhang, T.; Li, X.Z. Hybrid FWPS Cooperation Algorithm Based Unmanned Aerial Vehicle Constrained Path Planning. *Aerosp. Sci. Technol.* **2021**, *118*, 107004. [\[CrossRef\]](#)
12. Storn, R.; Price, K. Differential Evolution: A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [\[CrossRef\]](#)
13. Neri, F.; Tirronen, V. Recent Advances in Differential Evolution: A Review and Experimental Analysis. *Artif. Intell. Rev.* **2010**, *33*, 61–106. [\[CrossRef\]](#)
14. Das, S.; Suganthan, P.N. Differential Evolution: A Survey of the State-of-the-Art. *IEEE Trans. Evol. Comput.* **2011**, *9*, 4–31. [\[CrossRef\]](#)
15. Brintaki, A.N.; Nikolos, I.K. Coordinated UAV Path Planning Using Differential Evolution. *Oper. Res.* **2005**, *5*, 487–502. [\[CrossRef\]](#)
16. Yang, J.Q.; Yan, F.; Zhang, J.; Peng, C.J. Hybrid Chaos Game and Grey Wolf Optimization Algorithms for UAV Path Planning. *Appl. Math. Model.* **2025**, *142*, 115979.
17. Zhu, L.; Zhou, Y.; Zhou, G.; Luo, Q.; Wei, Y. Solving Spherical Multi-Aircraft Path Planning Problem Using Self-Adaptive Update Strategy Differential Evolution Algorithm. *Swarm Evol. Comput.* **2025**, *96*, 102004. [\[CrossRef\]](#)
18. Xu, P.L.; Luo, W.J.; Lin, X.; Zhang, J.J.; Wang, X. A Large-Scale Continuous Optimization Benchmark Suite with Versatile Coupled Heterogeneous Modules. *Swarm Evol. Comput.* **2023**, *78*, 101280. [\[CrossRef\]](#)
19. Zhang, H.; Sun, J.; Tan, K.C.; Xu, Z. Learning Adaptive Differential Evolution by Natural Evolution Strategies. *IEEE Trans. Emerg. Top. Comput. Intell.* **2023**, *7*, 872–886. [\[CrossRef\]](#)
20. Wang, Y.; Cai, Z. Combining Multiobjective Optimization with Differential Evolution to Solve Constrained Optimization Problems. *IEEE Trans. Evol. Comput.* **2012**, *16*, 117–134. [\[CrossRef\]](#)
21. Chiou, J.P.; Chang, C.F.; Su, C.T. Variable Scaling Hybrid Differential Evolution for Solving Network Reconfiguration of Distribution Systems. *IEEE Trans. Power Syst.* **2005**, *20*, 668–674. [\[CrossRef\]](#)
22. Zhang, X.Y.; Xia, S.; Li, X.Z.; Zhang, T. Multi-Objective Particle Swarm Optimization with Multi-Mode Collaboration Based on Reinforcement Learning for Path Planning of Unmanned Air Vehicles. *Knowl.-Based Syst.* **2022**, *250*, 109075. [\[CrossRef\]](#)
23. Gyongyosi, L.; Imre, S. A Survey on Quantum Computing Technology. *Comput. Sci. Rev.* **2019**, *31*, 51–71. [\[CrossRef\]](#)
24. Zhang, X.Y.; Xia, S.; Li, X.Z. Quantum Behavior-Based Enhanced Fruit Fly Optimization Algorithm with Application to UAV Path Planning. *Int. J. Comput. Intell. Syst.* **2020**, *13*, 1315–1331. [\[CrossRef\]](#)
25. Liu, T.Y.; Jiao, L.C.; Ma, W.P.; Shang, R.H. Quantum-Behaved Particle Swarm Optimization with Collaborative Attractors for Nonlinear Numerical Problems. *Commun. Nonlinear Sci. Numer. Simul.* **2017**, *44*, 167–183. [\[CrossRef\]](#)
26. Ozsoydan, F.B.; Baykasoglu, A. Quantum Firefly Swarms for Multimodal Dynamic Optimization Problems. *Expert Syst. Appl.* **2019**, *115*, 189–199. [\[CrossRef\]](#)

27. Zhang, X.Y.; Lu, X.Y.; Jia, S.M.; Li, X.Z. A Novel Phase Angle-Encoded Fruit Fly Optimization Algorithm with Mutation Adaptation Mechanism Applied to UAV Path Planning. *Appl. Soft Comput.* **2018**, *70*, 371–388. [[CrossRef](#)]
28. Yu, X.; Xu, W.Y.; Li, C.L. Opposition-Based Learning Grey Wolf Optimizer for Global Optimization. *Knowl.-Based Syst.* **2021**, *226*, 107139. [[CrossRef](#)]
29. Guo, H.; Li, S.; Qi, K.; Guo, Y.; Xu, Z. Learning Automata Based Competition Scheme to Train Deep Neural Networks. *IEEE Trans. Emerg. Top. Comput. Intell.* **2020**, *4*, 151–158. [[CrossRef](#)]
30. Nasser, A.B.; Zamli, K.Z.; Hujainah, F.; Ghanem, W.; Alduais, N. An Adaptive Opposition-Based Learning Selection: The Case for Jaya Algorithm. *IEEE Access* **2021**, *9*, 55581–55594. [[CrossRef](#)]
31. Besada-Portas, E.; de la Torre, L.; Moreno, A.; Risco-Martín, J.L. On the Performance Comparison of Multi-Objective Evolutionary UAV Path Planners. *Inf. Sci.* **2013**, *238*, 111–125. [[CrossRef](#)]

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.