

Article

Research on Multi-Strategy Fusion of the Chimpanzee Optimization Algorithm and Its Application in Path Planning

Xing He and Chenxu Guo * 

College of Information and Control Engineering, Xi'an University of Architecture and Technology (XUAT), No. 13 Yanta Road, Xi'an 710055, China; hexing614@xauat.edu.cn

* Correspondence: cxguo@xauat.edu.cn

Abstract: In this paper, a multi-strategy enhanced chimpanzee optimization algorithm (MSEChOA) acting on path planning for delivery vehicles is proposed to achieve the goal of shortening global path lengths for the delivery unmanned vehicles and obtaining safer paths. In the initialization phase, the algorithm introduces a hybrid good point set and chaos initialization strategy, combining the advantages of both to enhance the randomness and homogeneity of the initial population. After that, it incorporates a benchmark weight strategy and Gaussian-modulated cosine factor to adaptively adjust algorithm parameters, thus balancing the global and local search capabilities and improving the search efficiency. In the end, the algorithm incorporates a global search enhancer (GEE) to further enhance the global search capability in the later phases, thereby avoiding local optima. Experiments on several benchmark test functions show that MSEChOA outperforms traditional ChOA and other optimization algorithms in optimization accuracy and convergence speed. In simulation experiments, MSEChOA shows stronger path planning ability and good computational efficiency in both simple and complex environments, proving its feasibility and superiority in the field of path planning.

Keywords: delivery vehicles; path planning; optimization algorithms; chimpanzee optimization; multi-strategy optimization



Academic Editor: Yutaka Ishibashi

Received: 6 November 2024

Revised: 30 December 2024

Accepted: 31 December 2024

Published: 10 January 2025

Citation: He, X.; Guo, C. Research on Multi-Strategy Fusion of the Chimpanzee Optimization Algorithm and Its Application in Path Planning. *Appl. Sci.* **2025**, *15*, 608. <https://doi.org/10.3390/app15020608>

Copyright: © 2025 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) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In recent years, the demand for unmanned logistics has increased significantly. As key logistics equipment, unmanned delivery vehicles play a vital role in warehousing, sorting, and transportation processes [1]. Path planning, a core issue [2,3], directly impacts operational efficiency and task completion quality. Researchers worldwide have extensively studied path planning, achieving notable progress through algorithms ranging from traditional methods like the A* algorithm [4,5] (A-star algorithm), artificial potential field method [6], and Dijkstra's algorithm [7], for its improvement, and intelligent algorithms such as simulated annealing [8], ant colony optimization [9], genetic algorithms, and grey wolf optimization.

However, traditional algorithms struggle in complex environments with multiple obstacles, resulting in poor timeliness and lengthy paths, which hinder unmanned vehicle operations. To address this, metaheuristic algorithms inspired by natural biological systems have gained attention for their efficiency [10,11].

These algorithms, mimicking collective behavior or ecological mechanisms, excel in optimization tasks. Among them, genetic algorithms were among the first applied to combinatorial optimization problems and are widely used in robotic path planning [12].

Recently, the Grey Wolf Optimizer (GWO) [13] has become a prominent metaheuristic algorithm for various optimization problems. Alejo-Reyes et al. [14] improved GWO by introducing weight factors and displacement vectors to enhance exploration strategies and avoid infeasible solutions. Alkhraisat et al. [15] tackled local optima by incorporating mutation methods to control mutation rates effectively.

The Chimp Optimization Algorithm (ChOA) is a novel metaheuristic inspired by chimpanzee population behavior [16], combining the strengths of the Grey Wolf Optimization Algorithm, Dragonfly Algorithm, and Whale Optimization Algorithm. It features strong global search capability, adaptability, and fewer control parameters. However, traditional ChOA suffers from slow convergence and low accuracy. To address these issues, Khishe et al. (2021) proposed a weighted chimpanzee algorithm, improving convergence speed and mitigating local optima capture in high-dimensional optimization problems, validated through 13 high-dimensional and 10 real-world optimization cases [17]. The weighted Chimp Optimization Algorithm has shown improvements in high-dimensional optimization through a weighting mechanism. However, it still encounters common issues found in classic heuristic optimization algorithms, such as slow convergence, susceptibility to local optima, and high computational resource demands. In 2022, Bo Q et al. combined Particle Swarm Optimization (PSO) local search techniques with a new constraint handling method, ensuring robustness in photovoltaic model parameter estimation optimization [18]. Despite its versatility, PSO often falls into local optima, particularly in complex or high-dimensional problems, limiting its global search capability.

To address these issues, Qian L et al. (2022) introduced six spiral functions and proposed two new hybrid spiral functions into the ChOA algorithm, aiming to mitigate slow search speeds and premature convergence [19]. However, spiral functions require precise parameter tuning to remain effective across different problem domains. Wang J et al. also identified limitations of ChOA in handling binary problems and developed a binary version, BChOA, using four S- and V-shaped transfer functions alongside a new binary method [20]. Comparisons of the four new binary optimization algorithms with 18 state-of-the-art methods revealed that BChOA, supported by V-shaped transfer functions, achieved significant statistical improvements. However, BChOA is tailored for binary problems rather than continuous ones, limiting its applicability. While it performs well in specific tasks, its generalization ability may be insufficient.

In 2024, Lan Z et al. proposed a chimpanzee optimization algorithm integrating good point sets, Cauchy operators, and simplex strategies to solve two real engineering problems [21]. The good point strategy and Cauchy operator enhance global exploration but may weaken local exploitation, while the simplex method's reliance on local search makes balancing global and local exploration challenging. Zhou et al. [22] proposed a Nonlinear Improved Grey Wolf Optimization Algorithm (NI-GWO) in 2024, introducing nonlinear dynamic adjustment into the convergence factor mechanism and combining it with the Dynamic Window Approach (DWA) to handle dynamic obstacles. This approach improved UAV path planning by dynamically balancing global exploration and local exploitation while enhancing adaptability to dynamic environments. Li J et al. developed a Multi-Strategy Improved Seagull Optimization Algorithm (MSOA), which incorporates chaotic initialization, multi-directional somersault migration, and spiral leap predation strategies [23]. These strategies significantly improved global search and local exploitation capabilities. Experimental results showed that MSOA performs exceptionally well on standard test functions and robot path planning tasks, quickly generating obstacle-avoiding paths and demonstrating strong potential for path planning applications. However, it lacks comparisons with more advanced algorithms and sufficient testing in practical scenarios.

In summary, the traditional Chimp Optimization Algorithm (ChOA) and its improved versions exhibit strong global search capabilities and adaptability. Strategies such as weighting mechanisms, sine–cosine functions, and neighborhood search have improved convergence speed and optimization precision to some extent. However, these methods still face challenges, including reliance on initial solutions, high computational complexity, inadequate adaptability to dynamic environments, and suboptimal performance in certain complex scenarios.

To overcome these limitations, this paper proposes a multi-strategy improved method that integrates hybrid good point sets and chaos initialization, Gaussian-modulated cosine factors, benchmark weight strategies, and a global search enhancer (GEE). These enhancements aim to improve search efficiency and path planning accuracy. Experimental results confirm the effectiveness and superiority of the proposed method in solving path planning problems for unmanned delivery vehicles.

2. Chimpanzee Optimization Algorithm

The social behavior of chimpanzee populations inspires the Chimpanzee Optimization Algorithm (ChOA). In this algorithm, chimpanzees are categorized into four types of social roles based on their abilities: attackers, encircles, drivers, and chasers. The main search process is abstracted as chimpanzee hunting, which consists of two main processes: an exploration phase and an exploitation phase [24].

In the exploration phase of the ChOA, each chimpanzee adjusts its position according to the assigned task (attacker, encircler, driver, chaser). The formula for updating the current individual position is:

$$X_{chimp}(t+1) = X_{prey}(t) - a \cdot d \quad (1)$$

$$d = \left| c \cdot X_{prey}(t) - m \cdot X_{chimp}(t) \right| \quad (2)$$

X_{chimp} represents the current position vector of the chimpanzee, and X_{prey} represents the position vector of the prey. The vectors a and c , respectively, influence the distance between the chimpanzee and the prey and represent the impact factors of obstacles to the chimpanzee's hunting. d is the distance between the chimpanzee and the prey, and m is the chaos vector, as shown in the equation: when $|a| > 1$, the individual moves away from the prey for global search; when $|a| \leq 1$, the individual gradually approaches the prey for local exploitation.

The expressions for a and c are:

$$a = 2 \cdot f \cdot r_1 - f \quad (3)$$

$$c = 2 \cdot r_2 \quad (4)$$

where f is the convergence factor that nonlinearly decreases from 2.5 to 0 as the number of iterations increases, and r_1 and r_2 are two random numbers between [0, 1].

The formulas for f and c are as follows:

$$f = 2.5 \left(1 - \frac{t}{T} \right) \quad (5)$$

In the exploitation phase, the chimpanzee begins to encircle and capture the prey. During this process, the algorithm gradually approaches the optimal solution by updating the positions of the attacker, encircler, driver, and chaser. The position update model is as follows:

$$X_{(t+1)} = \frac{X_1 + X_2 + X_3 + X_4}{4} \quad (6)$$

where $X_{(t+1)}$ is the updated position vector of the current chimpanzee individual, and X_1 , X_2 , X_3 , and X_4 are the updated position vectors of the attacker, encircler, driver, and chaser, respectively. The mathematical model for their prey attack is as follows:

$$\begin{cases} X_1 = X_{Attacker} - a_1 \cdot d_{Attacker} \\ X_2 = X_{Barrier} - a_2 \cdot d_{Barrier} \\ X_3 = X_{Chaser} - a_3 \cdot d_{Chaser} \\ X_4 = X_{Driver} - a_4 \cdot d_{Driver} \end{cases} \quad (7)$$

$$\begin{cases} d_{Attacker} = |c_1 \cdot X_{Attacker} - m_1 \cdot X| \\ d_{Barrier} = |c_2 \cdot X_{Attacker} - m_2 \cdot X| \\ d_{Chaser} = |c_3 \cdot X_{Chaser} - m_3 \cdot X| \\ d_{Driver} = |c_4 \cdot X_{Driver} - m_4 \cdot X| \end{cases} \quad (8)$$

where d represents the distance between the chimpanzee individual and the prey, and c represents the impact factor of obstacles blocking the chimpanzee during hunting. After finally converging on the target, the chimpanzee will attack the prey, and once the prey stops moving, the chimpanzee completes the hunt.

3. Multi-Strategy Improved Chimpanzee Optimization Algorithm

3.1. Hybrid Good Point Set and Chaos Initialization Strategy

Population diversity is crucial for avoiding local optima and enhancing global exploration. The original ChOA's simple random initialization lacks sophistication. To address this, this paper proposes a hybrid strategy combining good point sets and chaotic initialization. The good point set ensures uniform distribution, while chaotic mapping introduces randomness and diversity, enhancing the algorithm's search capability.

3.1.1. Good Point Set Initialization

The good point set is an effective method for generating uniformly distributed points, particularly in high-dimensional search spaces, where it ensures better distribution than random initialization.

Originally proposed by Hua Luogeng et al. [25], its basic definition and construction are as follows:

The deviation $P_n(k) = \left\{ \left(\left\{ r_1^{(n)} \cdot k \right\}, \left\{ r_2^{(n)} \cdot k \right\}, \dots, \left\{ r_s^{(n)} \cdot k \right\} \right) \mid 1 \leq k \leq n \right\}$ satisfies $\varphi(n) = C(r, \varepsilon)N^{-1+\varepsilon}$, where $C(r, \varepsilon)N^{-1+\varepsilon}$ is a constant only related to r and ε . n denotes the number of points. The set $P_n(k)$ is called a good point set, with r as the good point. $\left\{ r_s^{(n)} \cdot k \right\}$ represents the fractional part, with $r = \left\{ 2 \cos\left(\frac{2\pi k}{p}\right) \mid 1 \leq k \leq s \right\}$, where p is the smallest prime number satisfying $\left(\frac{p-3}{2} \geq s\right)$.

Good point set initialization generates low-discrepancy points within a given dimension, ensuring uniform spatial distribution. This improves the quality of the initial population and mitigates early convergence to local optima. However, the strong regularity of good point sets limits randomness and dynamism, potentially reducing the algorithm's exploration capability in complex or high-dimensional problems.

3.1.2. Chaos Mapping

Chaotic mapping, as a nonlinear dynamic system, possesses pseudo-randomness and ergodicity, enabling the generation of random and diverse points during the initialization phase, thereby enhancing the algorithm's exploration capabilities.

The unpredictability of chaotic point distributions helps prevent falling into local optima. Chaos mapping is a nonlinear dynamic system characterized by sensitivity to initial conditions and ergodicity. Introducing chaos mapping into optimization algorithms

can further enhance population diversity and improve the algorithm's ability to escape local optima in later stages. Common chaos mappings include Logistic mapping, Tent mapping and so on.

In this study, we combined three classic chaotic mappings (Circle mapping, ICMIC mapping, and Sinusoidal mapping) to generate population points with different distribution characteristics, ensuring both uniformity and diversity during the population initialization phase.

The formula for the Circle mapping is as follows:

$$x_{i+1} = \text{mod}(x_i + a - \frac{b}{2\pi} \sin(2\pi x_i), 1) \quad (9)$$

The formula for the ICMIC mapping (Infinite Collapsing Iterative Chaotic Mapping) is as follows:

$$x_{i+1} = \sin\left(\frac{p}{x_i}\right) \quad (10)$$

The formula for the Sinusoidal mapping is as follows:

$$x_{i+1} = a \cdot x_i^2 \cdot \sin(\pi x_i) \quad (11)$$

However, the point distributions generated by chaotic mapping may lack uniformity, resulting in insufficient coverage of the search space.

3.1.3. Hybrid Good Point Set and Chaos Initialization

The initialization strategy proposed in this paper begins by generating a good point set, followed by applying chaotic mapping to introduce perturbations, thereby enhancing its randomness and diversity. This hybrid approach ensures that the initialized point set retains uniform distribution while incorporating sufficient randomness. The goal is to improve the coverage of the search space, enhancing the algorithm's exploration capability and reducing the likelihood of convergence to local optima.

The steps of the hybrid good point set and chaotic initialization are illustrated in the Figure 1 below:

To evaluate the effectiveness of our hybrid strategy that combines good point sets with chaotic initialization, we conducted experiments using various proportions of good point sets alongside chaotic initialization. The experimental results for both unimodal and multimodal functions are illustrated in Figure 2:

As shown in Figure 1, for unimodal functions, the average optimal fitness values are lowest and most stable at 0.6 and 0.8 proportions of the good point set. For multimodal functions, the 0.6 proportion achieves the lowest average optimal fitness value, indicating superior performance. Therefore, subsequent experiments and applications adopt a 0.6 proportion of the good point set for hybrid initialization.

3.1.4. Comparison of Different Initialization Methods

Assuming a two-dimensional search space with a range of $[-2, 2]$ and a population size of 100, the population distributions generated by random initialization, good point set initialization, chaos mapping initialization, and hybrid good point set chaos initialization are compared, as shown in Figure 3:

As shown in the figure above, by combining the good point set and chaos mapping, the advantages of both can be leveraged to ensure uniform distribution of the initial population while increasing diversity and randomness. As a result, this method helps to avoid issues related to local convergence and improves overall global search efficiency.

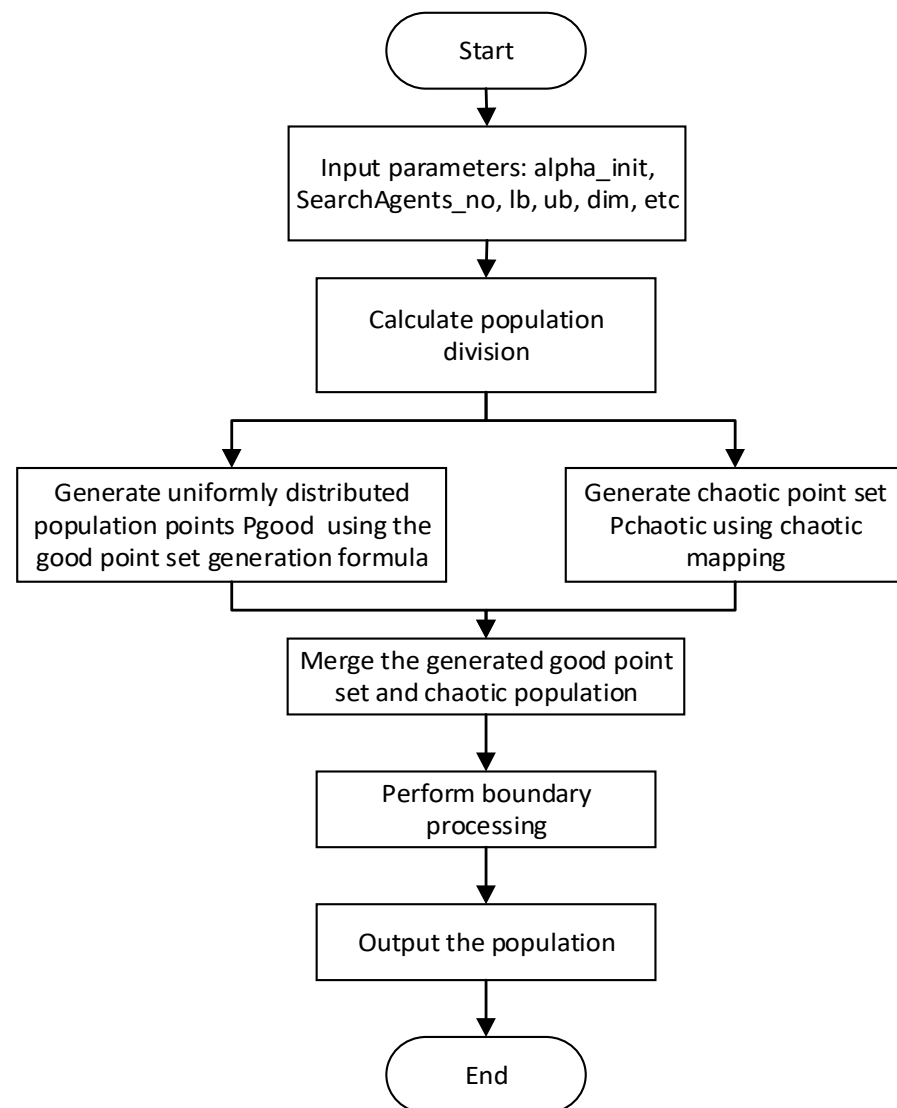
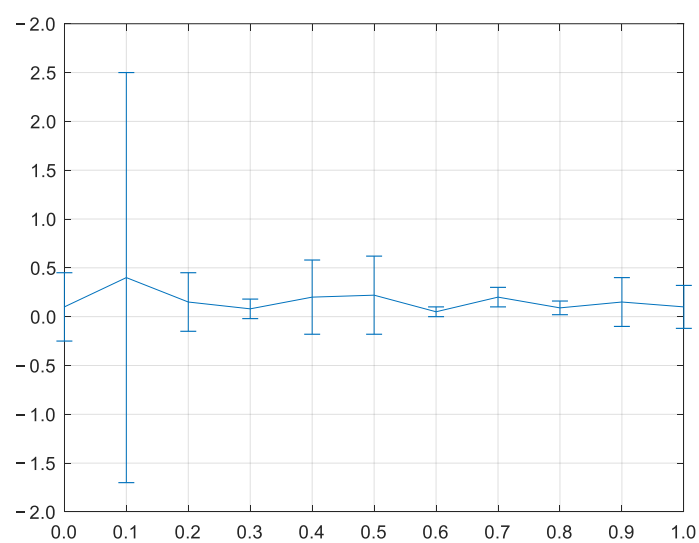


Figure 1. Flowchart of Hybrid Good Point Set and Chaotic Initialization.



(A)

Figure 2. Cont.

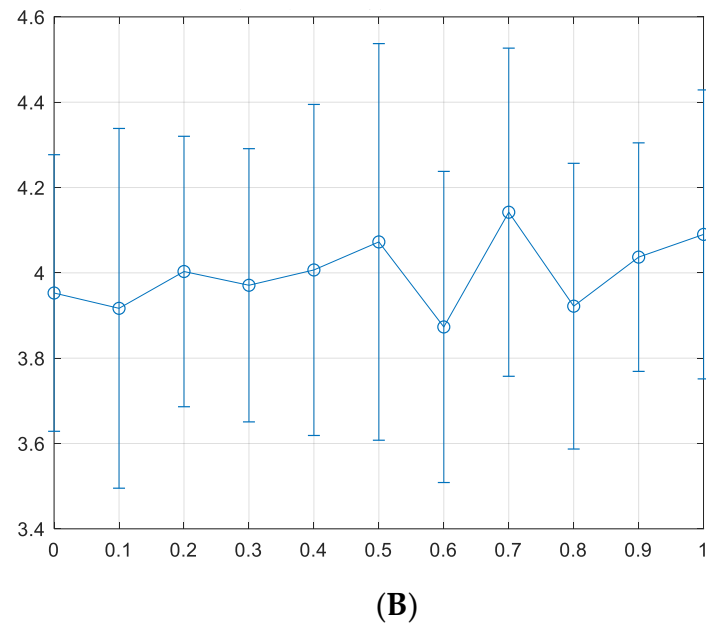


Figure 2. Average fitness for different proportions of good point set chaos initialization. (A) is the unimodal function test result; (B) is the multimodal function test result.

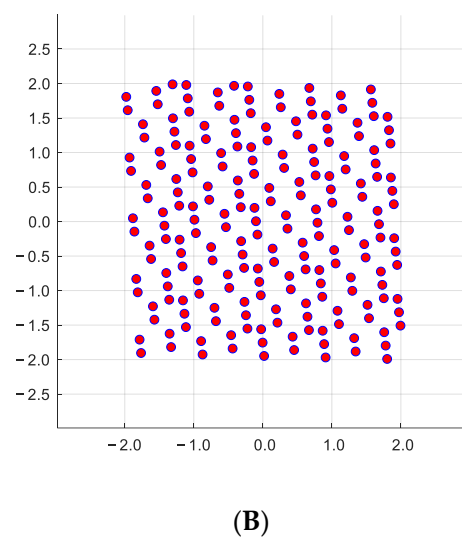
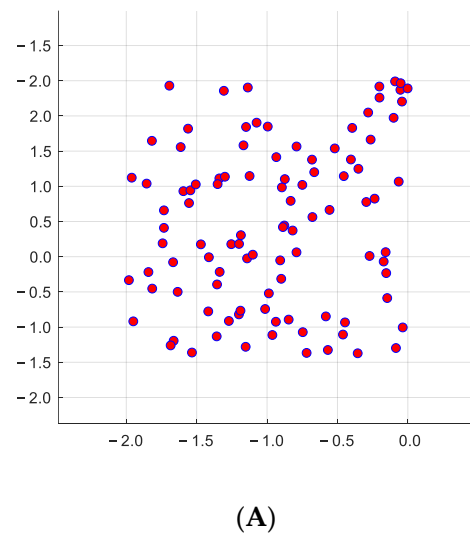
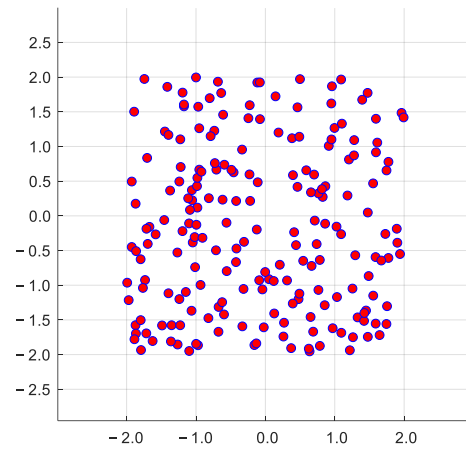
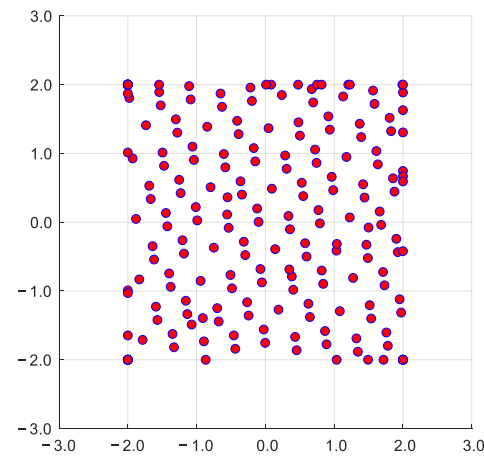


Figure 3. Cont.



(C)



(D)

Figure 3. Initial population distribution generated by different methods. (A) Generate 100 points using random method; (B) generate 100 points using good point set; (C) generate 100 points using chaos mapping; (D) generate 100 points using hybrid good point set and chaos initialization.

3.2. Benchmark Weight Strategy and Gaussian Modulated Cosine Factor

3.2.1. Benchmark Weight Strategy

In complex path planning problems, a simple average position update often fails to achieve rapid convergence to the global optimal solution. Since the attacker has the highest fitness in the current population and exerts the greatest influence, this paper introduces a benchmark weight strategy. In each iteration, the attacker's position (X_1) is fixed as a reference, and the positions of other roles are combined to calculate corresponding weights for position updates. This approach enhances the algorithm's search capability. The improved position update formula is shown below:

$$\left\{ \begin{array}{l} W_1 = \frac{|X_1|}{|X_1|+|X_2|+|X_3|+|X_4|} \\ W_2 = \frac{|X_1|}{|X_2|+|X_2|+|X_3|+|X_4|} \\ W_3 = \frac{|X_1|}{|X_3|+|X_2|+|X_3|+|X_4|} \\ W_4 = \frac{|X_1|}{|X_4|+|X_2|+|X_3|+|X_4|} \end{array} \right. \quad (12)$$

$$X(t+1) = \frac{1}{W_1+W_2+W_3+W_4} \times \frac{W_1 \cdot X_1 + W_2 \cdot X_2 + W_3 \cdot X_3 + W_4 \cdot X_4}{4} \quad (13)$$

Each weight W_1 , W_2 , W_3 and W_4 has its numerator fixed as X_1 , while the denominator consists of different combinations of the absolute values of the positions of the four roles.

This benchmark weight strategy enhances the search ability for the optimal solution, particularly in path planning problems, ensuring that the search agents better adapt to complex environments and improving the algorithm's performance and stability.

3.2.2. Gaussian-Modulated Cosine Factor

In the Chimp Optimization Algorithm (ChOA), the convergence factor f is primarily used to balance global exploration and local exploitation during the search process, playing a crucial role in the algorithm's performance.

However, in Equation (5), the linear decay of f does not align with the actual convergence behavior during the algorithm's execution. To enhance the global search capability and local exploitation ability of the Chimp Optimization Algorithm, this paper proposes a Gaussian-modulated cosine factor, as shown in Equation (14):

$$f = f_{start} + (f_{end} - f_{start}) \cdot \exp\left(-\frac{(t - T/2)^2}{2\sigma^2}\right) \cdot (1 - \cos(\frac{\pi t}{2 \cdot T})) \quad (14)$$

In the above equation, f_{start} and f_{end} are the initial and final values of the convergence factor, respectively. t is the number of current iterations, σ is the standard deviation of the Gaussian function, which controls the width of the Gaussian function, and T is the maximum number of iterations. The Gaussian distribution function enables a smooth adjustment of the factor throughout the entire search process. In the later stages of iteration, the exponential decay of the Gaussian function suppresses the oscillation of the cosine term, allowing the population to focus on the vicinity of the optimal solution and enhancing local exploitation. The cosine modulation term introduces periodic oscillations, which improve the algorithm's randomness and diversity during the global search phase. f_{start} and f_{end} control the dynamic range of the factor, enabling the algorithm to flexibly adapt to the requirements of different search stages. The Gaussian-modulated cosine factor combines the advantages of the Gaussian function and the cosine function, enabling dynamic and smooth adjustments to the convergence factor while providing a balanced effect that other factors cannot achieve simultaneously. Additionally, it does not require frequent parameter adjustments for specific problems, making it suitable for various scenarios, including high-dimensional, multimodal, and dynamic problems. In practical experiments, the Gaussian-modulated cosine factor demonstrated excellent convergence speed, precision, and robustness, particularly outperforming other factors in complex optimization problems.

3.3. Global Exploration Enhancer

In optimization algorithms, search agents must thoroughly explore the solution space to identify the global optimal solution. However, algorithms often become trapped in local optima, reducing search efficiency. To address this, this paper introduces a Global Exploration Enhancer (GEE), which integrates Dynamic Opposition-based Learning (DOBL) [26] and the Sine-Cosine Algorithm (SCA) [27,28]. GEE dynamically adjusts the application probabilities of these strategies across different iteration stages to enhance search performance. During early iterations, GEE prioritizes DOBL to increase population diversity and prevent premature convergence. In mid-iterations, it balances DOBL and SCA to steadily approach the optimal solution. In later stages, GEE predominantly applies SCA to refine solution accuracy.

3.4. Algorithm Flow

Step 1: Input parameters such as population size N , search dimensions dim , search boundaries ub and lb , and maximum number of iterations $Maxiter$.

Step 2: Initialize the population using a hybrid good point set and chaos mapping to ensure diversity and uniform distribution within the population.

Step 3: Calculate the fitness values and select the positions of the four best individuals, denoted as the attacker $X_{Attacker}$, encircler $X_{Barrier}$, chaser X_{Chaser} , and driver X_{Driver} .

Step 4: Calculate the convergence factor f and dynamically adjust its value based on the current iteration count t using the Gaussian-modulated cosine factor formula.

Step 5: In each iteration, selectively apply dynamic opposition-based learning and sine-cosine strategies according to the current stage ratio, and dynamically adjust their application probabilities. Update each chimpanzee's position based on the benchmark weight strategy, calculate new fitness values, and adjust the population structure to ensure convergence towards the optimal solution.

Step 6: Repeat steps 3 through 6 until the algorithm reaches the maximum number of iterations or the convergence criteria are met and the optimal solution is output.

The flowchart of the MSEChOA algorithm is shown in Figure 4:

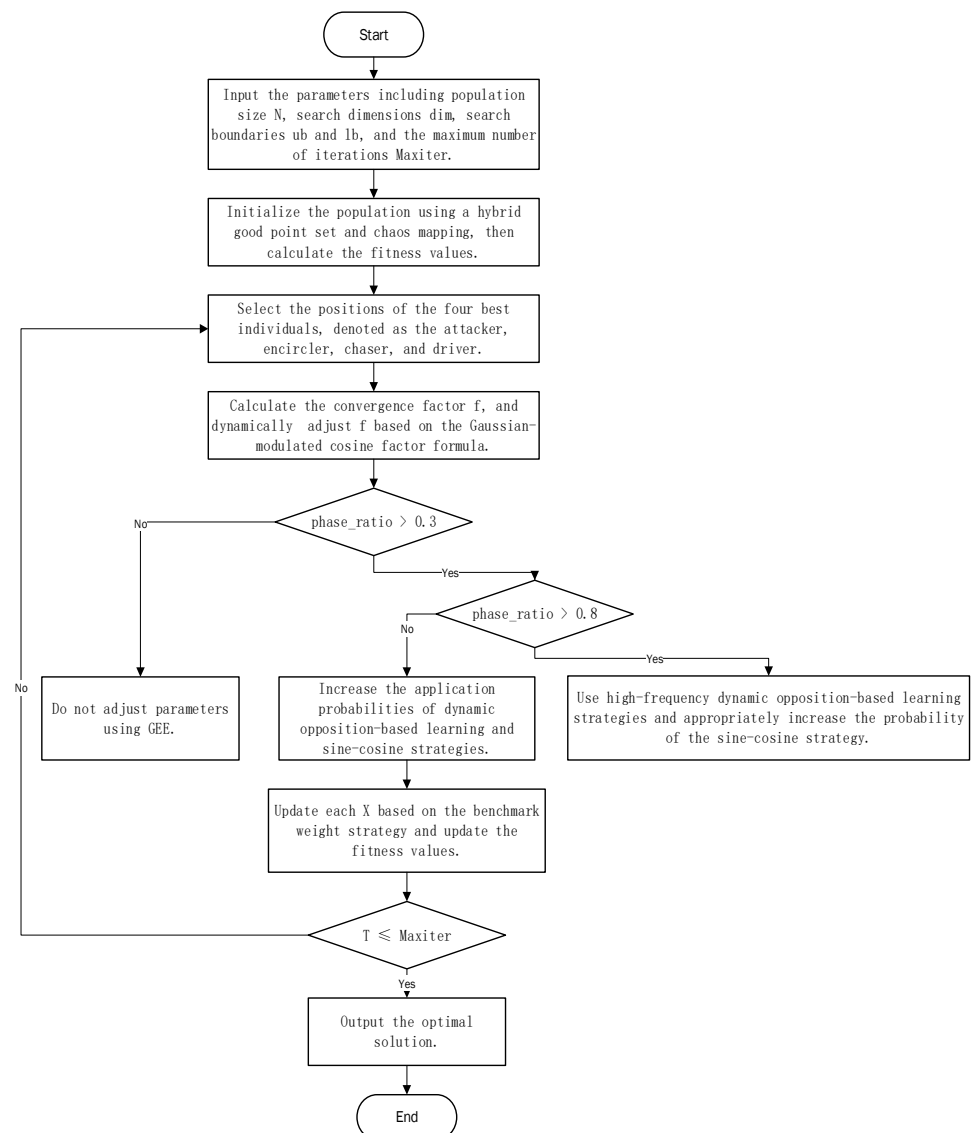


Figure 4. MSEChOA Algorithm Flowchart.

4. Experimental Simulation and Results Analysis

4.1. Experimental Environment and Parameter Settings

The proposed MSEChOA algorithm was tested in a simulation environment based on an Intel (R) Core (TM) i7-8300H CPU @ 2.30 GHz processor, 16 GB RAM, NVIDIA GeForce RTX 3050 Ti, and Windows 10 Professional (64-bit) operating system. The programming software used was Matlab R2021a. The experimental settings were as follows: search space dimension of 30, population size of 30, maximum iterations of 1000, and 30 independent runs.

Table 1 lists the basic information for 11 benchmark test functions, including five unimodal functions (F1 to F5), four nonlinear multimodal functions (F6 to F9), and two fixed-dimension multimodal functions (F10 and F11). The functions F1 to F11 cover a range of optimization problems, from simple to complex, from unimodal to multimodal, and from variable dimensions to fixed dimensions. Detailed information and the search ranges for each function are as follows:

Table 1. List of Benchmark Test Functions.

Function Number	Function Name	Dimension	Domain	Remarks
F1	Sphere	30	[−100, 100]	Unimodal
F2	Schwefel 2.22	30	[−10, 10]	Unimodal
F3	Schwefel 1.26	30	[−100, 100]	Unimodal
F4	Schwefel max	30	[−100, 100]	Unimodal
F5	Rosenbrockp	30	[−30, 30]	Unimodal
F6	Step	30	[−100, 100]	Unimodal
F7	Quartic	30	[−1.28, 1.28]	Unimodal
F8	Schwefel 2.26	30	[−500, 500]	Multimodal
F9	Rastrigin	30	[−5.12, 5.12]	Multimodal
F10	Ackley	30	[−32, 32]	Multimodal (Fixed Dimension)
F11	Griewank	30	[−600, 600]	Multimodal (Fixed Dimension)

4.2. Analysis of the Impact of Different Improvement Strategies on Algorithm Performance

To analyze the impact of different improvement strategies on the performance of the Chimp Optimization Algorithm (ChOA), this paper designs a series of ablation experiments and conducts comparative studies on the following strategies: ChOA refers to the original Chimp Optimization Algorithm; CChOA is the improved algorithm that only applies the hybrid good point set and chaos initialization strategy; GChOA is the improved algorithm that only applies the benchmark weight strategy and Gaussian-modulated cosine factor strategy; EChOA is the improved algorithm that only applies the Global Exploration Enhancer (dynamic opposition-based learning + sine cosine strategy); MSEChOA is the multi-strategy enhanced Chimp Optimization Algorithm that integrates all the above strategies. The results are shown in Table 2.

The results in Table 2 provide qualitative insights into the impact of each improvement strategy on ChOA's performance. The hybrid initialization strategy (CChOA) combines good point sets and chaotic mapping, enhancing population diversity, and uniformity for comprehensive exploration of the search space. This reduces the risk of premature convergence in the early stages but lacks refinement mechanisms for precise local exploitation in later stages. The Gaussian-modulated cosine factor and benchmark weight strategy (GChOA) dynamically adjust the convergence factor, balancing global exploration and local exploitation. This approach maintains population diversity in early iterations, facilitating

the discovery of high-quality solutions, and strengthens local exploitation in later stages, significantly improving solution precision. Consequently, GChOA excels in unimodal and fixed-dimension functions where high convergence accuracy is crucial.

Table 2. Test Results for Different Improved Strategy Algorithms.

Algorithm	Standard Function	Average Fitness Value	Standard Deviation of Fitness Value
ChOA	F1	5.3205×10^{20}	1.4612×10^{20}
CChOA		1.0185×10^{49}	4.4854×10^{49}
GChOA		9.0227×10^{259}	0
EChOA		5.3708×10^{142}	0
MSEChOA		0	0
ChOA	F2	1.1672×10^{14}	1.4103×10^{14}
CChOA		7.9241×10^{37}	2.3578×10^{36}
GChOA		1.1178×10^{206}	0
EChOA		3.3878×10^{116}	0
MSEChOA		2.7981×10^{318}	0
ChOA	F3	2.3789×10^3	5.9865×10^3
CChOA		5.6502×10^{58}	2.6851×10^{58}
GChOA		0	0
EChOA		3.57×10^{118}	0
MSEChOA		0	0
ChOA	F4	2.5449×10^2	4.0083×10^2
CChOA		1.2281×10^{10}	3.8544×10^5
GChOA		1.9831×10^{198}	0
EChOA		3.2545×10^{60}	0
MSEChOA		0	0
ChOA	F5	2.6484×10^1	6.6254×10^1
CChOA		2.7376×10^1	1.1346
GChOA		2.6859×10^1	1.8091×10^2
EChOA		2.5143×10^1	1.8091×10^2
MSEChOA		2.7710×10^1	1.7452×10^1
ChOA	F6	3.3181	3.8517×10^1
CChOA		2.0818	6.9924×10^1
GChOA		2.8806	4.1811×10^1
EChOA		2.6397	4.1811×10^1
MSEChOA		3.1859×10^{17}	3.0727×10^1
ChOA	F7	9.1266×10^2	0.00087611
CChOA		3.0555×10^3	2.2088×10^3
GChOA		3.8075×10^5	3.1309×10^5
EChOA		3.8677×10^5	3.1309×10^5
MSEChOA		1.3864×10^7	1.2646×10^{07}
ChOA	F8	-5.7409×10^3	6.6939×10^1
CChOA		-6.0430×10^3	1.6773×10^2
GChOA		-8.3671×10^3	8.5857×10^1
EChOA		-6.3068×10^3	8.5857×10^1
MSEChOA		-9.8562×10^4	7.1199
ChOA	F9	1.7892×10^2	9.7979×10^2
CChOA		6.2543×10^{12}	4.2156×10^{12}
GChOA		0	0
EChOA		0	0
MSEChOA		0	0

Table 2. Cont.

Algorithm	Standard Function	Average Fitness Value	Standard Deviation of Fitness Value
ChOA	F10	1.9962×10^1	1.2464×10^3
CChOA		1.2967×10^{14}	5.327×10^{15}
GChOA		8.8818×10^{16}	0
EChOA		0.097619	0
MSEChOA		8.8818×10^{16}	0
ChOA	F11	7.1037×10^3	1.3919×10^3
CChOA		1.0514×10^6	3.1708×10^6
GChOA		0	0
EChOA		0	0
MSEChOA		0	0

The Global Exploration Enhancer (EChOA), integrating dynamic opposition-based learning and sine–cosine strategies, enhances global search capabilities. Opposition-based learning generates high-quality opposite solutions, expanding the search range, while the sine–cosine strategy introduces nonlinear updates to escape local optima, particularly excelling in multimodal functions. However, EChOA’s lack of refinement for local exploitation limits its performance on unimodal functions compared to GChOA. By integrating these strategies, MSEChOA achieves a comprehensive balance between exploration and exploitation, excelling across all test functions.

The MSEChOA algorithm surpasses both the original algorithm and those with individual strategy improvements in optimization accuracy, convergence speed, global search capability, and algorithm stability. This demonstrates the significant advantages of the multi-strategy fusion approach in addressing complex optimization problems, validating its effectiveness and superiority.

4.3. Performance Comparison Between MSEChOA and Other Algorithms

To validate the practical effectiveness of MSEChOA, this paper conducts a comprehensive performance comparison with several state-of-the-art metaheuristic optimization algorithms. These include the Grey Wolf Optimizer (GWO), Equilibrium Optimizer (EO) [29], Ant Lion Optimizer (ALO) [30], Multi-Strategy Grey Wolf Optimizer (MSGWO) [31], Chimp Optimization Algorithm (ChOA), Improved Chimp Optimization Algorithm (IChOA) [32], Weighted Chimp Optimization Algorithm (WChOA), and Adaptive Weighted Chimp Optimization Algorithm (AWChOA), which applies adaptive convergence factors and dynamic weight strategies. The specific experimental settings and parameters are shown in Table 3.

Table 3. Parameter Settings of Each Algorithm in the Simulation Experiments.

Algorithm	Parameters
GWO	$r_1 = [0, 1], r_2 = [0, 1], a = [0, 2]$
EO	$r_1 = [0, 1], r_2 = [0, 1], a_1 = 2, a_2 = 1, GP = 0.3$
ALO	$r_1 = [0, 1], r_2 = [0, 1], c > 1$
MSGWO	$r_1 = [0, 1], r_2 = [0, 1], a = [0, 2], h = 0.3$
ChOA	$r_1 = [0, 1], r_2 = [0, 1], f = [0, 2.5]$
IChOA	$r_1 = [0, 1], r_2 = [0, 1], f = [0, 2.5]$
WChOA	$r_1 = [0, 1], r_2 = [0, 1], f = [0, 2.5]$
AWChOA	$r_1 = [0, 1], r_2 = [0, 1], f = [0, 2.5]$
MSEChOA	$r_1 = [0, 1], r_2 = [0, 1], f = [0, 2.5], k = [0, 1], \gamma_1 = [0, 2], \gamma_2 = [0, 1], \gamma_3 = [-1, 1], \gamma_4 = [0, 1]$

In MSEChOA, the parameter k is a dynamic opposition-based learning (DOBL) parameter with a value range of $[0, 1]$. Its design aims to control the generation range of opposition solutions to adapt to the search requirements of different optimization stages. In the early stages of optimization, k has a larger value (close to 1), resulting in a broader distribution of opposition solutions, which helps to cover the search space and enhance global search capabilities. In the later stages, k gradually decreases, leading to a more concentrated distribution of opposition solutions, facilitating the fine-tuning of the local search space. Parameters γ_1 , γ_2 , γ_3 , and γ_4 , respectively, represent the step size adjustment parameter, which controls the magnitude of the search step size; the variation range parameter, which limits the range of solution updates; the direction switching parameter, which enables switching between forward and reverse searches; and the random perturbation parameter, which introduces randomness during the solution update process.

4.3.1. Solution Accuracy Analysis

As shown in Table 4, in the tests on unimodal functions (F1 to F5), the MSEChOA algorithm achieved the theoretical optimal value of 0 on F1 to F4. Although MSEChOA did not surpass MSGWO and AWChOA on the F5 function, its performance was still significantly better than the original ChOA and most other improved algorithms, indicating its robustness in handling complex functions.

Table 4. The experimental results of each algorithm on benchmark test functions.

Algorithm	Standard Function	Best Fitness Value	Average Fitness Value	Worst Fitness Value	Standard Deviation of Fitness Value
GWO	F1	2.7451×10^{61}	2.0058×10^{59}	8.0392×10^{59}	3.4286×10^{59}
EO		8.1172×10^{89}	7.6076×10^{87}	2.7059×10^{86}	1.1166×10^{86}
ALO		4.7580×10^7	5.0654×10^6	9.8827×10^6	3.7870×10^6
MSGWO		0	0	0	0
ChOA		4.0154×10^{22}	2.6677×10^{21}	5.6054×10^{21}	2.3480×10^{21}
ICHOA		5.7817×10^{289}	5.2753×10^{280}	2.6309×10^{279}	0
WChOA		0	0	0	0
AWChOA		0	0	0	0
MSEChOA		0	0	0	0
GWO	F2	4.4984×10^{36}	1.1163×10^{34}	6.5505×10^{34}	1.4335×10^{34}
EO		1.2043×10^{50}	2.927×10^{49}	1.3489×10^{48}	4.0016×10^{49}
ALO		2.8345×10^1	2.6784×10^1	1.2578×10^2	4.3768×10^1
MSGWO		0	0	0	0
ChOA		3.8479×10^{17}	4.7866×10^{14}	1.9832×10^{13}	6.4244×10^{14}
ICHOA		3.3371×10^{171}	8.3392×10^{161}	1.4128×10^{159}	3.1574×10^{160}
WChOA		1.5529×10^{270}	1.4878×10^{267}	1.4111×10^{266}	0
AWChOA		0	0	0	0
MSEChOA		0	0	0	0
GWO	F3	5.9076×10^{19}	2.0702×10^{13}	4.1257×10^{12}	9.2235×10^{13}
EO		4.1664×10^{27}	1.0605×10^{20}	2.0582×10^{19}	4.5960×10^{20}
ALO		4.3682×10^2	1.2297×10^3	1.9752×10^3	4.7699×10^2
MSGWO		0	0	0	0
ChOA		1.4114×10^7	2.3789×10^3	2.5539×10^2	5.9865×10^3
ICHOA		6.2521	6.3564×10^3	2.0569×10^4	5.9676×10^3
WChOA		0	0	0	0
AWChOA		0	0	0	0
MSEChOA		0	0	0	0
GWO	F4	6.7639×10^{16}	2.2771×10^{14}	8.1989×10^{14}	2.5581×10^{14}
EO		6.721×10^{23}	2.0782×10^{21}	1.215×10^{20}	3.8011×10^{21}
ALO		9.3198×10^{23}	4.9083×10^{23}	4.8676×10^{22}	1.5379×10^{22}
MSGWO		0	0	0	1.5379×10^{22}
ChOA		8.3812×10^4	2.5449×10^2	1.1852×10^1	4.0083×10^2
ICHOA		1.1553×10^{90}	1.9693×10^{44}	1.3232×10^{43}	4.4478×10^{44}
WChOA		5.4692×10^{253}	1.1032×10^{249}	1.0184×10^{248}	0
AWChOA		0	0	0	0
MSEChOA		0	0	0	0

Table 4. Cont.

Algorithm	Standard Function	Best Fitness Value	Average Fitness Value	Worst Fitness Value	Standard Deviation of Fitness Value
GWO	F5	2.6014×10^1	2.6887×10^1	2.7976×10^1	6.6254×10^1
EO		2.4195×10^1	2.4447×10^1	2.4845×10^1	1.6682×10^1
ALO		2.6732×10^1	2.6905×10^2	1.7752×10^3	4.3496×10^2
MSGWO		2.8824×10^1	2.8955×10^1	2.8997×10^1	3.5780×10^2
ChOA		2.8062×10^1	2.8836×10^1	2.8948×10^1	1.7312×10^1
ICHOA		2.8743×10^1	2.8794×10^1	2.8868×10^1	2.7903×10^2
WChOA		29	29	29	5.7969×10^{10}
AWChOA		29	29	29	6.5972×10^{16}
MSEChOA		2.7710×10^1	2.8460×10^1	2.8821×10^1	4.2720×10^{18}
GWO	F6	2.9949×10^1	6.6693×10^1	1.5180×10^1	3.4501×10^1
EO		4.8801×10^{11}	2.0216×10^8	1.8594×10^8	1.3300×10^9
ALO		8.4663×10^7	9.1442×10^6	8.3931×10^5	8.3129×10^6
MSGWO		5.7383	6.0207	6.5039	9.4713×10^1
ChOA		3.0442	3.3381	3.6401	2.3828×10^1
ICHOA		1.9045	1.6633	2.5593	9.3163×10^1
WChOA		1.3409	1.4316	1.5224	1.283×10^1
AWChOA		1.1671	1.3172	1.5180	1.8088
MSEChOA		2.1209×10^{17}	4.1698×10^7	5.4862×10^6	1.2778×10^6
GWO	F7	1.9937×10^4	8.3273×10^4	1.8214×10^3	3.7865×10^4
EO		1.8614×10^4	6.3278×10^4	1.6592×10^3	3.9671×10^4
ALO		4.6633×10^2	9.9301×10^2	1.8258×10^1	3.0897×10^2
MSGWO		8.3223×10^7	2.9184×10^5	1.6351×10^4	3.4050×10^5
ChOA		2.1619×10^5	7.8746×10^4	4.5600×10^3	9.6417×10^4
ICHOA		2.1131×10^5	6.4404×10^4	2.1390×10^3	5.3923×10^4
WChOA		3.6663×10^6	8.5284×10^5	5.1775×10^4	1.0510×10^4
AWChOA		9.8998×10^7	2.8929×10^5	9.1816×10^5	2.4063×10^5
MSEChOA		2.4499×10^7	1.5532×10^5	7.1409×10^5	1.5829×10^5
GWO	F8	-7.0131×10^3	-6.0293×10^3	-3.9242×10^3	7.0841×10^2
EO		-9.9502×10^3	-8.8828×10^3	-7.7454×10^3	6.9652×10^2
ALO		-8.8387×10^3	-5.5972×10^3	-5.4177×10^3	6.1683×10^2
MSGWO		-3.3303×10^3	-2.3413×10^3	-1.2683×10^3	4.6097×10^2
ChOA		-5.8594×10^3	-5.7289×10^3	-5.6299×10^3	5.5124×10^1
ICHOA		-1.2569×10^4	-8.0906×10^3	-3.8683×10^3	2.1861×10^3
WChOA		-4.1874×10^3	-3.3371×10^3	-2.5980×10^3	3.1731×10^2
AWChOA		-3.0328×10^3	-2.2503×10^3	-1.5103×10^3	4.0934×10^2
MSEChOA		-1.2569×10^4	-1.2569×10^4	-1.2565×10^4	9.9146×10^1
GWO	F9	0	2.6289×10^1	7.8868	1.4707
EO		0	0	0	0
ALO		6.2682×10^1	8.5168×10^1	1.3432×10^2	1.7589×10^1
MSGWO		0	0	0	0
ChOA		0	5.5223×10^1	1.0257×10^1	2.0008
ICHOA		0	1.3263×10^{14}	1.1369×10^{13}	2.8649×10^{14}
WChOA		0	0	0	0
AWChOA		0	0	0	0
MSEChOA		0	0	0	0
GWO	F10	1.1546×10^{14}	1.8296×10^{14}	2.2204×10^{14}	2.2534×10^{15}
EO		4.4409×10^{15}	4.7962×10^{15}	7.9936×10^{15}	1.1235×10^{15}
ALO		1.6562×10^3	2.0799	3.4621	7.2054×10^1
MSGWO		8.8818×10^{16}	1.5987×10^{15}	4.4409×10^{15}	1.4580×10^{15}
ChOA		1.9956×10^1	1.9962×10^1	1.9964×10^1	1.2985×10^3
ICHOA		4.4409×10^{15}	7.2831×10^{15}	1.5099×10^{14}	3.6692×10^{15}
WChOA		4.4409×10^{15}	4.4409×10^{15}	4.4409×10^{15}	0
AWChOA		8.8818×10^{16}	1.0658×10^{15}	4.4409×10^{15}	7.9441×10^{16}
MSEChOA		8.8818×10^{16}	8.8818×10^{16}	8.8818×10^{16}	0
GWO	F11	0	1.3528×10^3	9.8275×10^3	3.3126×10^3
EO		0	6.1605×10^4	1.2321×10^2	2.7551×10^3
ALO		6.6037×10^4	1.8867×10^2	7.8795×10^2	2.0600×10^2
MSGWO		0	0	0	0
ChOA		0	5.6528×10^3	6.3903×10^2	1.5818×10^2
ICHOA		0	2.2204×10^{17}	1.1102×10^{16}	4.5563×10^{17}
WChOA		0	0	0	0
AWChOA		0	0	0	0
MSEChOA		0	0	0	0

In the multimodal functions (F6 to F9), the MSEChOA algorithm also demonstrated superior performance. On the F6 function, while the fitness value of MSEChOA was slightly lower than that of AWChOA, it had a lower standard deviation, indicating better stability. In the F7 and F8 functions, MSEChOA showed good optimization results, especially in the F8 function, where MSEChOA achieved the optimal result among all tested algorithms, with a standard deviation close to zero, indicating high stability and accuracy. On the F9 function, MSEChOA, like several other improved algorithms, achieved the optimal value of 0, and its performance was comparable to other top algorithms such as AWChOA.

In the fixed-dimension multimodal functions (F10, F11), MSEChOA continued to exhibit excellent performance, particularly in complex high-dimensional search spaces. In both of these functions, MSEChOA achieved the theoretical optimal value and displayed a low standard deviation value, demonstrating its advantage in high-dimensional multimodal optimization problems.

In summary, the MSEChOA algorithm has demonstrated excellent optimization capabilities across various benchmark test functions, especially in handling complex optimization problems.

4.3.2. Convergence Curve Analysis

Convergence curves are commonly used to illustrate the overall performance of optimization algorithms during the iterative process. They depict the changes in the objective function value over successive iterations. By analyzing the convergence curves, one can intuitively assess the convergence speed and stability of the algorithm. As shown in the convergence curves in Figure 5, the MSEChOA algorithm demonstrates outstanding performance across multiple test functions, showing significant advantages compared to other algorithms.

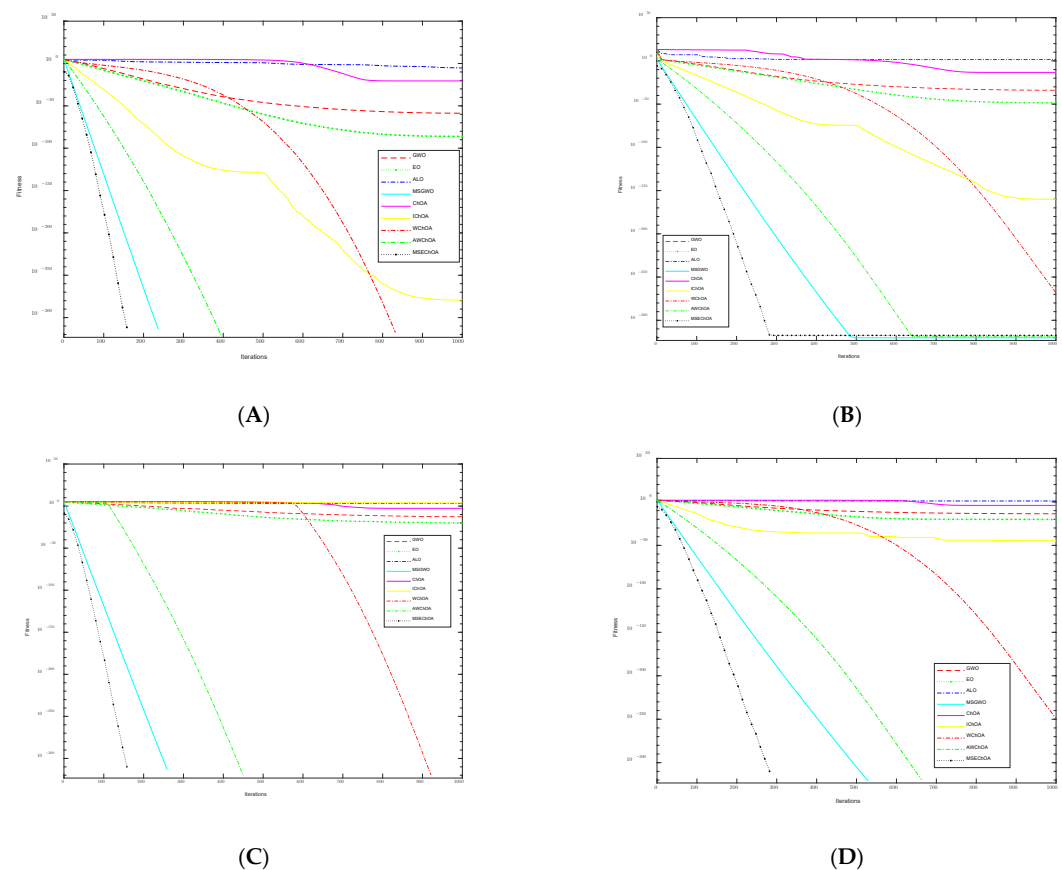


Figure 5. Cont.

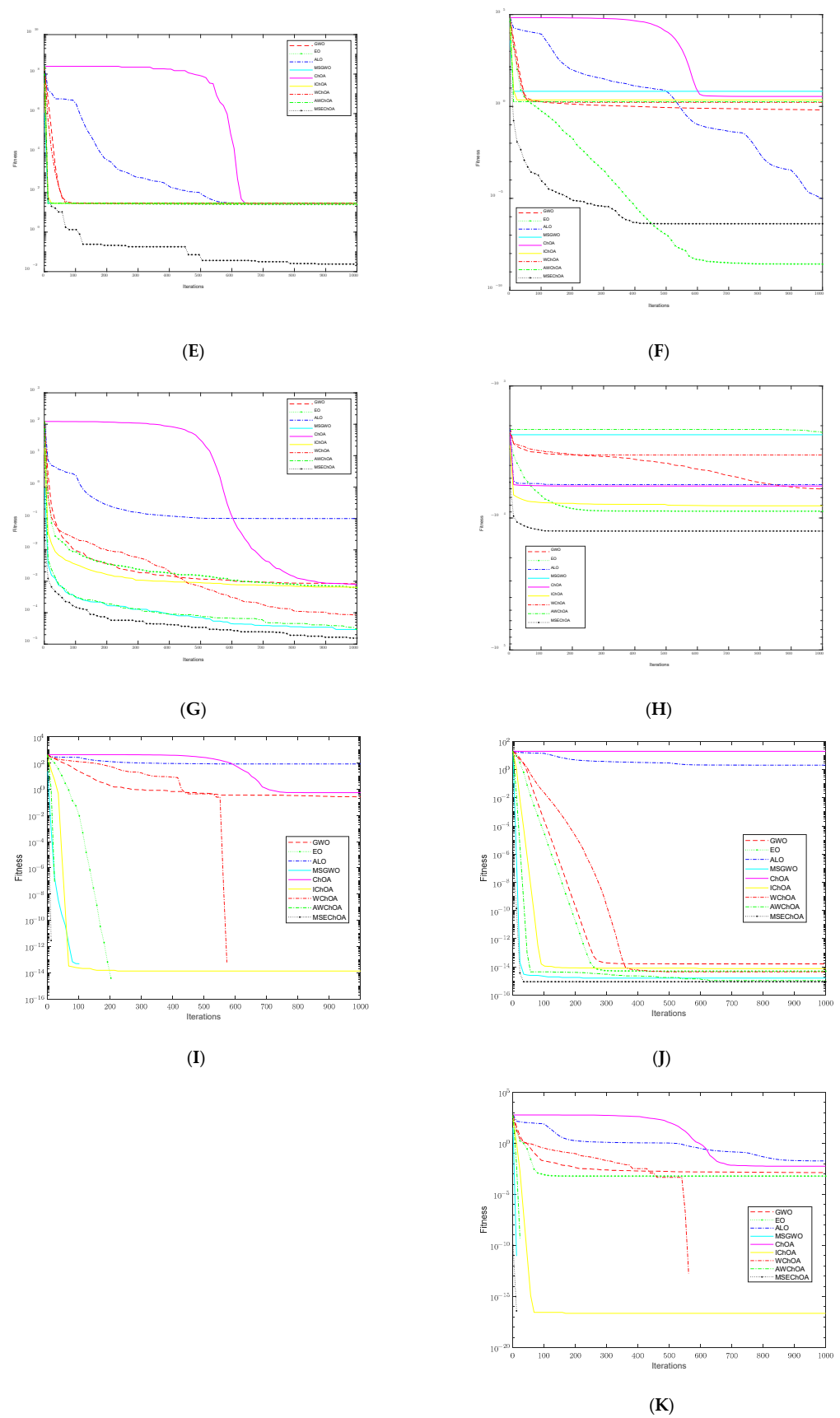


Figure 5. Convergence curves of various algorithms on benchmark functions. (A–E) Convergence results on fixed-dimension multimodal functions (F1~F5); (F–I) Convergence results on fixed-dimension multimodal functions (F6~F9); (J,K) Convergence results on fixed-dimension multimodal functions (F10, F11).

In the tests on unimodal functions (F1 to F5), the convergence curve of the MSEChOA algorithm exhibits an extremely rapid convergence trend. Particularly on the F1, F2, F3, and F4 functions, MSEChOA quickly approaches the theoretical optimal value of 0 almost in the initial iteration stages.

In the tests on multimodal functions (F6 to F9), although the EO algorithm slightly outperforms in some iterations on the F6 function, the convergence curve of MSEChOA shows a more stable downward trend, particularly as it approaches the optimal solution, where the curve's volatility significantly decreases, indicating good convergence stability of the algorithm. The F7, F8, and F9 functions, as typical representatives of complex multimodal functions, show MSEChOA's outstanding performance, with the convergence curve not only quickly approaching the optimal value but also displaying high stability in later iterations, with a standard deviation close to zero.

In the tests on fixed-dimension multimodal functions (F10, F11), MSEChOA also performs excellently. Its convergence curve shows that when dealing with high-dimensional complex search spaces, although other algorithms such as MSGWO and AWChOA perform well, they still exhibit some fluctuations in later iterations, whereas MSEChOA's curve tends to stabilize, with a low standard deviation.

4.3.3. Time Complexity Analysis

Unmanned delivery vehicles need to plan paths quickly in dynamic and complex environments to ensure efficient logistics operations. To ensure that unmanned vehicles can plan paths in real time in complex dynamic environments, the computation time and stability of each algorithm must be rigorously tested and compared. Table 5 shows the average runtime and standard deviation of each algorithm on unimodal, multimodal, and fixed-dimension multimodal functions, thereby evaluating their time complexity in path planning.

Table 5. Average Execution Time and Standard Deviation of Different Algorithms on Various Benchmark Functions.

Algorithm Name	GWO	EO	ALO	MSGWO	ChOA	IChOA	WChOA	AWChOA	MSEChOA
Unimodal Function	0.1840	0.1658	26.8618	0.6503	0.6324	0.0391	0.6649	4.9864	0.2147
Multimodal Function	0.1873	0.1538	25.9900	0.6485	0.6322	0.0375	0.6190	4.8141	0.2034
Fixed-Dimension Multimodal Function	0.1949	0.1610	25.8121	0.6499	0.6467	0.0509	0.6355	4.7628	0.2155
Average Runtime (s)	0.1887	0.1602	26.2213	0.6495	0.6371	0.0425	0.6398	4.8544	0.2112
Average Standard Deviation (s)	0.0068	0.0148	0.5746	0.0282	0.0231	0.004	0.0411	0.1186	0.0115

As shown in Table 5 and Figure 6, the runtime of the ALO algorithm is significantly higher than that of other algorithms, indicating its higher computational cost when handling path planning problems in complex environments. In contrast, the EO and GWO algorithms exhibit higher computational efficiency. The average runtime of the MSEChOA algorithm is moderate; although it is slightly higher than EO and GWO, it is lower than the original ChOA and other improved algorithms. Combined with the algorithm performance results discussed earlier, MSEChOA demonstrates a good balance between computational efficiency and path planning performance.

Further analysis of the standard deviation shows that the MSEChOA algorithm has a lower standard deviation when dealing with multimodal functions and fixed-dimension multimodal functions, indicating better operational stability. This stability is especially important for the path planning of delivery unmanned vehicles in dynamic environments, as it means the algorithm can maintain consistent and efficient performance across different scenarios.

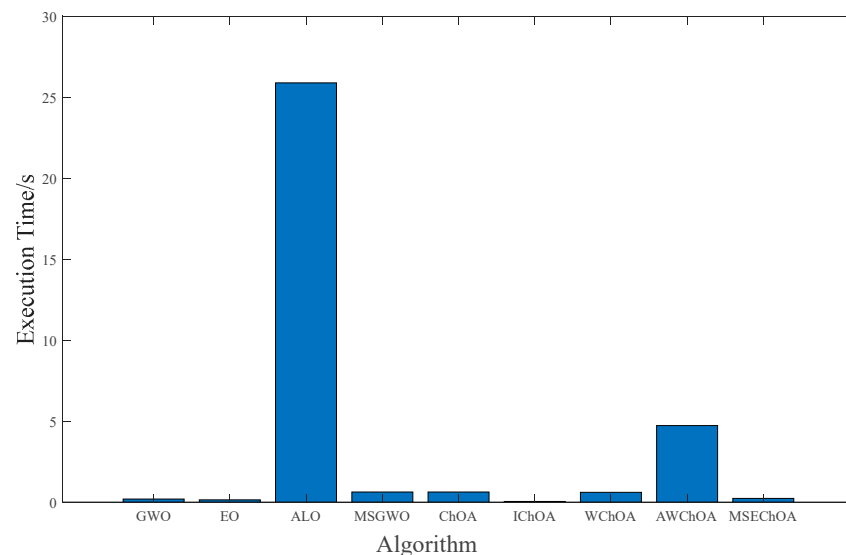


Figure 6. Comparison of execution times among different algorithms.

5. Simulation of Unmanned Delivery Vehicle Path Planning Based on the MSEChOA Algorithm

To validate the effectiveness of the MSEChOA algorithm in actual unmanned delivery vehicle path planning, a simulation test was conducted using the grid method in MATLAB. The experiments were performed on 30×30 scale maps with obstacle coverage rates of approximately 15% and 42%, representing simple and complex environmental maps, respectively, to simulate different real-world delivery scenarios. The experimental parameters were set as follows: a population size of 100 and a maximum iteration value of 1000. To ensure the reliability of the results, each scenario was repeated 10 times.

In grid maps, it is usually necessary to convert between row–column indices and linear indices. Assuming the map size is rows $\times$ cols, the conversion formula for rows and columns is as shown in Equation (15):

$$\begin{cases} index = (r - 1) \times cols + c \\ r = \left\lfloor \frac{index-1}{cols} \right\rfloor + 1 \\ c = index - (r - 1) \times cols \end{cases} \quad (15)$$

The starting and ending points of the delivery unmanned vehicle are marked with green and red dots, respectively, and obstacles are represented by black squares. Different colored paths represent the path planning results of each algorithm. By comparing and analyzing these results, the differences in obstacle avoidance capability, path length, and operational efficiency of the algorithms can be observed. The MSEChOA algorithm was compared with other metaheuristic algorithms and various improved Chimp Optimization Algorithms. A randomly selected experimental result is shown in Figure 7:

Figure 7A,B shows the path planning results of the MSEChOA algorithm compared to other metaheuristic algorithms and various improved Chimp Optimization Algorithms in a simple environment. The paths generated by MSEChOA are shorter and smoother, significantly outperforming the other algorithms. Compared to other improved chimp algorithms, MSEChOA still shows advantages, excelling in both path length and planning efficiency.

Figure 7C,D present the comparison of path planning in a complex environment. MSEChOA still demonstrates excellent obstacle avoidance capabilities when faced with complex obstacles, generating paths that are superior in both length and smoothness compared to other algorithms. Even though algorithms like IChOA and WChOA approach

optimality in some cases, MSEChOA's overall performance is more stable, validating its superiority in complex scenarios. Table 6 presents the data on average path length, average standard deviation, and average runtimes for each algorithm in both simple and complex environments. The following is a detailed analysis of the data in the table:

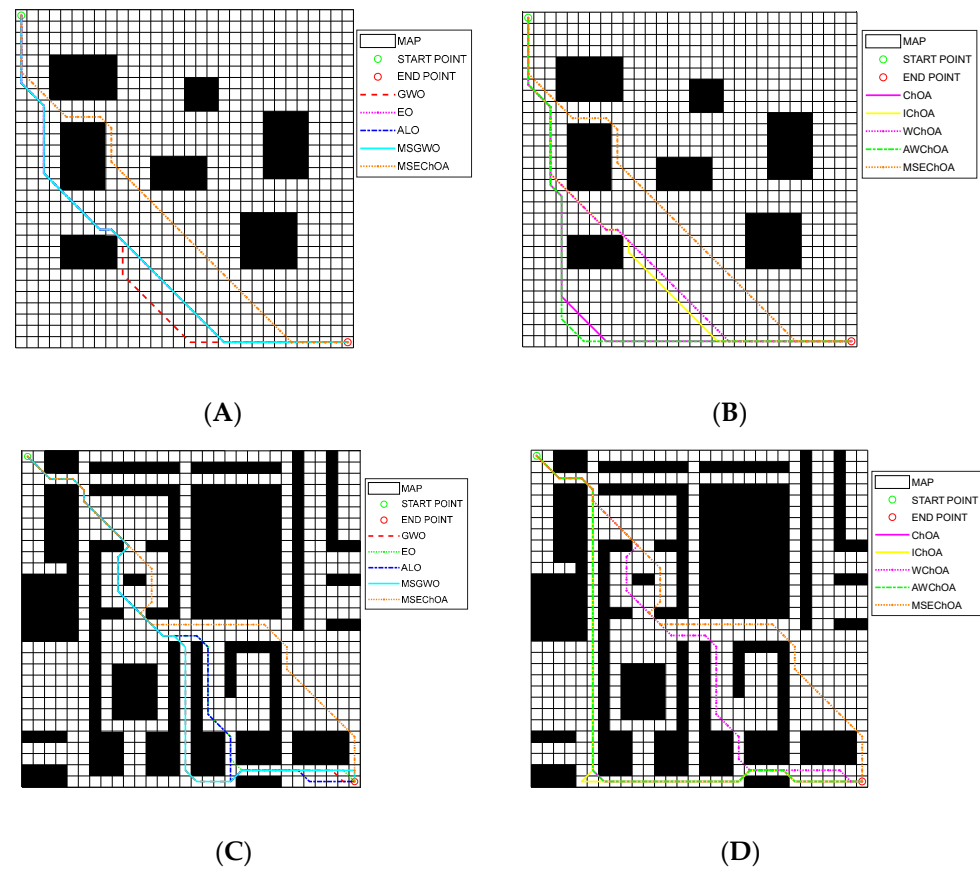


Figure 7. Path planning results of various algorithms in simple and complex environments. (A,B) Path comparison of MSEChOA with other intelligent optimization algorithms in a simple map; (C,D) path comparison of MSEChOA with other intelligent optimization algorithms in a complex map.

Table 6. Path Planning Results of Various Algorithms on Two Different Maps.

Algorithm Name		GWO	EO	ALO	MSGWO	ChOA	IChOA	WChOA	AWChOA	MSEChOA
Simple Map	Average Path Length	49.799	48.0416	48.0416	48.0416	55.0711	47.3137	48.2132	54.4853	45.6985
	Average Standard Deviation	0.6679	1.3099	0.96255	4.5299	2.6	1.2083	1.0479	1.5778	1.1419
	Average Runtime (s)	7.9166	7.8611	11.8017	9.9254	7.7053	7.7646	5.5806	8.1135	5.2658
Complex Map	Average Journey Length	49.799	48.0416	48.0416	48.0416	55.0711	47.3137	48.2132	54.4853	45.6985
	Average Standard Deviation	0.26197	1.0664	1.4163	2.8211	1.6867	2.1569	0.95768	1.8779	0.52394
	Average Runtime (s)	9.9677	9.8957	11.8057	8.9797	8.8052	10.5313	7.9110	8.1135	7.6658

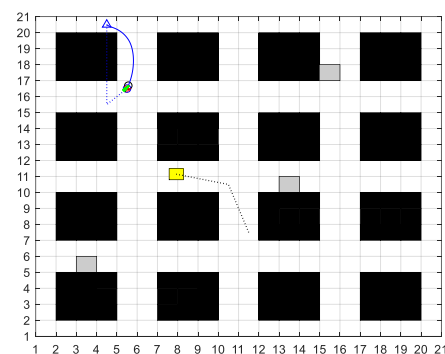
In a simple environment, the MSEChOA algorithm demonstrates the best path planning capability among all algorithms, with an average path length of 45.6985, indicating that this algorithm can effectively find the shortest path in simple environments. In contrast, traditional algorithms like ALO and MSGWO have relatively longer average path lengths, revealing their performance disadvantages in such environments. In a complex environment, MSEChOA also performs excellently, with an average path length of 45.6985, significantly better than other algorithms, especially ChOA (55.0711) and AW-

ChOA (54.4853), demonstrating MSEChOA's significant advantage in handling complex path planning tasks.

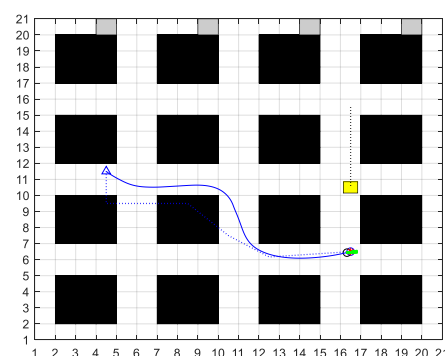
Regarding path planning stability, the MSEChOA algorithm has an average standard deviation of 1.1419 in a simple environment, slightly higher than WChOA's 1.0479, but still at a low level. In a complex environment, MSEChOA's average standard deviation further decreases to 0.52394, the lowest among all algorithms, indicating that the algorithm can not only find shorter paths in complex environments, but also has high path stability and robustness. Additionally, in terms of runtime, MSEChOA also demonstrates high computational efficiency in path planning tasks.

An analysis of Table 6 shows that the MSEChOA algorithm performs excellently in both simple and complex environments, with its performance in complex environments further validating its potential as an efficient and reliable solution for unmanned delivery vehicle path planning.

Meanwhile, in order to simulate a more realistic delivery scenario, this study also simulates a warehouse environment and merges two randomized tasks, where the start point and end point are randomly generated (triangles represent the start point, circles represent the end point, and the blue lines are the routes planned and traveled by the unmanned vehicle) as well as dynamic obstacles. The yellow squares are dynamic obstacles and the dotted line behind them is the trajectory of the dynamic obstacles. The gray squares are unknown obstacles. Dynamic obstacles change their positions through random movements, further increasing the complexity of the scenario and the adaptability requirements of the algorithm, as shown in Figure 8:



(A)



(B)

Figure 8. Simulated real delivery scenarios: (A,B) represent two random tasks.

The experimental results show that in tasks simulating real delivery scenarios, the MSEChOA algorithm can quickly generate optimal paths that avoid dynamic obstacles. Moreover, its stability is evident in complex scenarios involving random tasks and dy-

dynamic obstacles, demonstrating superior global search and local exploitation capabilities compared to other algorithms. This further validates the potential of the MSEChOA algorithm in practical applications, particularly in dynamic logistics and unmanned vehicle delivery domains.

6. Conclusions

To address the inefficiency and limited capability of traditional path planning algorithms in complex environments, this paper proposes a multi-strategy enhanced Chimpanzee Optimization Algorithm (MSEChOA). By incorporating multiple strategies, MSEChOA significantly improves optimization efficiency and adaptability. Comparative experiments with other classic optimization algorithms demonstrate that MSEChOA provides an efficient and reliable solution for unmanned delivery vehicle path planning. The main conclusions are as follows:

- (1) A chimpanzee algorithm (MSEChOA) combining hybrid good point set chaos initialization, benchmark weight strategy, Gaussian-modulated cosine factor, and Global Exploration Enhancer (GEE) strategies was proposed. Compared to the original chimpanzee algorithm, which has limited global search capability and is prone to becoming trapped in local optima when dealing with complex path planning, MSEChOA effectively addresses these limitations, enhancing the algorithm's global search capability and convergence accuracy.
- (2) The experimental results show that MSEChOA performs excellently across eleven benchmark test functions, particularly in optimization accuracy, convergence speed, and global search capability, surpassing traditional and other improved algorithms. In the simulation experiments for delivery unmanned vehicle path planning, MSEChOA not only generated superior path planning schemes in both simple and complex environments, but also exhibited outstanding performance in terms of computational efficiency and algorithm stability. Compared to other algorithms, MSEChOA effectively balances global search and local exploitation capabilities, avoiding the pitfalls of local optima, demonstrating strong adaptability and robustness, making it suitable for handling complex and dynamic real-world scenarios.

The MSEChOA algorithm provides an efficient and reliable solution for delivery unmanned vehicle path planning, with broad application prospects. In addition to its proven effectiveness in simple and complex map environments, the algorithm has also been validated in experiments simulating more realistic logistics scenarios involving random tasks and dynamic obstacles. Future research can focus on further optimizing the algorithm's computational efficiency, improving its adaptability in dynamic and real-world logistics environments, and enhancing its performance in simulations of more realistic logistics scenarios. These efforts aim to refine the algorithm's practical utility and accelerate the development and widespread adoption of unmanned delivery technology.

Moreover, the superior optimization performance of the MSEChOA algorithm also demonstrates extensive application potential in other practical scenarios. For instance, in logistics distribution center scheduling, the algorithm can optimize vehicle and cargo allocation to improve logistical efficiency. In intelligent warehouse path planning, it can assist robots in generating optimal paths within dynamic storage environments, thereby reducing operation time. In industrial production scheduling, it provides efficient scheduling solutions for multi-process manufacturing tasks, thereby enhancing production efficiency.

Despite its outstanding performance in various optimization problems, MSEChOA still has certain limitations. For example, the algorithm's capability in handling real-time path planning in dynamic environments requires further validation and improvement. Additionally, its adaptability to high-dimensional and complex environments also has

room for enhancement. Future research can focus on the following directions: optimizing the real-time performance of MSEChOA in dynamic environments to ensure stability and efficiency in rapidly changing obstacle and target scenarios; integrating deep learning or reinforcement learning methods to further enhance the algorithm's adaptability to high-dimensional and complex scenarios; and conducting deeper investigations into logistics and distribution applications, such as logistics scheduling and traffic information. Further research can explore and improve these issues, enabling MSEChOA to realize its potential in a wider range of practical scenarios, providing more efficient solutions for unmanned delivery, robotic navigation, intelligent logistics, and industrial optimization.

Author Contributions: Research conceptualization, X.H. Advice and guidance, X.H. Overall direction control, X.H. Investigation, X.H. Resources, X.H. Methodology, X.H. Study design, C.G. Software, C.G. Validation, C.G. Experimental execution, C.G. Data analysis, C.G. Visualization, C.G. Preparation of first draft, C.G. Project management, X.H. Review and revision of the manuscript, X.H. All authors have read and agreed to the published version of the manuscript.

Funding: The authors thank the National Natural Science Foundation of China for partial support under Grant No. 72071153. The authors also express their gratitude to the Key Laboratory Fund under Grant no. 6142003190204, and the Natural Science Special Program of Xi'an University of Architecture and Technology under Grant no. ZR20048.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article; further inquiries can be directed to the corresponding author.

Acknowledgments: The authors would like to thank their colleagues at Xi'an University of Architecture and Technology for their constructive discussions and feedback during the research. The authors also acknowledge the anonymous reviewers for their valuable suggestions that greatly improved the manuscript.

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

References

1. Srinivas, S.; Ramachandiran, S.; Rajendran, S. Autonomous robot-driven deliveries: A review of recent developments and future directions. *Transp. Res. Part E Logist. Transp. Rev.* **2022**, *165*, 102834. [\[CrossRef\]](#)
2. Abed, M.S.; Lutfy, O.F.; Al-Doori, Q.F. A Review on Path Planning Algorithms for Mobile Robots. *Eng. Technol.* **2021**, *39*, 804–820. [\[CrossRef\]](#)
3. Das, P.K.; Jena, P.K. Multi-robot path planning using improved particle swarm optimization algorithm through novel evolutionary operators. *Appl. Soft Comput.* **2020**, *92*, 106312. [\[CrossRef\]](#)
4. Guruji, A.K.; Agarwal, H.; Parsediya, D.K. Time-efficient A* Algorithm for Robot Path Planning. In Proceedings of the 3rd International Conference on Innovations in Automation and Mechatronics Engineering, Vallabh Vidyanagar, India, 5–6 February 2016; pp. 144–149.
5. Wei, S.; Yunfeng, L.V.; Hongwei, T.; Min, X. Mobile Robot Path Planning Based on an Improved A Algorithm. *J. Hunan Univ. (Nat. Sci.)* **2017**, *44*, 94–101. (In Chinese) [\[CrossRef\]](#)
6. Costa, A.N.; Medeiros, F.L.; Dantas, J.P.; Geraldo, D.; Soma, N.Y. Formation control method based on artificial potential fields for aircraft flight simulation. *Simulation* **2022**, *98*, 575–595. [\[CrossRef\]](#)
7. Fadzli, S.A.; Abdulkadir, S.I.; Makhtar, M. Robotic Indoor Path Planning Using Dijkstra's Algorithm with Multi-Layer Dictionaries. In Proceedings of the 2015 2nd International Conference on Information Science and Security (ICISS), Seoul, Republic of Korea, 14–16 December 2015; pp. 1–4.
8. Kirkpatrick, S. Optimization by Simulated Annealing. *Science* **1983**, *220*, 671–680. [\[CrossRef\]](#)
9. Eda, Y.; Salman, F.S.; Erdoğan, G. Optimizing two-dimensional vehicle loading and dispatching decisions in freight logistics. *Eur. J. Oper. Res.* **2022**, *302*, 954–969.
10. Abedinia, O.; Amjadi, N.; Ghasemi, A. A new metaheuristic algorithm based on shark smell optimization. *Complexity* **2016**, *21*, 97–116. [\[CrossRef\]](#)

11. Agrawal, P.; Abutarboush, H.F.; Ganesh, T.; Mohamed, A.W. Metaheuristic Algorithms on Feature Selection: A Survey of One Decade of Research (2009–2019). *IEEE Access* **2021**, *9*, 26766–26791. [\[CrossRef\]](#)
12. Yao, M.; Wang, N.; Zhao, L. Improved simulated annealing algorithm and genetic algorithm for TSP. *Comput. Eng. Appl.* **2013**, *49*, 60–65.
13. Mirjalili, S.; Mirjalili, S.M.; Lewis, A.D. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [\[CrossRef\]](#)
14. Alejo-Reyes, A.; Cuevas, E.; Rodríguez, A.; Mendoza, A.; Olivares-Benitez, E. An Improved Grey Wolf Optimizer for a Supplier Selection and Order Quantity Allocation Problem. *Mathematics* **2020**, *8*, 1457. [\[CrossRef\]](#)
15. Alkhraisat, H.; Dalbah, L.M.; Al-Betar, M.A.; Awadallah, M.A.; Assaleh, K.; Deriche, M. Size optimization of truss structures using improved grey wolf optimizer. *IEEE Access* **2023**, *11*, 13383–13397. [\[CrossRef\]](#)
16. Khishe, M.; Mosavi, M.R. Chimp optimization algorithm. *Expert Syst. Appl.* **2020**, *149*, 113338. [\[CrossRef\]](#)
17. Khishe, M.; Nezhadshahbodaghi, M.; Mosavi, M.R.; Martín, D. A Weighted Chimp Optimization Algorithm. *IEEE Access* **2021**, *9*, 158508–158539. [\[CrossRef\]](#)
18. Bo, Q.; Cheng, W.; Khishe, M.; Mohammadi, M.; Mohammed, A.H. Solar photovoltaic model parameter identification using robust niching chimp optimization. *Sol. Energy* **2022**, *239*, 179–197. [\[CrossRef\]](#)
19. Qian, L.; Khishe, M.; Huang, Y.; Mirjalili, S. SEB-ChOA: An improved chimp optimization algorithm using spiral exploitation behavior. *Neural Comput. Appl.* **2024**, *36*, 4763–4786. [\[CrossRef\]](#)
20. Wang, J.; Khishe, M.; Kaveh, M.; Mohammadi, H. Binary Chimp Optimization Algorithm (BChOA): A New Binary Meta-heuristic for Solving Optimization Problems. *Cogn. Comput.* **2021**, *13*, 1297–1316. [\[CrossRef\]](#)
21. Lan, Z.; He, Q. A novel chimp optimization algorithm with cauchy perturbation. *J. Chin. Comput. Syst.* **2023**, *44*, 715–723.
22. Zhou, X.; Shi, G.; Zhang, J. Improved Grey Wolf Algorithm: A Method for UAV Path Planning. *Drones* **2024**, *8*, 675. [\[CrossRef\]](#)
23. Li, J.; Shang, W.; Hu, Y. Multi-Strategy Improved Seagull Algorithm for Solving Robot Path Planning Problems. *Mach. Tool Autom. Technol.* **2024**, *4*, 19–25, 30. (In Chinese) [\[CrossRef\]](#)
24. Askari, Q.; Younas, I.; Saeed, M. Political Optimizer: A novel socio-inspired meta-heuristic for global optimization. *Knowl.-Based Syst.* **2020**, *195*, 105709. [\[CrossRef\]](#)
25. Hua, L.; Wang, Y. *Applications of Number Theory to Numerical Analysis*; Springer: Berlin, Germany, 1981; pp. 83–87.
26. Xie, Y.; Lin, L.; Wu, Z. Dynamic opposition-based learning in nature-inspired optimization: A review of concepts and applications. *Swarm Evol. Comput.* **2023**, *85*, 101057.
27. Chen, C.; Wang, X.; Yu, H.; Wang, M.; Chen, H. Dealing with multi-modality using synthesis of Moth-flame optimizer with sine cosine mechanisms. *Math. Comput. Simul.* **2021**, *188*, 291–318. [\[CrossRef\]](#)
28. Gupta, R.; Singh, P.K.; Ghosh, A. Hybrid sine cosine algorithm for solving engineering optimization problems. *Expert Syst. Appl.* **2023**, *223*, 119725.
29. Faramarzi, A.; Heidarinejad, M.; Stephens, B.; Mirjalili, S. Equilibrium optimizer: A novel optimization algorithm. *Knowl.-Based Syst.* **2020**, *191*, 105190. [\[CrossRef\]](#)
30. Mirjalili, S. The Ant Lion Optimizer. *Adv. Eng. Softw.* **2015**, *83*, 80–98. [\[CrossRef\]](#)
31. Qin, H.; Wang, L.; Fu, Y.; Sui, M.; He, B.J. Grey Wolf Optimization Algorithm Based on Multi-Strategy Combination and Its Application. *Shandong Univ.* **2024**, *59*, 51–60. (In Chinese)
32. Liu, H.; Lei, B.; Wang, W.; Yun, Y.; Chai, H. Path Planning for Warehouse Mobile Robots Based on an Improved Chimpanzee Optimization Algorithm. *Inf. Control* **2023**, *52*, 689–700. (In Chinese)

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.