

Article

Research on Path Planning for Mobile Robot Using the Enhanced Artificial Lemming Algorithm

Pengju Qu ^{1,*}, Xiaohui Song ² and Zhijin Zhou ¹¹ Engineering Training Center, Guizhou Institute of Technology, Guiyang 550025, China; 20140590@git.edu.cn² College of Agriculture, Anshun University, Anshun 561000, China; 18985436547@163.com

* Correspondence: 20140497@git.edu.cn

Abstract

To address the key challenges in shortest path planning for known static obstacle maps—such as the tendency to converge to local optima in U-shaped/narrow obstacle regions, unbalanced computational efficiency, and suboptimal path quality—this paper presents an enhanced Artificial Lemming Algorithm (DMSALAs). The algorithm integrates a dynamic adaptive mechanism, a hybrid Nelder–Mead method, and a localized perturbation strategy to improve the search performance of ALAs. To validate DMSALAs efficacy, we conducted ablation studies and performance comparisons on the IEEE CEC 2017 and CEC 2022 benchmark suites. Furthermore, we evaluated the algorithm in mobile robot path planning scenarios, including simulated grid maps (10×10 , 20×20 , 30×30 , 40×40) and a real-world experimental environment built by our team. These experiments confirm that DMSALAs effectively balance optimization accuracy and practical applicability in path planning problems.

Keywords: mobile robot; planning issues; algorithm; meta-heuristic**MSC:** 68W50

Academic Editors: Rui Zhong and Jun Yu

Received: 28 September 2025

Revised: 21 October 2025

Accepted: 30 October 2025

Published: 4 November 2025

Citation: Qu, P.; Song, X.; Zhou, Z. Research on Path Planning for Mobile Robot Using the Enhanced Artificial Lemming Algorithm. *Mathematics* **2025**, *13*, 3533. <https://doi.org/10.3390/math13213533>

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

Meta-heuristic (MH) algorithms excel at solving complex optimization problems like mobile robot path planning in static environments, where traditional methods struggle with non-convex obstacle avoidance and high-dimensional search spaces. Their ability to balance global exploration (finding feasible paths around fixed obstacles) and local exploitation (optimizing path length) makes them particularly suitable for grid-based navigation systems. The No Free Lunch theorem [1] further justifies developing specialized variants for structured environments with predetermined obstacle configurations.

Recently, the field of MH algorithms has witnessed a remarkable increase in research focus and attention. There are many types of MH algorithms, including swarm intelligence algorithms, evolutionary algorithms, and physics-inspired algorithms. There are many swarm intelligence algorithms; for example, the improved gorilla troops optimizer (IGTO) [2] represents a novel meta-heuristic approach that integrates lens opposition and adaptive hill strategies. The Horse Herd Optimization Algorithm (HOA) [3] derives from observed social behaviors in equine populations, systematically encoding six characteristic traits across growth phases. The Graylag Goose Optimization Algorithm (GGO) [4] computationally emulates the self-organizing flight patterns of geese that emerge from altitude and meteorological variations. Binary variants of the Artificial Bee Colony algorithm

(BABC) [5] preserve the defined criteria while inheriting the advantages of the original ABC algorithm. The Modified Artificial Hummingbird Algorithm (MAHA) [6] improves upon the performance of its predecessor. The Whale Optimization Algorithm (WOA) [7] is inspired by the bubble-net hunting strategy, which can precisely circumvent obstacles along local paths. The hybrid gray wolf optimization (HGWO) [8–10] utilizes combined metaheuristic principles to generate collision-free optimal paths for autonomous robots. The improved PSO (IPSO) [11] improves the performance of autonomous path planners by optimizing 3D route generation for speed and precision. The improvement adaptive ant colony algorithm (IAACO) [12] integrates four key factors—directional guidance, obstacle exclusion, adaptive heuristic information, and adjustable pheromone decay to optimize indoor robotic path planning. The proposed algorithm, named the hybrid Particle Swarm Optimization-Simulated Annealing algorithm [13,14], successfully avoids local optima entrapment and enhances search diversity, resulting in quicker convergence and decreased resource consumption. Each of these algorithms possesses distinct characteristics. For example, the IGTs algorithm exhibits strong exploration capability, excellent balance, and high robustness; however, it suffers from increased computational complexity and requires careful parameter tuning. Similarly, the WOAs algorithm offers a simple structure, ease of implementation, and strong local exploitation ability, yet it is prone to premature convergence and demonstrates limited global exploration performance. Evolutionary algorithms include many algorithms, such as the Population State Evaluation-based Differential Evolution (PSEDEs) [15], which incorporates insights from the evolutionary states of Differential Evolution (DEs), and the enhanced Genetic Algorithm (EGAs) [15,16], which focuses on advancing path initialization techniques in continuous space, ultimately deriving the best-performing route between given start and goal coordinates. The PSEDEs algorithm features a simple structure and ease of implementation, with strong local exploitation capability and fast convergence speed. However, it is prone to receiving traps in local optima, and its search behavior heavily depends on population distribution. The EGAs algorithm demonstrates robust global exploration ability and low dependency on specific problem structures, yet it suffers from slow convergence and high sensitivity to parameter settings. In addition to the above two algorithms, there are also physics-inspired algorithms, for example, the Multi-Strategy Boosted Snow Ablation Optimizer (MSAO) [16] presents a new SAO version enhanced by four optimization strategies. The Interior Search algorithm (ISA) [17] takes methodological cues from harmonic balance concepts in decor composition. The Farmland Fertility algorithm (FFAs) [18] computationally models the productivity cycles inherent in arable land biomes. The MSAOs algorithm features a novel search mechanism with clear physical intuition, yet it suffers from insufficient exploitation capability. The ISAs algorithm incorporates a unique update mechanism but is prone to receiving traps in local optima. The FFAs algorithm exhibits a strong metaphorical foundation and adaptability, along with robust global exploration ability; however, it has weaknesses in local exploitation and requires careful parameter tuning. These original and improved algorithms leverage various strategies, such as Levy flight [19] can improve the global search efficiency of robots in sparse obstacle environments, bidirectional search strategy [20] is capable of narrowing down the search scope and decreasing the time required to identify the optimal trajectory, and adaptive parameters strategy [21] can use a feedback mechanism to achieve dynamic adaptation of parameters, thereby improving the quality of the current path solution. However, many algorithms perform well in dynamic environments, but when optimizing paths for fixed maps, they suffer from excessive exploration (such as unnecessary random walking) or lack a directional search mechanism for known obstacle structures, resulting in suboptimal path lengths.

The ALAs [22] represents an innovative swarm intelligence algorithm inspired by lemming behaviors: migratory patterns, burrowing activities, and food-searching strategies. The proposed method exhibits a straightforward architecture, implementation simplicity, and superior computational performance. It incorporates Brownian motion, random walk, and Levy flight mechanisms to achieve exploration-exploitation equilibrium, while adaptive parameter tuning progressively narrows the search range with iterations. However, the algorithm has certain limitations, including fixed parameters for search radius and jump probability, a lack of an adaptive mechanism to dynamically adjust based on population diversity or iteration progress, and an absence of a local fine-tuning strategy for the current optimal solution, which may lead to entrapment in local optima. Additionally, population diversity may decline rapidly with iterations, resulting in premature convergence. Especially when applied to mobile robot path planning problems, the ALAs algorithm further exposes the following issues: a fixed search radius makes it prone to generating collision paths in dense obstacle areas, a lack of heuristic guidance for known map structures (prioritizing search along obstacle contours), and a decline in population diversity causes path optimization to prematurely converge to suboptimal solutions.

The aforementioned algorithms have been widely applied in various research domains. However, when employed for path planning problems, they commonly face challenges such as an imbalance between exploration and exploitation, weak local search capability, and a tendency toward premature convergence—issues that also apply to the ALAs algorithm. To systematically address these limitations, this study addresses the optimal path planning problem in predefined static obstacle environments, with its primary challenges including: ensuring global optimal pathfinding within bounded constraint spaces, avoiding algorithmic entrapment in local optima within U-shaped or narrow obstacle regions, and striking a balance between computational efficiency and path optimality. To tackle these limitations, we introduce DMSALAs, which integrate a dynamic adaptive mechanism, hybrid Nelder–Mead optimization, and localized perturbation strategies to boost the search efficiency of the ALAs algorithm in path planning for mobile robots within static obstacle environments. The dynamic adaptive mechanism is designed to modulate the search granularity according to the proximity between the current trajectory and obstacles. The Hybrid Nelder–Mead optimization is employed for local smoothing of key turning points in the path. The small-range perturbation can conduct directional exploration for dead-end exits (such as the openings of U-shaped obstacles).

To verify the performance of the DMSALAs algorithm, we conducted a series of comprehensive experimental evaluations. First, to evaluate the efficacy of the three strategies, we performed ablation studies on the IEEE CECs 2017 benchmark, comparing results with six variant algorithms. Both IEEE CECs 2017 and CECs 2022 benchmarks were utilized to comprehensively assess DMSALAs against eight state-of-the-art algorithms spanning different categories. Furthermore, to explore the algorithm's performance in mobile robot path planning, we set the shortest path as the optimization goal. DMSALAs were tested on four maps with varying dimensions (10×10 , 20×20 , 30×30 , and 40×40), along with the real-world experimental setup constructed by our team. All experimental results effectively validated the superiority of the DMSALAs algorithm. This study provides a new solution for path length optimization based on prior maps in static environments, and its improved strategies specifically perform collaborative optimization for convergence speed and path optimality in structured obstacle spaces.

2. ALA Algorithm

ALAs primarily models key behaviors of lemmings, as depicted in Figure 1.

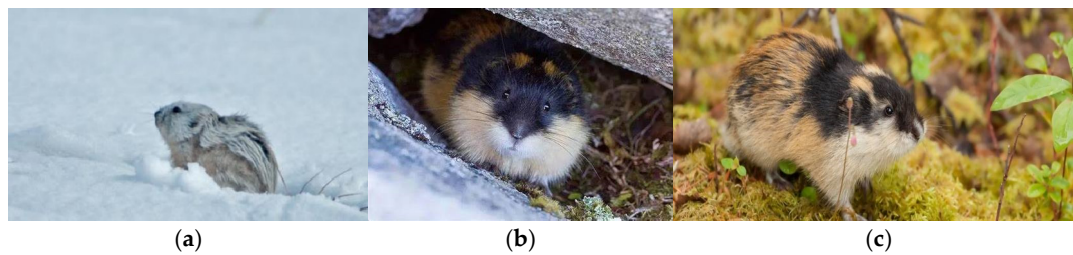


Figure 1. ALAs mainly simulation three behaviors of lemmings. (a) long-distance migration; (b) digging holes; (c) foraging behavior.

2.1. Initialization

In the initialization phase, similar to other algorithms, the algorithm generates an initial solution randomly within a predefined solution space Ω , N is the population size and D is a dimension. The formula is as follows:

$$\Omega_{i,j} = LB_j + rand \cdot (UB_j - LB_j) \quad (1)$$

where $i = 1, 2, \dots, N$, $j = 1, 2, \dots, D$, $rand \in (0, 1)$, UB_j and LB_j is the j^{th} upper and lower boundaries.

2.2. Long-Distance Migration (Exploration)

The formula of lemmings' long-distance migration is as follows:

$$\Omega_i(t+1) = \Omega_{best}(t) + F \cdot B \cdot (r \cdot (\Omega_{best}(t) - \Omega_i(t)) + (1-r) \cdot (\Omega_i(t) - \Omega_a(t))) \quad (2)$$

where $\Omega_i(t+1)$ and $\Omega_i(t)$ are the location of i^{th} search agent at $(t+1)^{th}$ and t^{th} iteration. $\Omega_{best}(t)$ is the current optimal solution. $r = 2 \cdot rand - 1$, as is shown in Equation (5). r is a random number uniformly distributed within the interval $[-1, 1]$. It determines, at each iteration, whether an individual is more inclined to move toward the best solution or to random exploration. When r approaches 1, the weight of the term $(\Omega_{best}(t) - \Omega_i(t))$ increases, enhancing the exploitation behavior and causing the individual to closely follow the current best solution. When r approaches -1 , the weight of the term $(\Omega_i(t) - \Omega_a(t))$ increases due to the growing value of $(1-r)$, thereby strengthening the exploration behavior and promoting random walks within the search space. In this way, r adaptively adjusts the balance between exploration and exploitation for each individual at every iteration, rather than relying on a fixed strategy. F is the flag for the search direction conversion, as is shown in Equation (3). This introduces a stochastic reverse mechanism into the search process. When $F = 1$ the individual moves in the original direction determined by the subsequent vector; when $F = -1$, the individual moves in the opposite direction. This mechanism effectively prevents individuals from blindly persisting in a single direction and enhances the likelihood of escaping local optima. It is a random vector representing Brownian motion; its value follows the probability density of a standard normal distribution and primarily controls the step size. B acts as a random multiplicative factor applied to the entire movement vector. A larger B value results in a significant jump, while a smaller B value leads to a fine-grained local move. B interacts with F the product $F \cdot B$ jointly determines the basic step size and direction of the current movement. This base random step is then combined with the exploration-exploitation directional vector weighted by r ,

ultimately determining the position update, as is shown in Equation (4). $\Omega_a(t)$ represents a randomly chosen individual within the population, a is an integer between 1 and N .

$$F = \begin{cases} 1 & \text{if } |2 \cdot \text{rand} + 1| = 1 \\ -1 & \text{if } |2 \cdot \text{rand} + 2| = 2 \end{cases} \quad (3)$$

$$B = \frac{1}{\sqrt{2 \cdot \pi}} \cdot \exp\left(-\frac{x^2}{2}\right) \quad (4)$$

2.3. Digging Holes (Exploration)

The secondary behavior of lemmings entails burrowing to achieve dual purposes: self-protection and food storage. The corresponding model is described below:

$$\Omega_i(t+1) = \Omega_i(t) + F \cdot L \cdot (\Omega_{best}(t) - \Omega_b(t)) \quad (5)$$

where $L = \text{rand} \cdot (1 + \sin(\frac{t}{2}))$, L is a randomly generated step size that periodically oscillates within the interval $[0, 2)$. The exploration step size expands during iterations when the sine function approaches 1, and contracts when the sine function approaches -1 ; however, due to the addition of 1, the minimum value remains non-negative. This mechanism helps balance global search capability with the avoidance of excessive jumps. $\Omega_b(t)$ represents a randomly chosen individual from the population, b is an integer between 1 and N . Through the coordinated interaction of L oscillating random step size, F stochastic direction reversal, and the base random vector, the model effectively implements a comprehensive, randomized exploration strategy with adaptive step-size variation.

2.4. Foraging (Exploitation)

The third behavior of lemmings involves foraging, wherein they move around within their burrows to locate food. The formula is as follows:

$$\Omega_i(t+1) = \Omega_{best}(t) + F \cdot \text{spiral} \cdot \text{rand} \cdot \Omega_i(t) \quad (6)$$

where *spiral* is for spiral foraging, as is shown in Equation (7). R is the search radius, as is shown in Equation (8). This equation generates a local search behavior characterized by a random spiral trajectory around the optimal solution through the coordinated action of the *spiral* function and F . R controls the scale of the spiral, while the combination of trigonometric functions determines its specific shape and phase. Together, these components enable the spiral function to dynamically generate a spiral vector whose amplitude is proportional to the distance and whose shape varies randomly, guiding individuals in conducting fine-grained and diverse exploration around the optimal solution.

$$\text{spiral} = R \cdot (\sin(2 \cdot \pi \cdot \text{rand}) + \cos(2 \cdot \pi \cdot \text{rand})) \quad (7)$$

$$R = \sqrt{\sum_{j=1}^D (\Omega_{best,j}(t) - \Omega_{i,j}(t))^2} \quad (8)$$

2.5. Exploration and Exploitation Transition Mechanism

The transition mechanism between exploration and exploitation is governed by the energy coefficient E .

$$E = 4 \cdot \arctan[1 - t/T_{\max}] \cdot \ln(1/\text{rand}) \quad (9)$$

In the initialization phase of ALAs, some control parameters were set. Afterwards, at each iteration, if the energy coefficient $E > 1$, ALAs enter the exploration phase, executing long-distance migration behavior or digging behavior. When $E \leq 1$, ALAs perform exploitation through foraging behavior or evading natural predators' behavior. These behaviors are employed by all search agents to update new candidate solutions and determine the currently found optimal solution. As the number of iterations increases, the value of E gradually decreases, allowing search agents to transition robustly from exploration to exploitation. Once the termination condition is satisfied, the procedure will terminate and output the optimal global solution. DMSALAs workflow is shown in Figure 2.

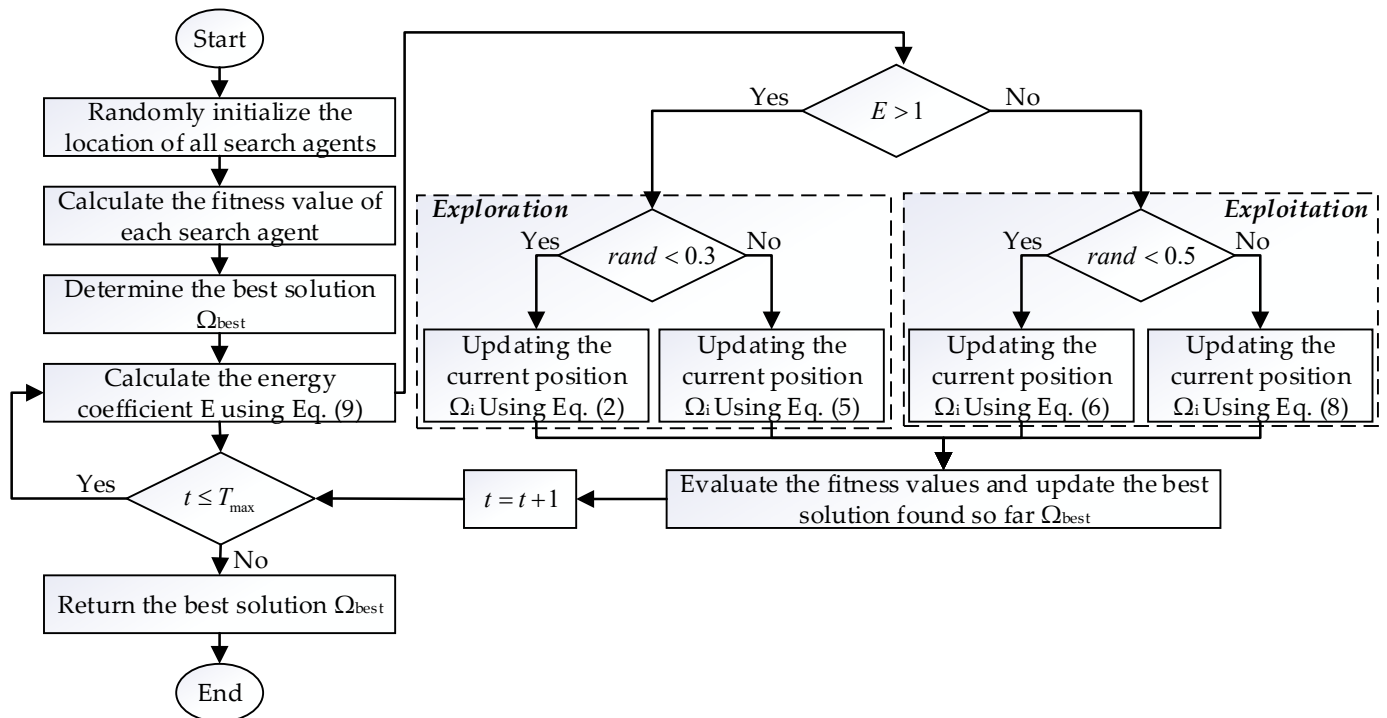


Figure 2. DMSALAs workflow.

3. Improvement Strategy

In this section, we implement a dynamic adaptive mechanism, a hybrid Nelder–Mead approach, and a localized perturbation strategy to improve the search performance of the ALAs algorithm.

3.1. Dynamic Adaptive Mechanism

The original algorithm features a fixed search radius and jump probability, resulting in poor robustness [23]. Consequently, we introduce an exponentially attenuating radius to enable the DMSALAs algorithm to dynamically adjust its search radius and jump probability [24,25], thereby achieving adaptability to different problem characteristics. The formula is presented as follows:

$$C = r_{\max} \cdot \exp(-\lambda \cdot t / T) \quad (10)$$

where the maximum search radius $r_{\max}$ is an Euclidean norm expressed as $r_{\max} = \text{norm}(UB - LB)$, where the minimum search radius is $r_{\min} = 0.01 \cdot r_{\max}$, and the radius decay coefficient is $\lambda = 7$. Let Fes and $Maxfes$ denote the current and maximum number of objective function evaluations.

In the initial stage, when $Fes \rightarrow 0$, $\exp(Fes) \rightarrow 1$, so the search radius $C \approx r_{\max}$. This allows the mobile robot to explore possible paths in a large area, and in the Late stage of algorithm search, $Fes \rightarrow Maxfes$, $(Fes/Maxfes) \rightarrow 1$, $\exp(-\lambda) \rightarrow r_{\min}$, causing $C \rightarrow 0$. The algorithm then prioritizes local fine-grained search, enabling the robot to fine-tune around the discovered suboptimal paths and converge near the optimal solution. The search radius C decays exponentially with the number of evaluations Fes , scanning the environment on a large scale using the map's diagonal length $r_{\max}$ in the early stage, and optimizing paths with $r_{\min}$ level precision in the late stage.

3.2. The Hybrid Nelder–Mead Method

To strengthen the algorithm's aptitude for local search, we incorporate the hybrid Nelder–Mead method into the ALAs algorithm [26,27]. The hybrid Nelder–Mead method enables adaptive local search and an elite retention strategy, replaces inferior solutions with superior ones, and prevents population degradation [28]. The pseudocode for the Hybrid Nelder–Mead Method is as follows (Algorithm 1):

Algorithm 1. Pseudocode for the Hybrid Nelder–Mead Method

```

IF current_Fes > 0.7 · max_Fes AND mod (current_Fes 5) = 0 THEN
// Perform local search using fminsearch
[new_position, new_fitness] = fminsearch(objective_function, current_position)
IF new_fitness < current_fitness THEN
// Update current individual
current_position = new_position
global_best_fitness = new_fitness
// Find worst individual in population
worst_index = index_of_max(fitness_population)
// Replace worst individual with new solution
population[worst_index] = new_position
fitness_population[worst_index] = new_fitness
END IF
END IF

```

where `worst_index` denotes the encoded address of the current suboptimal solution, `new_position` signifies the optimal outcome of the local search process, and `new_fitness` stands for the corresponding objective function value.

In the late phase of the algorithm, a hybrid Nelder–Mead approach is integrated, with the primary aim of enhancing the quality and convergence accuracy of mobile robot path planning results. Specifically, the Nelder–Mead simplex method is employed to conduct fine-grained searches on the current global optimal path (`current_position`), enhancing the algorithm's exploitation capability to improve convergence precision and the quality of the solved path. Meanwhile, population diversity is maintained by replacing the worst path in the population.

When `current_Fes > 0.7 · max_Fes`, it signifies that the algorithm has entered the terminal phase of iteration. Additionally, `mod (current_Fes, 5) = 0` means the current evaluation number `current_Fes` is divisible by 5. When both conditions `current_Fes > 0.7 · max_Fes` and `mod (current_Fes, 5) = 0` are satisfied, a gradient-free Nelder–Mead local search is performed once every five evaluations. This strategy substantially cuts down the algorithm's computational overhead while guaranteeing the quality of path planning results.

During local search, the `fminsearch` function is used, which takes the objective function `fobj`, the current global optimal path (`current_position`). This strategy substantially cuts

down the algorithm's computational overhead while guaranteeing the quality of path planning results. If the fitness value of the new path obtained through local enhancement search is superior to that of the current global optimal path ($\text{new_fitness} < \text{global_best_fitness}$), the global optimal path (current_position) and its fitness value Score are updated. This indicates that a better path for the mobile robot has been found, which is then taken as the new global optimal path.

Next, the new path and fitness value new_fitness obtained from fminsearch are used to replace the worst path in the population. Specifically, the fitness values of all paths within the population are computed using the max function to determine the index of the least optimal path. The coordinate values and fitness value of this worst path are then replaced by the new path P and new_fitness . This procedure iteratively discards suboptimal path solutions to enhance the overall quality of the population.

3.3. Small-Range Perturbation Strategy

To address the ALAs algorithms' propensity to converge to local optima, we devised a localized perturbation strategy [29] grounded in an evaluation of the current population diversity. The formula is presented as follows:

$$p = p_{\min} + (p_{\max} - p_{\min}) \cdot (d/r_{\max}) \quad (11)$$

The maximum jump probability is $p_{\max} = 0.5$, and the minimum jump probability is $p_{\min} = 0.1$. To characterize the distribution of the population within the solution space and identify premature convergence of the ALAs algorithm, the standard deviation of each column of X is calculated as $\text{diversity} = \text{std}(X)$. A larger standard deviation indicates greater individual differences in that dimension (better population diversity). The average standard deviation across all dimensions is $d = \text{mean}(\text{diversity})$, representing the overall average diversity of the population across all dimensions.

When the path distribution is concentrated ($d \rightarrow 0$), a 10% basic perturbation is retained; when the distribution is dispersed ($d \rightarrow r_{\max}$), the perturbation intensity is increased to 50%. This mechanism ensures that mobile robots can quickly locate feasible paths in complex environments and eliminate unnecessary jitter during the convergence phase.

When the path distribution is dispersed ($d \approx r_{\max}$), $p \rightarrow p_{\max}$ maintaining a 50% perturbation intensity to achieve a high jump probability. When the path distribution is concentrated ($d \rightarrow 0$), $p \rightarrow p_{\min}$, reducing the jump probability while retaining 10% basic perturbation to prevent the algorithm from falling into local optima. This mechanism ensures that mobile robots can both quickly locate feasible paths in complex environments and eliminate unnecessary jitter during the convergence phase. The formula is as follows:

$$\Omega_i(t+1) = \Omega_i(t) + 0.1 \cdot C \cdot \text{rand} \text{ if } \text{rand} \geq p \quad (12)$$

3.4. DMSALAs Flow

The workflow of DMSALAs in mobile robot path planning is as follows:

(a) Initialization Phase:

Set the initial global optimal fitness value $\text{Score} = \text{inf}$, indicating no optimal path solution has been found yet. Initialize the global optimal position as a zero vector $X_{\text{best}} = \text{zeros}(1, N)$, indicating that path fitness has not been evaluated. Initialize the convergence curve storage parameter $\text{Convergence} = []$ to record the algorithm's convergence characteristics during iterations.

(b) Initial Fitness Evaluation:

Initialize the evaluation counter $Fes = 0$ and the global optimal path cost Score. Decode each path in the population and compute its comprehensive performance using the fitness function $fobj$, which optimizes for path length.

(c) Dynamic Adaptive Adjustment Mechanism:

Formulate a dynamic adaptation parameter C to accommodate environmental dynamics encountered by the robot. Set a small-range perturbation factor p to enable local path refinement.

(d) Search Strategy Selection

Global Search ($E > 1$)

If $rand < 0.3$, initiate a hybrid search combining the global best solution and a random individual to leverage historical information and random exploration. If $rand \geq 0.3$, execute an adaptive step-size random walk for global sampling.

Local Search ($E \leq 1$)

If $rand < 0.5$ perform a constrained spiral search around the current position.

If $rand \geq 0.5$ further decide.

If $rand < p$ apply an adaptive Levy flight strategy to perturb the current best path X_{best} for escaping local optima.

If $rand \geq p$ execute a small-range perturbation for fine-tuning the path.

(e) New Path Evaluation:

Compute the fitness of new paths generated by both global and local searches using the objective function $fobj$. Apply a greedy strategy to compare new paths with the current best path, updating the solution if the new path is superior.

(f) Nelder–Mead Local Enhancement:

When $current_Fes > 0.7 \cdot \max_Fes$ AND $\text{mod}(current_Fes, 5) = 0$ (after 70% of evaluations and every 5 iterations).

Invoke the `fminsearch` function with parameters $fobj$, $current_position$ to perform gradient-free Nelder–Mead local search.

Substitute the least optimal individual in the population with the new trajectory p and its fitness value to preserve diversity.

(g) Termination and Output:

Upon completion of all iterations, the global optimal trajectory and its corresponding fitness value are recorded, followed by the output of the mobile robot's optimal path plan.

3.5. Time Complexity Analysis

To analyze the time complexity of the DMSALAs algorithm, we adopt a standard methodology. Specifically, we use the first problem from the CEC 2017 benchmark suite, which has a dimensionality of 10. For consistency across all algorithms being compared, we set a maximum of 1000 iterations. The equation is evaluated using T_0 given below:

```
tic;
for i = 1 : 1000000
    x = 0.55 + double(i);
    x = x + x; x = x/2; x = x * x; x = sqrt(x); x = log(x); x = exp(x); x = x/(x + 2);
end
toc;
```

T_1 is the time taken by the first problem from the CEC 2017 benchmark suite (called CEC 2017 F1) for a single run, T_2 is the time taken by an algorithm for evaluating CEC

2017 F1 for five runs. The ratio $(T_1 - T_2)/T_0$ reflects the relative difference between the single—run time and the time for five runs of an algorithm, normalized by the benchmark time T_0 . A larger ratio indicates a more significant discrepancy between these two-time metrics relative to T_0 . Thereby showcasing the variations in time complexity among different algorithms when solving this problem and highlighting the superior time-related performance and competitiveness of a specific algorithm compared to others.

For the DMSALAs algorithm, its ratio value is 5.089566. Among the tested algorithms, this places it in second position, just behind the GWOs algorithm. Notably, it surpasses the GAs and SFOAs algorithms. The time complexity analysis is presented in Table 1.

Table 1. Time Complexity Analysis.

	Function	Dimension	Algorithm	T_0	T_1	T_2	$(T_1 - T_2)/T_0$
CEC2017	F1	10	DMSALAs	0.045560	0.814482	0.582601	5.089566
			GWO	0.045560	0.626984	0.375094	5.528745
			GA	0.045560	0.782276	0.610065	3.779854
			SFOA	0.045560	0.689711	0.549774	3.071483

3.6. Parameter_Sensitivity_Analysis

To conduct parameter sensitivity analysis, we selected five test functions (F1, F7, F15, F23, and F30) from the CEC 2017 test suite, with a dimension of $\text{dim} = 10$ and a maximum fitness evaluation count of $\text{Maxfex} = 1000 \cdot \text{dim}$. Each function was independently run for 30 rounds. The values of the three parameters, namely the radius decay coefficient $\lambda = 1, 3, 5, 7, 10$ in the Dynamic Adaptive Mechanism, the maximum jump probability $p_{\max} = 0.1, 0.3, 0.5, 0.7, 0.9$, and the population size $N = 20, 30, 40, 50, 60$, were set differently. The average solutions obtained by the DMSALAs algorithm on the five functions under various combinations of λ , $p_{\max}$, and N are presented in Table 2 and Figure 3. The optimal value is highlighted with a red dot in Figure 3.

Table 2. The average solutions obtained by the DMSALAs on five functions under various combinations of λ , $p_{\max}$, and N .

λ Values					$p_{\max}$ Values					N Values				
1	3	5	7	10	0.1	0.3	0.5	0.7	0.9	20	30	40	50	60
1678.34	1682.84	1678.24	1677.95	1678.99	1681.26	1673.98	1670.95	1679.99	1686.48	1678.28	1672.61	1678.60	1673.34	1674.91

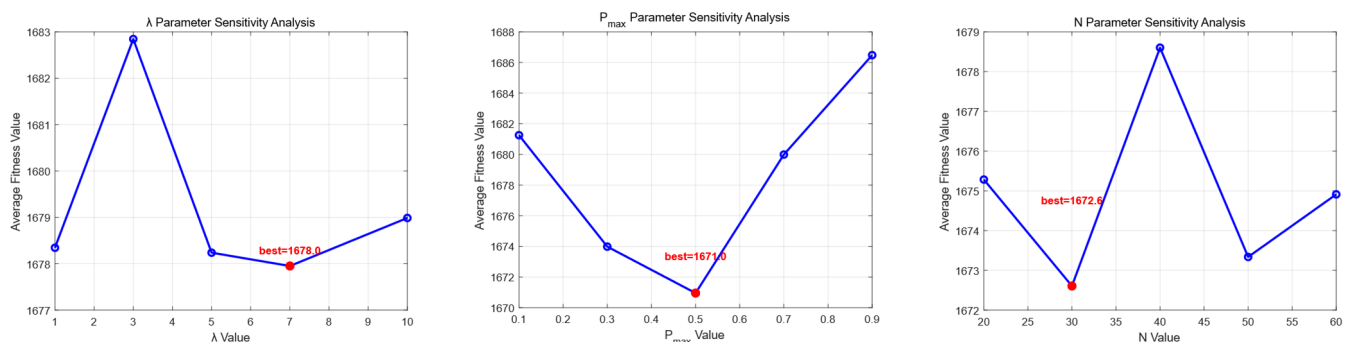


Figure 3. The solutions of the DMSALAs on 5 functions when λ , $p_{\max}$, and N take different values.

The sensitivity analysis of the average fitness values under different parameter settings through system testing shows that when the λ parameter varies within the range of 1 to 10, DMSALAs' performance remains relatively stable. The best performance is achieved when $\lambda = 7$, with an average fitness value of 1677.95, which is not significantly different from other values (fluctuating between 1677.95 and 1682.84), indicating that the λ parameter has

a minor impact on the algorithm's performance within the tested range and demonstrates good robustness. The $p_{\max}$ parameter exhibits a distinct sensitivity characteristic. As $p_{\max}$ increases from 0.1 to 0.5, the average fitness value improves from 1681.26 to 1670.95, representing a performance enhancement of approximately 0.6%. However, when $p_{\max}$ further increases to 0.9, the performance drops to 1686.48. This suggests that there is an optimal $p_{\max}$ value (0.5), and exceeding this range leads to performance deterioration. The sensitivity analysis of the N parameter indicates that the algorithm's performance is optimal at $N = 30$ (1672.61). As N increases from 20 to 60, the performance shows an initial improvement followed by slight fluctuations, but the overall change is relatively small (1672.61–1678.60), indicating that the N parameter has a limited impact on the algorithm's performance within a reasonable range. The results of the sensitivity analysis suggest that all three parameters have some influence on the algorithm's performance, but the degree and pattern of their influence vary. The DMSALAs can maintain the best performance when $\lambda = 0.7$, $p_{\max} = 0.5$, and $N = 30$.

4. Experimental Verifications

To verify the performance of the DMSALAs algorithm, we conducted a series of comprehensive experimental evaluations, encompassing an ablation study on the IEEE CEC 2017 benchmark and comparative assessments on both IEEE CEC 2017 and CEC 2022. To ensure fairness in algorithm evaluation, we employed the maximum number of fitness evaluations as the primary metric for assessing algorithm efficiency. We set $T = 1000$ and $N = 30$. To maintain consistency with the majority of existing studies and enable fair comparisons, the experiments primarily focus on the standard 30D setting on the IEEE CEC 2017 and 10D on the IEEE CEC 2022. This dimensionality is widely recognized as a benchmark for evaluating algorithm performance, as it strikes a favorable balance between problem complexity and computational cost.

All experimental results reported in this section are presented as statistical averages and standard deviations derived from 30 independent trial runs. The comparative analysis employs a sign test notation system, where '+' denotes superior performance of a comparison algorithm over DMSALAs on a given test function, '−' indicates inferior performance relative to DMSALAs, and '=' signifies statistically equivalent results. For a comprehensive performance evaluation, we implement Friedman's test analysis, incorporating both average ranking scores and ordinal rankings. Before conducting the Friedman test, the Lilliefors normality test was performed on all results, and the corresponding p -values are presented in Tables 3–5. These dual metrics provide a quantitative assessment of each algorithm's overall optimization capability across the benchmark suite. To enhance the credibility of our experimental results, we conducted the Wilcoxon rank-sum test for statistical significance analysis.

Table 3. DMSALAs performance compared with its variants on IEEE CECs 2017.

		DMSALAs	ALAs1	ALAs2	ALAs3	ALAs4	ALAs5	ALAs6
CEC2017-F1	Mean	1.00×10^2	4.82×10^3	1.00×10^2	8.11×10^2	1.00×10^2	4.33×10^3	1.00×10^2
	Std	2.45×10^{-9}	4.05×10^3	3.39×10^{-9}	1.10×10^3	4.08×10^{-9}	4.09×10^3	4.02×10^{-9}
	p -value	1.50×10^{-2}	7.25×10^{-3}	1.61×10^{-3}	1.00×10^{-3}	4.33×10^{-2}	3.78×10^{-2}	3.81×10^{-2}
CEC2017-F3	Mean	3.00×10^2	3.15×10^2	3.00×10^2	3.00×10^2	3.00×10^2	3.04×10^2	3.00×10^2
	Std	5.36×10^{-10}	4.10×10^1	3.66×10^{-14}	4.56×10^{-9}	2.27×10^{-13}	2.00×10^1	1.29×10^{-13}
	p -value	1.02×10^{-2}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	3.47×10^{-2}	4.51×10^{-3}	1.00×10^{-3}
CEC2017-F4	Mean	4.00×10^2	4.13×10^2	4.00×10^2	4.00×10^2	4.00×10^2	4.04×10^2	4.00×10^2
	Std	1.01×10^0	2.01×10^1	1.72×10^{-13}	8.14×10^{-1}	1.01×10^0	1.55×10^0	1.01×10^0
	p -value	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.47×10^{-2}	1.00×10^{-3}
CEC2017-F5	Mean	5.21×10^2	5.19×10^2	5.22×10^2	5.14×10^2	5.16×10^2	5.20×10^2	5.14×10^2
	Std	1.00×10^1	6.03×10^0	6.44×10^0	4.98×10^0	1.00×10^1	9.23×10^0	4.41×10^0
	p -value	3.47×10^{-1}	3.87×10^{-1}	1.91×10^{-2}	3.18×10^{-1}	4.56×10^{-2}	1.00×10^{-1}	1.66×10^{-1}

Table 3. Cont.

		DMSALAs	ALAs1	ALAs2	ALAs3	ALAs4	ALAs5	ALAs6
CEC2017-F6	Mean	6.00×10^2	6.00×10^2	6.00×10^2	6.00×10^2	6.01×10^2	6.00×10^2	6.00×10^2
	Std	7.83×10^{-1}	5.20×10^{-1}	1.33×10^{-1}	2.17×10^{-1}	1.80×10^0	4.46×10^{-2}	3.11×10^{-1}
	p-value	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}
CEC2017-F7	Mean	7.22×10^2	7.29×10^2	7.27×10^2	7.22×10^2	7.31×10^2	7.29×10^2	7.36×10^2
	Std	1.03×10^1	8.69×10^0	5.64×10^0	4.46×10^0	1.10×10^1	1.05×10^1	7.65×10^0
	p-value	2.45×10^{-1}	5.00×10^{-1}	5.00×10^{-1}	2.64×10^{-1}	5.00×10^{-1}	4.62×10^{-1}	7.40×10^{-2}
CEC2017-F8	Mean	8.18×10^2	8.19×10^2	8.16×10^2	8.13×10^2	8.22×10^2	8.20×10^2	8.16×10^2
	Std	9.68×10^0	9.44×10^0	6.32×10^0	4.89×10^0	9.68×10^0	7.74×10^0	6.91×10^0
	p-value	1.26×10^{-2}	5.00×10^{-1}	6.32×10^{-1}	1.52×10^{-1}	1.70×10^{-2}	4.52×10^{-1}	6.91×10^{-1}
CEC2017-F9	Mean	9.00×10^2	9.08×10^2	9.00×10^2	9.02×10^2	9.01×10^2	9.02×10^2	9.00×10^2
	Std	4.98×10^0	2.64×10^0	6.38×10^{-1}	8.40×10^{-2}	1.40×10^1	5.56×10^0	2.03×10^{-1}
	p-value	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}
CEC2017-F10	Mean	1.57×10^3	1.81×10^3	1.58×10^3	1.62×10^3	1.63×10^3	1.85×10^3	1.60×10^3
	Std	1.89×10^2	2.31×10^2	1.80×10^2	2.46×10^2	1.67×10^2	2.41×10^2	2.00×10^2
	p-value	2.37×10^{-2}	2.25×10^{-1}	1.00×10^{-1}	9.05×10^{-3}	1.77×10^{-1}	2.49×10^{-1}	3.48×10^{-2}
CEC2017-F11	Mean	1.10×10^3	1.11×10^3	1.11×10^3	1.12×10^3	1.12×10^3	1.11×10^3	1.11×10^3
	Std	8.64×10^0	4.17×10^0	3.99×10^0	2.73×10^0	9.81×10^0	4.71×10^1	3.96×10^0
	p-value	1.00×10^{-3}	3.73×10^{-2}	4.54×10^{-2}	3.39×10^{-2}	1.00×10^{-3}	1.00×10^{-3}	3.67×10^{-2}
CEC2017-F12	Mean	1.54×10^3	2.74×10^4	1.65×10^3	1.69×10^3	3.13×10^3	3.15×10^4	1.57×10^3
	Std	1.93×10^2	2.03×10^4	3.13×10^2	1.72×10^2	4.62×10^3	2.17×10^4	2.80×10^2
	p-value	5.00×10^{-1}	3.10×10^{-2}	4.08×10^{-2}	5.00×10^{-1}	1.00×10^{-3}	3.64×10^{-3}	8.22×10^{-2}
CEC2017-F13	Mean	1.31×10^3	1.79×10^3	1.84×10^3	1.38×10^3	1.36×10^3	1.38×10^3	1.31×10^3
	Std	1.18×10^2	3.98×10^2	1.81×10^1	2.39×10^2	1.32×10^2	3.26×10^2	9.25×10^0
	p-value	1.00×10^{-3}	1.03×10^{-2}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	3.36×10^{-1}	1.00×10^{-3}
CEC2017-F14	Mean	1.42×10^3	1.44×10^3	1.44×10^3	1.44×10^3	1.44×10^3	1.42×10^3	1.42×10^3
	Std	1.14×10^1	1.02×10^1	9.13×10^0	7.75×10^0	9.31×10^0	9.41×10^0	7.48×10^0
	p-value	2.96×10^{-1}	4.63×10^{-2}	1.00×10^{-3}	1.00×10^{-3}	5.00×10^{-1}	2.97×10^{-1}	1.00×10^{-3}
CEC2017-F15	Mean	1.51×10^3	1.62×10^3	1.51×10^3	1.65×10^3	1.60×10^3	1.61×10^3	1.51×10^3
	Std	1.40×10^2	6.74×10^1	1.70×10^1	4.11×10^0	7.36×10^1	5.63×10^1	8.18×10^0
	p-value	1.00×10^{-3}	1.50×10^{-1}	1.00×10^{-3}	1.24×10^{-1}	4.29×10^{-2}	1.59×10^{-1}	2.61×10^{-2}
CEC2017-F16	Mean	1.64×10^3	1.69×10^3	1.68×10^3	1.66×10^3	1.68×10^3	1.66×10^3	1.65×10^3
	Std	6.57×10^1	8.47×10^1	5.55×10^1	8.03×10^1	9.01×10^1	9.03×10^1	6.35×10^1
	p-value	1.00×10^{-3}	2.38×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}
CEC2017-F17	Mean	1.73×10^3	1.76×10^3	1.77×10^3	1.78×10^3	1.77×10^3	1.74×10^3	1.74×10^3
	Std	4.86×10^1	4.87×10^1	9.68×10^0	2.26×10^1	4.71×10^1	5.29×10^1	2.79×10^1
	p-value	1.00×10^{-3}	1.00×10^{-3}	4.54×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}
CEC2017-F18	Mean	2.44×10^3	3.82×10^3	1.84×10^3	1.82×10^3	2.45×10^3	4.83×10^3	1.84×10^3
	Std	1.12×10^3	4.31×10^3	1.64×10^1	4.93×10^0	1.20×10^3	6.81×10^3	2.13×10^1
	p-value	1.00×10^{-3}	1.00×10^{-3}	3.39×10^{-1}	1.23×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}
CEC2017-F19	Mean	1.94×10^3	2.00×10^3	1.91×10^3	1.90×10^3	1.95×10^3	1.95×10^3	1.91×10^3
	Std	4.30×10^1	1.71×10^2	3.60×10^0	1.43×10^0	1.39×10^2	4.00×10^1	3.22×10^0
	p-value	1.00×10^{-3}	1.00×10^{-3}	2.03×10^{-2}	5.00×10^{-1}	1.00×10^{-3}	1.00×10^{-3}	5.29×10^{-2}
CEC2017-F20	Mean	2.04×10^3	2.07×10^3	2.04×10^3	2.08×10^3	2.10×10^3	2.05×10^3	2.03×10^3
	Std	6.06×10^1	6.55×10^1	4.00×10^1	3.59×10^1	7.10×10^1	4.54×10^1	2.63×10^1
	p-value	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	6.93×10^{-2}	1.00×10^{-3}	1.00×10^{-3}
CEC2017-F21	Mean	2.23×10^3	2.23×10^3	2.25×10^3	2.24×10^3	2.21×10^3	2.23×10^3	2.25×10^3
	Std	5.02×10^1	4.88×10^1	5.82×10^1	5.58×10^1	3.10×10^1	5.20×10^1	5.89×10^1
	p-value	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}
CEC2017-F22	Mean	2.30×10^3	2.44×10^3	2.43×10^3	2.30×10^3	2.30×10^3	2.30×10^3	2.30×10^3
	Std	1.34×10^1	1.30×10^1	6.23×10^{-1}	2.25×10^1	4.32×10^2	4.25×10^2	7.33×10^{-1}
	p-value	1.00×10^{-3}	1.00×10^{-3}	5.00×10^{-1}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.01×10^{-1}
CEC2017-F23	Mean	2.61×10^3	2.62×10^3	2.62×10^3	2.62×10^3	2.62×10^3	2.62×10^3	2.61×10^3
	Std	7.46×10^0	1.30×10^1	4.83×10^0	7.30×10^0	7.32×10^0	8.43×10^0	5.63×10^0
	p-value	3.82×10^{-3}	7.62×10^{-3}	5.00×10^{-1}	3.45×10^{-3}	4.71×10^{-2}	1.34×10^{-2}	2.39×10^{-1}
CEC2017-F24	Mean	2.73×10^3	2.74×10^3	2.73×10^3	2.75×10^3	2.75×10^3	2.73×10^3	2.74×10^3
	Std	6.38×10^1	4.61×10^1	6.38×10^1	5.81×10^0	4.74×10^1	6.30×10^1	4.88×10^0
	p-value	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	5.00×10^{-1}	1.00×10^{-3}	1.00×10^{-3}	2.77×10^{-1}
CEC2017-F25	Mean	2.92×10^3	2.92×10^3	2.92×10^3	2.93×10^3	2.93×10^3	2.93×10^3	2.92×10^3
	Std	3.27×10^1	6.66×10^1	2.27×10^1	2.31×10^1	6.80×10^1	2.64×10^1	2.39×10^1
	p-value	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}
CEC2017-F26	Mean	2.94×10^3	3.13×10^3	3.07×10^3	3.06×10^3	3.11×10^3	3.03×10^3	2.94×10^3
	Std	2.91×10^2	4.06×10^2	1.07×10^2	3.86×10^2	3.14×10^2	3.10×10^2	1.63×10^2
	p-value	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}
CEC2017-F27	Mean	3.09×10^3	3.09×10^3	3.09×10^3	3.09×10^3	3.09×10^3	3.09×10^3	3.09×10^3
	Std	2.22×10^0	9.01×10^0	9.52×10^0	8.71×10^0	2.92×10^0	2.85×10^0	2.22×10^0
	p-value	1.07×10^{-1}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	5.00×10^{-1}	1.00×10^{-3}	1.00×10^{-3}
CEC2017-F28	Mean	3.17×10^3	3.41×10^3	3.33×10^3	3.34×10^3	3.18×10^3	3.18×10^3	3.17×10^3
	Std	5.64×10^1	1.63×10^2	5.46×10^1	2.29×10^2	6.29×10^1	1.47×10^2	6.08×10^1
	p-value	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}
CEC2017-F29	Mean	3.20×10^3	3.21×10^3	3.17×10^3	3.17×10^3	3.19×10^3	3.19×10^3	3.17×10^3
	Std	6.42×10^1	5.18×10^1	3.33×10^1	4.10×10^1	4.40×10^1	4.91×10^1	2.96×10^1
	p-value	1.00×10^{-3}	1.03×10^{-1}	6.05×10^{-3}	1.00×10^{-3}	2.04×10^{-1}	8.48×10^{-3}	5.16×10^{-2}

Table 3. Cont.

		DMSALAs	ALAs1	ALAs2	ALAs3	ALAs4	ALAs5	ALAs6
CEC2017-F30	Mean	3.29×10^3	2.95×10^5	3.33×10^3	6.52×10^5	3.31×10^3	3.11×10^5	3.30×10^3
	Std	8.67×10^1	6.75×10^5	1.68×10^2	1.08×10^6	4.76×10^1	5.80×10^5	5.73×10^1
	<i>p</i> -value	1.46×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	2.36×10^{-2}
(+ / − / =) Statistic		(22/1/6)	(3/11/15)	(10/4/15)	(11/7/11)	(6/9/14)	(5/5/19)	(17/2/10)
Friedman mean		1.48	3.52	2.34	2.55	2.90	2.97	1.62
Ranking		1	7	3	4	5	6	2

The best mean results and standard deviation values are highlighted in bold. The *p*-value is the result of the Lilliefors normality test.

Table 4. DMSALAs performance compared with 8 advanced algorithms on IEEE CEC 2017.

		DMSALAs	ALAs	WMAs	PSOs	SFOAs	BKAs	GOOSEs	BWOs	GWOs
CEC2017-F1	Mean	1.00×10^2	1.65×10^3	3.34×10^3	1.78×10^3	2.51×10^9	4.28×10^5	2.75×10^3	7.35×10^9	3.92×10^7
	Std	9.25×10^{-9}	1.86×10^3	3.15×10^3	1.99×10^3	1.8×10^9	2.55×10^5	2.64×10^3	2.56×10^9	1.99×10^8
	<i>p</i> -value	1.66×10^{-2}	1.00×10^{-3}	2.56×10^{-2}	1.00×10^{-3}	1.60×10^{-2}	1.18×10^{-3}	4.54×10^{-2}	3.48×10^{-2}	1.00×10^{-3}
CEC2017-F3	Mean	3.00×10^2	3.00×10^2	3.00×10^2	3.00×10^2	3.20×10^4	9.68×10^3	3.21×10^2	9.51×10^3	2.56×10^3
	Std	6.24×10^{-10}	8.18×10^{-9}	3.50×10^{-14}	1.02×10^{-3}	1.64×10^4	7.72×10^3	7.71×10^1	1.84×10^3	2.30×10^3
	<i>p</i> -value	2.92×10^{-2}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	5.00×10^{-1}	3.23×10^{-3}	1.00×10^{-3}	2.97×10^{-2}	1.00×10^{-3}
CEC2017-F4	Mean	4.00×10^2	4.00×10^2	4.00×10^2	4.07×10^2	5.76×10^2	4.06×10^2	4.10×10^2	7.58×10^2	4.09×10^2
	Std	7.28×10^{-1}	4.78×10^{-1}	9.05×10^{-3}	1.21×10^1	1.10×10^2	4.41×10^0	2.01×10^1	1.32×10^2	6.13×10^0
	<i>p</i> -value	1.00×10^{-3}	3.49×10^{-2}	1.73×10^{-2}	1.00×10^{-3}	1.85×10^{-2}	6.12×10^{-3}	1.00×10^{-3}	5.00×10^{-1}	1.00×10^{-3}
CEC2017-F5	Mean	5.20×10^2	5.15×10^2	5.11×10^2	5.47×10^2	5.84×10^2	5.30×10^2	5.84×10^2	5.86×10^2	5.18×10^2
	Std	8.08×10^0	5.91×10^0	4.30×10^0	1.34×10^1	2.00×10^1	1.01×10^1	2.74×10^1	1.15×10^1	1.09×10^1
	<i>p</i> -value	1.96×10^{-2}	5.00×10^{-1}	4.25×10^{-2}	3.37×10^{-1}	4.41×10^{-3}	5.00×10^{-1}	2.41×10^{-2}	5.00×10^{-3}	1.92×10^{-2}
CEC2017-F6	Mean	6.00×10^2	6.00×10^2	6.02×10^2	6.12×10^2	6.50×10^2	6.12×10^2	6.57×10^2	6.47×10^2	6.01×10^2
	Std	1.43×10^0	9.82×10^{-1}	2.83×10^0	1.02×10^1	1.49×10^1	5.36×10^0	1.13×10^1	6.29×10^0	1.76×10^0
	<i>p</i> -value	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.39×10^{-4}	1.50×10^{-2}	5.00×10^{-1}	5.00×10^{-1}	2.85×10^{-2}	1.00×10^{-3}
CEC2017-F7	Mean	7.33×10^2	7.27×10^2	7.26×10^2	7.24×10^2	8.42×10^2	7.67×10^2	1.00×10^3	7.98×10^2	7.32×10^2
	Std	1.04×10^1	8.36×10^0	8.58×10^0	6.68×10^0	4.96×10^1	2.21×10^1	1.91×10^2	1.02×10^1	9.33×10^0
	<i>p</i> -value	4.15×10^{-2}	2.91×10^{-2}	3.23×10^{-2}	4.01×10^{-2}	1.18×10^{-3}	1.75×10^{-2}	1.00×10^{-3}	9.56×10^{-2}	5.00×10^{-1}
CEC2017-F8	Mean	8.22×10^2	8.17×10^2	8.12×10^2	8.20×10^2	8.78×10^2	8.42×10^2	8.60×10^2	8.52×10^2	8.14×10^2
	Std	1.03×10^1	6.80×10^0	5.89×10^0	9.08×10^0	1.95×10^1	1.46×10^1	2.40×10^1	6.15×10^0	5.86×10^0
	<i>p</i> -value	1.71×10^{-2}	5.00×10^{-1}	3.08×10^{-2}	5.00×10^{-1}	3.40×10^{-1}	4.02×10^{-1}	2.70×10^{-2}	1.66×10^{-1}	9.11×10^{-2}
CEC2017-F9	Mean	9.00×10^2	9.03×10^2	9.04×10^2	9.34×10^2	2.48×10^3	1.35×10^3	2.30×10^3	1.42×10^3	9.08×10^2
	Std	1.81×10^{-1}	4.45×10^0	8.18×10^0	1.31×10^2	1.00×10^3	2.37×10^2	6.68×10^2	9.67×10^1	1.37×10^1
	<i>p</i> -value	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.65×10^{-2}	1.50×10^{-2}	1.53×10^{-2}	3.47×10^{-1}	1.00×10^{-3}
CEC2017-F10	Mean	1.71×10^3	1.67×10^3	1.59×10^3	1.90×10^3	2.74×10^3	1.74×10^3	2.52×10^3	2.46×10^3	1.63×10^3
	Std	1.62×10^2	2.57×10^2	2.42×10^2	3.00×10^2	2.32×10^2	1.89×10^2	3.61×10^2	1.39×10^2	2.98×10^2
	<i>p</i> -value	4.75×10^{-3}	4.61×10^{-2}	3.37×10^{-2}	4.00×10^{-1}	4.77×10^{-2}	2.83×10^{-2}	4.50×10^{-2}	5.00×10^{-3}	1.59×10^{-2}
CEC2017-F11	Mean	1.10×10^3	1.11×10^3	1.15×10^3	1.14×10^3	2.11×10^3	1.15×10^3	1.21×10^3	2.02×10^3	1.14×10^3
	Std	2.91×10^0	9.38×10^0	4.88×10^1	2.07×10^1	1.23×10^3	7.12×10^1	6.06×10^1	5.64×10^2	3.05×10^1
	<i>p</i> -value	2.09×10^{-2}	1.51×10^{-2}	1.16×10^{-2}	2.78×10^{-2}	1.00×10^{-3}	1.00×10^{-3}	2.26×10^{-1}	1.54×10^{-2}	1.00×10^{-3}
CEC2017-F12	Mean	2.01×10^3	1.75×10^3	1.71×10^4	1.62×10^4	7.00×10^7	5.14×10^5	2.65×10^5	4.73×10^7	8.07×10^5
	Std	1.48×10^3	2.23×10^2	1.49×10^4	1.38×10^4	1.08×10^8	1.47×10^6	2.13×10^5	3.72×10^7	7.83×10^5
	<i>p</i> -value	1.00×10^{-3}	2.80×10^{-2}	2.80×10^{-3}	1.42×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.85×10^{-3}	7.33×10^{-3}
CEC2017-F13	Mean	1.32×10^3	1.34×10^3	7.23×10^3	8.74×10^3	2.04×10^6	3.16×10^3	1.85×10^4	9.66×10^5	1.25×10^4
	Std	1.12×10^1	6.13×10^1	7.61×10^3	7.19×10^3	5.52×10^6	3.91×10^3	1.36×10^4	1.01×10^6	6.19×10^3
	<i>p</i> -value	1.00×10^{-3}	1.93×10^{-2}	1.00×10^{-3}	9.45×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.86×10^{-2}	1.00×10^{-3}	1.97×10^{-2}
CEC2017-F14	Mean	1.42×10^3	1.43×10^3	1.47×10^3	2.16×10^3	1.06×10^4	1.58×10^3	4.78×10^3	1.66×10^3	3.58×10^3
	Std	9.60×10^0	1.18×10^1	3.57×10^1	1.20×10^3	8.59×10^3	9.21×10^1	4.46×10^3	7.51×10^1	1.84×10^3
	<i>p</i> -value	4.16×10^{-2}	1.00×10^{-3}	1.44×10^{-2}	1.00×10^{-3}	1.48×10^{-2}	2.32×10^{-2}	1.00×10^{-3}	3.22×10^{-2}	1.00×10^{-3}
CEC2017-F15	Mean	1.63×10^3	1.51×10^3	1.71×10^3	2.78×10^3	3.55×10^4	2.70×10^3	2.12×10^4	7.50×10^3	5.70×10^3
	Std	7.14×10^1	2.87×10^0	1.48×10^2	1.53×10^3	3.31×10^4	1.19×10^3	2.05×10^4	2.17×10^3	4.66×10^3
	<i>p</i> -value	1.56×10^{-1}	3.22×10^{-1}	1.25×10^{-1}	1.00×10^{-3}	1.56×10^{-2}	1.00×10^{-3}	7.95×10^{-3}	9.32×10^{-3}	3.22×10^{-3}
CEC2017-F16	Mean	1.63×10^3	1.66×10^3	1.71×10^3	1.86×10^3	2.09×10^3	1.67×10^3	2.25×10^3	2.03×10^3	1.71×10^3
	Std	4.28×10^1	7.35×10^1	1.16×10^2	1.27×10^2	1.70×10^2	7.21×10^1	2.40×10^2	8.49×10^1	1.10×10^2
	<i>p</i> -value	1.00×10^{-3}	1.00×10^{-3}	3.85×10^{-3}	1.08×10^{-2}	2.68×10^{-1}	1.00×10^{-3}	1.37×10^{-1}	3.53×10^{-2}	1.76×10^{-3}
CEC2017-F17	Mean	1.74×10^3	1.77×10^3	1.78×10^3	1.77×10^3	1.90×10^3	1.75×10^3	2.12×10^3	1.82×10^3	1.76×10^3
	Std	4.88×10^1	2.37×10^1	4.78×10^1	3.77×10^1	9.56×10^1	1.21×10^1	2.17×10^2	2.21×10^1	3.16×10^1
	<i>p</i> -value	3.37×10^{-3}	1.00×10^{-3}	2.28×10^{-3}	1.68×10^{-3}	9.43×10^{-3}	3.03×10^{-2}	1.50×10^{-3}	5.00×10^{-1}	1.00×10^{-3}

Table 4. Cont.

		DMSALAs	ALAs	WMAs	PSOs	SFOAs	BKAs	GOOSEs	BWOs	GWOs
CEC2017-F18	Mean	2.65×10^3	1.83×10^3	1.62×10^4	1.38×10^4	2.69×10^6	8.53×10^3	1.70×10^4	5.65×10^6	2.76×10^4
	Std	3.36×10^3	2.75×10^1	1.17×10^4	1.17×10^4	5.47×10^6	1.04×10^4	1.44×10^4	5.29×10^6	1.70×10^4
	p-value	1.00×10^{-3}	1.00×10^{-3}	2.48×10^{-2}	3.81×10^{-2}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	4.17×10^{-3}	1.00×10^{-3}
CEC2017-F19	Mean	1.90×10^3	1.94×10^3	2.06×10^3	4.40×10^3	1.04×10^5	2.40×10^3	9.01×10^3	4.87×10^4	2.69×10^4
	Std	4.41×10^1	1.32×10^0	1.48×10^2	3.28×10^3	1.63×10^5	2.21×10^2	8.03×10^3	2.85×10^4	6.64×10^4
	p-value	1.00×10^{-3}	5.00×10^{-1}	2.30×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	3.37×10^{-3}	1.00×10^{-3}	1.51×10^{-2}	1.00×10^{-3}
CEC2017-F20	Mean	2.02×10^3	2.06×10^3	2.06×10^3	2.11×10^3	2.27×10^3	2.05×10^3	2.29×10^3	2.22×10^3	2.08×10^3
	Std	1.02×10^1	4.88×10^1	5.45×10^1	7.21×10^1	1.09×10^2	2.75×10^1	1.39×10^2	4.88×10^1	5.71×10^1
	p-value	1.00×10^{-3}	1.76×10^{-2}	1.00×10^{-3}	1.76×10^{-3}	1.50×10^{-1}	1.00×10^{-3}	1.00×10^{-3}	5.00×10^{-1}	1.00×10^{-3}
CEC2017-F21	Mean	2.25×10^3	2.25×10^3	2.31×10^3	2.30×10^3	2.26×10^3	2.29×10^3	2.37×10^3	2.27×10^3	2.31×10^3
	Std	5.85×10^1	5.86×10^1	2.93×10^1	6.62×10^1	2.29×10^1	5.48×10^1	4.97×10^1	2.69×10^1	1.94×10^1
	p-value	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	5.00×10^{-1}	1.00×10^{-3}
CEC2017-F22	Mean	2.30×10^3	2.30×10^3	2.30×10^3	2.30×10^3	2.67×10^3	2.35×10^3	2.70×10^3	2.65×10^3	2.31×10^3
	Std	1.37×10^2	1.44×10^1	2.78×10^0	8.66×10^0	6.66×10^2	2.43×10^2	7.49×10^2	2.09×10^2	1.78×10^1
	p-value	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.17×10^{-1}	1.00×10^{-3}
CEC2017-F23	Mean	2.62×10^3	2.62×10^3	2.62×10^3	2.71×10^3	2.68×10^3	2.62×10^3	2.77×10^3	2.68×10^3	2.62×10^3
	Std	7.21×10^0	7.52×10^0	6.76×10^0	3.34×10^1	1.60×10^1	5.47×10^0	6.96×10^1	1.20×10^1	1.25×10^1
	p-value	1.40×10^{-2}	2.88×10^{-2}	1.41×10^{-2}	3.71×10^{-2}	2.12×10^{-2}	3.28×10^{-2}	2.50×10^{-2}	5.00×10^{-1}	5.00×10^{-1}
CEC2017-F24	Mean	2.74×10^3	2.75×10^3	2.74×10^3	2.79×10^3	2.81×10^3	2.75×10^3	2.88×10^3	2.75×10^3	2.75×10^3
	Std	4.60×10^1	9.84×10^0	6.48×10^0	1.00×10^2	1.43×10^1	2.59×10^1	1.41×10^2	7.24×10^1	1.39×10^1
	p-value	1.00×10^{-3}	3.64×10^{-2}	1.09×10^{-1}	1.00×10^{-3}	3.40×10^{-1}	1.00×10^{-3}	1.00×10^{-3}	7.96×10^{-2}	1.00×10^{-3}
CEC2017-F25	Mean	2.92×10^3	2.92×10^3	2.93×10^3	2.93×10^3	3.05×10^3	2.93×10^3	2.94×10^3	3.17×10^3	2.94×10^3
	Std	1.97×10^1	2.36×10^1	2.58×10^1	2.26×10^1	1.07×10^2	1.86×10^1	7.67×10^1	7.60×10^1	2.20×10^1
	p-value	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	3.79×10^{-3}	4.77×10^{-2}	1.00×10^{-3}	5.00×10^{-1}	1.00×10^{-3}
CEC2017-F26	Mean	3.04×10^3	2.97×10^3	3.04×10^3	3.19×10^3	3.57×10^3	3.01×10^3	4.07×10^3	3.76×10^3	3.11×10^3
	Std	2.64×10^2	2.03×10^2	2.96×10^2	5.15×10^2	4.73×10^2	7.43×10^1	6.36×10^2	2.36×10^2	3.87×10^2
	p-value	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.36×10^{-3}	3.92×10^{-3}	5.00×10^{-1}	5.59×10^{-2}	3.08×10^{-1}	1.00×10^{-3}
CEC2017-F27	Mean	3.09×10^3	3.09×10^3	3.10×10^3	3.18×10^3	3.11×10^3	3.09×10^3	3.27×10^3	3.15×10^3	3.10×10^3
	Std	2.27×10^0	9.39×10^0	2.15×10^1	4.13×10^1	1.40×10^1	1.85×10^0	9.38×10^1	1.30×10^1	1.20×10^1
	p-value	4.58×10^{-1}	1.00×10^{-3}	1.00×10^{-3}	5.00×10^{-1}	1.21×10^{-2}	3.15×10^{-3}	1.35×10^{-3}	5.00×10^{-1}	1.00×10^{-3}
CEC2017-F28	Mean	3.18×10^3	3.34×10^3	3.39×10^3	3.32×10^3	3.44×10^3	3.22×10^3	3.37×10^3	3.55×10^3	3.37×10^3
	Std	5.98×10^1	1.78×10^2	1.29×10^2	1.33×10^2	1.18×10^2	4.98×10^1	1.50×10^2	1.04×10^2	1.03×10^2
	p-value	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.74×10^{-1}	1.61×10^{-3}	1.00×10^{-3}	5.00×10^{-1}	1.00×10^{-3}
CEC2017-F29	Mean	3.17×10^3	3.21×10^3	3.20×10^3	3.24×10^3	3.43×10^3	3.18×10^3	3.61×10^3	3.37×10^3	3.20×10^3
	Std	5.95×10^1	2.74×10^1	4.81×10^1	5.76×10^1	1.07×10^2	3.10×10^1	2.20×10^2	5.75×10^1	4.47×10^1
	p-value	7.20×10^{-2}	6.33×10^{-2}	1.52×10^{-2}	1.25×10^{-1}	5.00×10^{-1}	1.00×10^{-3}	3.08×10^{-1}	2.96×10^{-1}	1.24×10^{-3}
CEC2017-F30	Mean	3.28×10^3	5.84×10^5	5.10×10^5	2.75×10^5	8.52×10^5	1.51×10^5	1.51×10^6	2.84×10^6	5.96×10^5
	Std	4.10×10^1	8.16×10^5	9.33×10^5	4.94×10^5	1.31×10^5	2.25×10^5	2.11×10^6	2.61×10^6	9.05×10^5
	p-value	1.92×10^{-2}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}
(+ / − / =) Statistic		(21/0/8)	(12/0/17)	(8/0/21)	(3/0/26)	(0/10/19)	(2/0/27)	(0/12/17)	(0/7/22)	(1/0/28)
Friedman mean		1.66	2.03	3.07	3.86	6.79	3.52	6.45	6.31	4.21
Ranking		1	2	3	5	9	4	8	7	6

The best mean results and standard deviation values are highlighted in bold. The p -value is the result of the Lilliefors normality test.

Table 5. DMSALAs performance compared with 8 advanced algorithms on IEEE CEC 2022.

		DMSALAs	ALAs	WMAs	PSOs	SFOAs	BKAs	GOOSEs	BWOs	GWOs
CEC2022-F1	Mean	3.00×10^2	3.00×10^2	3.00×10^2	3.00×10^2	2.30×10^4	8.38×10^3	3.23×10^2	6.94×10^3	2.18×10^3
	Std	8.06×10^{-10}	3.16×10^{-9}	4.09×10^{-14}	3.40×10^{-3}	1.24×10^4	4.97×10^3	1.25×10^2	1.77×10^3	2.15×10^3
	p-value	2.81×10^{-2}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	3.62×10^{-2}	2.32×10^{-2}	1.00×10^{-3}	3.75×10^{-1}	1.00×10^{-3}
CEC2022-F2	Mean	4.02×10^2	4.07×10^2	4.06×10^2	4.11×10^2	5.88×10^2	4.07×10^2	4.09×10^2	7.87×10^2	4.30×10^2
	Std	2.01×10^0	2.72×10^0	3.32×10^0	2.13×10^1	1.43×10^2	3.39×10^0	1.57×10^1	1.45×10^2	2.68×10^1
	p-value	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.24×10^{-2}	1.00×10^{-3}	1.00×10^{-3}	2.24×10^{-2}	1.00×10^{-3}
CEC2022-F3	Mean	6.00×10^2	6.00×10^2	6.00×10^2	6.06×10^2	6.73×10^2	6.17×10^2	6.38×10^2	6.41×10^2	6.02×10^2
	Std	1.46×10^0	7.78×10^{-4}	1.47×10^0	6.76×10^0	1.76×10^1	4.54×10^0	1.54×10^1	5.39×10^0	1.19×10^0
	p-value	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	6.98×10^{-3}	2.50×10^{-2}	1.50×10^{-3}	5.00×10^{-1}	5.00×10^{-1}	3.63×10^{-3}
CEC2022-F4	Mean	8.20×10^2	8.14×10^2	8.12×10^2	8.19×10^2	8.69×10^2	8.37×10^2	8.49×10^2	8.47×10^2	8.14×10^2
	Std	8.20×10^0	6.81×10^0	5.16×10^0	6.67×10^0	1.29×10^1	1.53×10^1	2.04×10^1	5.45×10^0	6.90×10^0
	p-value	1.78×10^{-2}	4.27×10^{-2}	2.21×10^{-2}	2.70×10^{-2}	1.66×10^{-2}	2.26×10^{-2}	1.16×10^{-2}	1.48×10^{-2}	2.89×10^{-3}
CEC2022-F5	Mean	9.00×10^2	9.05×10^2	9.06×10^2	9.55×10^2	2.16×10^3	1.39×10^3	2.15×10^3	1.42×10^3	9.20×10^2
	Std	5.64×10^0	3.55×10^{-1}	6.33×10^0	1.06×10^2	7.73×10^2	2.99×10^2	6.09×10^2	1.06×10^2	3.66×10^1
	p-value	2.56×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	9.01×10^{-2}	5.12×10^{-2}	3.77×10^{-2}	5.00×10^{-1}	1.00×10^{-3}

Table 5. Cont.

		DMSALAs	ALAs	WMAs	PSOs	SFOAs	BKAs	GOOSEs	BWOs	GWOs
CEC2022-F6	Mean	1.80×10^3	1.84×10^3	1.93×10^3	1.87×10^3	1.29×10^7	3.70×10^3	3.00×10^3	1.72×10^6	3.91×10^3
	Std	1.11×10^1	2.75×10^1	1.65×10^3	1.71×10^3	3.12×10^7	1.95×10^3	2.52×10^3	1.22×10^6	2.38×10^3
	<i>p</i> -value	1.14×10^{-1}	1.00×10^{-3}	3.53×10^{-1}	7.48×10^{-3}	1.00×10^{-3}	3.26×10^{-1}	3.80×10^{-3}	1.87×10^{-2}	1.00×10^{-3}
CEC2022-F7	Mean	2.02×10^3	2.02×10^3	2.02×10^3	2.04×10^3	2.10×10^3	2.02×10^3	2.15×10^3	2.09×10^3	2.03×10^3
	Std	6.34×10^0	4.89×10^0	7.32×10^0	1.92×10^1	3.43×10^1	1.36×10^0	6.72×10^1	1.36×10^1	1.64×10^1
	<i>p</i> -value	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	7.05×10^{-2}	2.50×10^{-2}	1.35×10^{-1}	1.00×10^{-3}	4.31×10^{-2}	7.25×10^{-2}
CEC2022-F8	Mean	2.22×10^3	2.22×10^3	2.22×10^3	2.22×10^3	2.27×10^3	2.23×10^3	2.42×10^3	2.24×10^3	2.22×10^3
	Std	5.49×10^0	4.95×10^0	9.40×10^0	2.20×10^1	6.42×10^1	1.43×10^0	1.66×10^2	4.32×10^0	5.75×10^0
	<i>p</i> -value	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	2.33×10^{-1}	6.11×10^{-3}	3.07×10^{-1}	1.00×10^{-3}
CEC2022-F9	Mean	2.49×10^3	2.53×10^3	2.53×10^3	2.53×10^3	2.68×10^3	2.52×10^3	2.56×10^3	2.70×10^3	2.56×10^3
	Std	7.10×10^{-12}	0.00×10^0	0.00×10^0	3.67×10^1	6.42×10^1	2.91×10^1	3.54×10^1	2.00×10^1	3.72×10^1
	<i>p</i> -value	1.92×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	5.00×10^{-1}	1.00×10^{-3}	4.59×10^{-3}	1.37×10^{-2}	1.00×10^{-3}
CEC2022-F10	Mean	2.50×10^3	2.50×10^3	2.57×10^3	2.56×10^3	2.73×10^3	2.50×10^3	3.05×10^3	2.55×10^3	2.56×10^3
	Std	6.32×10^1	8.71×10^1	5.88×10^1	6.66×10^1	4.96×10^2	2.27×10^{-1}	5.25×10^2	5.52×10^1	6.02×10^1
	<i>p</i> -value	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.00×10^{-3}	1.18×10^{-1}	3.85×10^{-3}	1.00×10^{-3}	1.00×10^{-3}
CEC2022-F11	Mean	2.73×10^3	2.66×10^3	2.78×10^3	2.75×10^3	3.06×10^3	2.67×10^3	2.76×10^3	3.12×10^3	2.82×10^3
	Std	1.61×10^2	1.46×10^2	1.67×10^2	1.63×10^2	3.42×10^2	4.54×10^1	1.66×10^2	1.82×10^2	2.01×10^2
	<i>p</i> -value	1.00×10^{-3}	1.00×10^{-3}	2.40×10^{-3}	1.00×10^{-3}	2.16×10^{-3}	1.74×10^{-1}	1.00×10^{-3}	3.83×10^{-1}	1.00×10^{-3}
CEC2022-F12	Mean	2.86×10^3	2.86×10^3	2.87×10^3	2.89×10^3	2.87×10^3	2.86×10^3	3.10×10^3	2.90×10^3	2.86×10^3
	Std	2.39×10^0	1.86×10^0	1.09×10^0	5.20×10^1	7.29×10^0	1.34×10^0	7.53×10^1	1.73×10^1	1.17×10^1
	<i>p</i> -value	6.79×10^{-3}	1.24×10^{-2}	1.09×10^{-1}	2.98×10^{-3}	4.37×10^{-3}	5.55×10^{-3}	1.82×10^{-1}	3.70×10^{-1}	1.00×10^{-3}
(+ / − / =) Statistic		(10/0/2)	(7/0/5)	(5/0/7)	(2/0/10)	(0/5/7)	(3/0/9)	(0/4/8)	(0/3/9)	(2/0/10)
Friedman mean		1.42	1.58	2.42	3.08	6.25	3.17	5.17	5.58	3.50
Ranking		1	2	3	4	9	5	7	8	6

The best mean results and standard deviation values are highlighted in bold. The *p*-value is the result of the Lilliefors normality test.

4.1. Ablation Experiment

An ablation study was carried out using the IEEE CEC 2017 dataset to assess the performance of various strategies. Six variants of the ALAs algorithm, namely ALAs1–ALAs6, were designed, each integrating distinct strategies. Specifically, ALA1 features a dynamic adaptive mechanism; ALAs2 utilizes the hybrid Nelder–Mead approach; ALAs3 applies a small-scale perturbation technique; ALAs4 combines the dynamic adaptive mechanism with the hybrid Nelder–Mead method; ALAs5 integrates the dynamic adaptive mechanism and the small-scale perturbation strategy; and ALAs6 merges the hybrid Nelder–Mead approach and the small-scale perturbation technique.

As presented in Table 3, the results of the Lilliefors normality test are as follows: a total of 203 tests (29 test functions and seven algorithms), with 38 cases where the normal distribution cannot be rejected and 165 cases of non-normal distribution. These results confirm that our data indeed have significant non-parametric characteristics, thereby providing a sufficient statistical basis for the application of the Friedman test. The DMSALAs algorithm outperformed others by identifying 22 optimal solutions, with a (+ / − / =) statistical outcome of (22/1/6) and a Friedman mean of 1.48, securing the top rank. Following closely, ALAs6 detected 17 optimal values, yielding a (+ / − / =) statistic of (17/2/10) and a Friedman mean of 1.62, ranking second. ALAs5 identified five optimal values, with a (+ / − / =) statistical result of (5/5/19) and a Friedman mean of 2.97, placing sixth. ALAs4 found six optimal solutions, recording a (+ / − / =) statistic of (6/9/14) and a Friedman mean of 2.90, ranking fifth. ALAs3 recognized 11 optimal values, presenting a (+ / − / =) statistical outcome of (11/7/11) and a Friedman mean of 2.55, ranking fourth. ALAs2 discovered 10 optimal solutions, with a (+ / − / =) statistic of (10/4/15) and a Friedman mean of 2.34, ranking third. In contrast, ALAs1 only identified 10 optimal values, with a (+ / − / =) statistical result of (3/11/15) and a Friedman mean of 3.52, ranking seventh. These findings not only validate the effectiveness of the three strategies mentioned above

but also emphasize the remarkable superiority of the DMSALAs algorithm, as depicted in Figure 4.

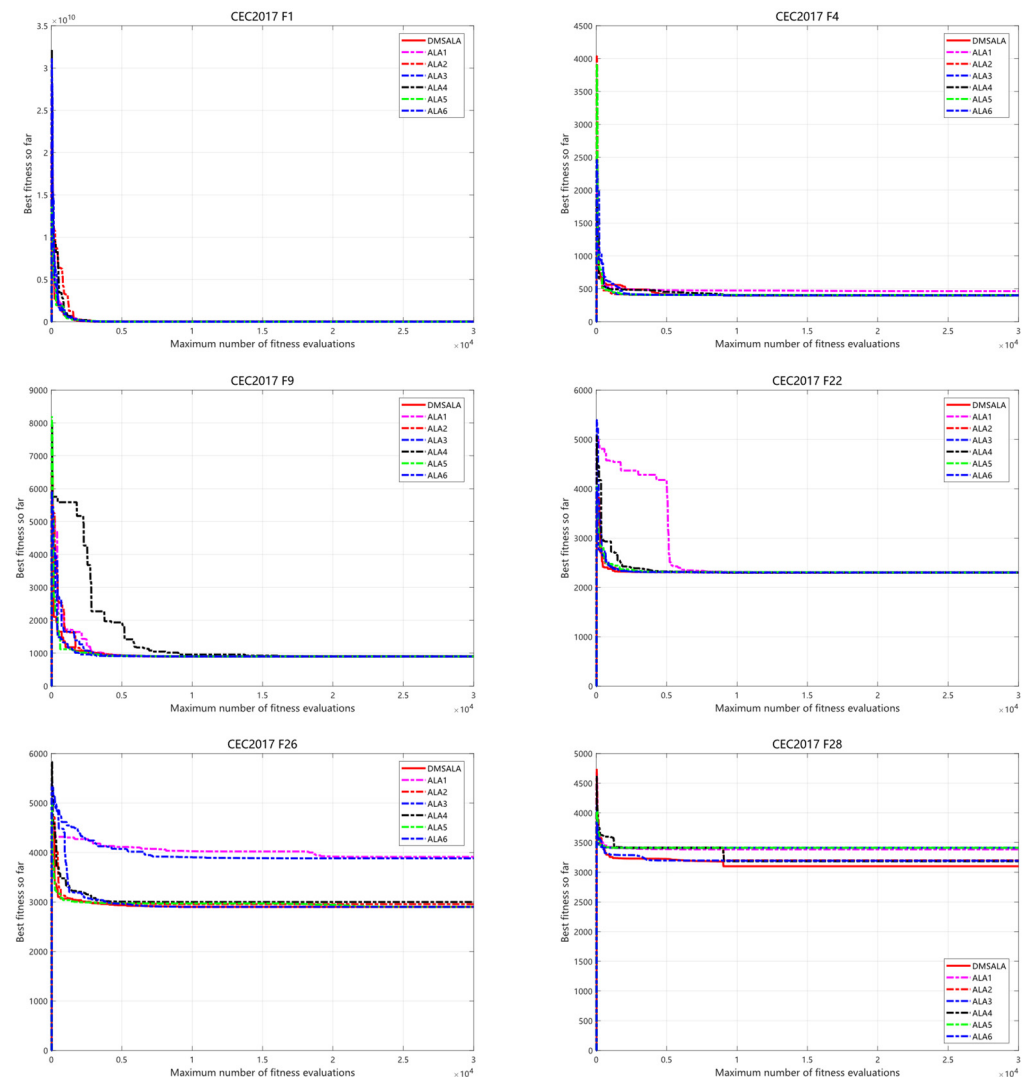


Figure 4. Convergence curve of DMSALAs and its variants on IEEE CEC 2017.

4.2. Comparison Experiment on IEEE CEC2017

To thoroughly evaluate the optimization performance of the proposed DMSALAs algorithm, we performed comprehensive experiments on the IEEE CECs 2017 benchmark. DMSALAs were compared with eight MHs algorithms, including (ALAs) [22], WMAs [30], PSOs [31], SFOAs [32], BKAs [33], GOOSEs [34], BWOs [35], and GWOs [10], selected for their diverse search mechanisms and proven effectiveness in optimization tasks.

As illustrated in Table 4, the results of the Lilliefors normality test are as follows: a total of 261 tests (29 test functions and nine algorithms), with 48 cases where the normal distribution cannot be rejected and 213 cases of non-normal distribution. These results confirm that our data indeed have significant non-parametric characteristics, thereby providing a sufficient statistical basis for the application of the Friedman test. The DMSALAs successfully pinpointed 21 out of 29 optimal solutions, with a (+/−/=) statistical result of (21/0/8). Its Friedman mean score stands at 1.66, securing the top rank and significantly outperforming all comparative algorithms. ALAs achieved 12 optimal solutions, which confirms that the improvements, including dynamic adaptation mechanisms, hybrid Nelder–Mead integration, and perturbation strategies, have notably enhanced its

performance. The (+/−/=) statistic is (12/0/17), accompanied by a Friedman mean of 2.03, ranking second. WMA obtained eight optimal solutions, demonstrating competitive yet less consistent results. Its (+/−/=) statistic is (8/0/21), with a Friedman mean of 3.07, placing it third. BKA identified two optimal values, yielding a (+/−/=) statistic of (2/0/27) and a Friedman mean of 3.52 (fourth place). PSOs secured three optimal solutions, with corresponding metrics of (3/0/26) and 3.86 (fifth place). GWOs only achieved one optimal value, reflecting its limitations in handling complex or high-dimensional problem spaces, with a (+/−/=) statistic of (1/0/28) and a Friedman mean of 4.21 (sixth place). Notably, BWOs, SFOAs, and GOOSEs failed to converge to any global optima, indicating inherent weaknesses in balancing exploration and exploitation. These findings, as visualized in Figure 5, unequivocally validate the dominant performance of the DMSALAs algorithm across evaluated benchmarks.

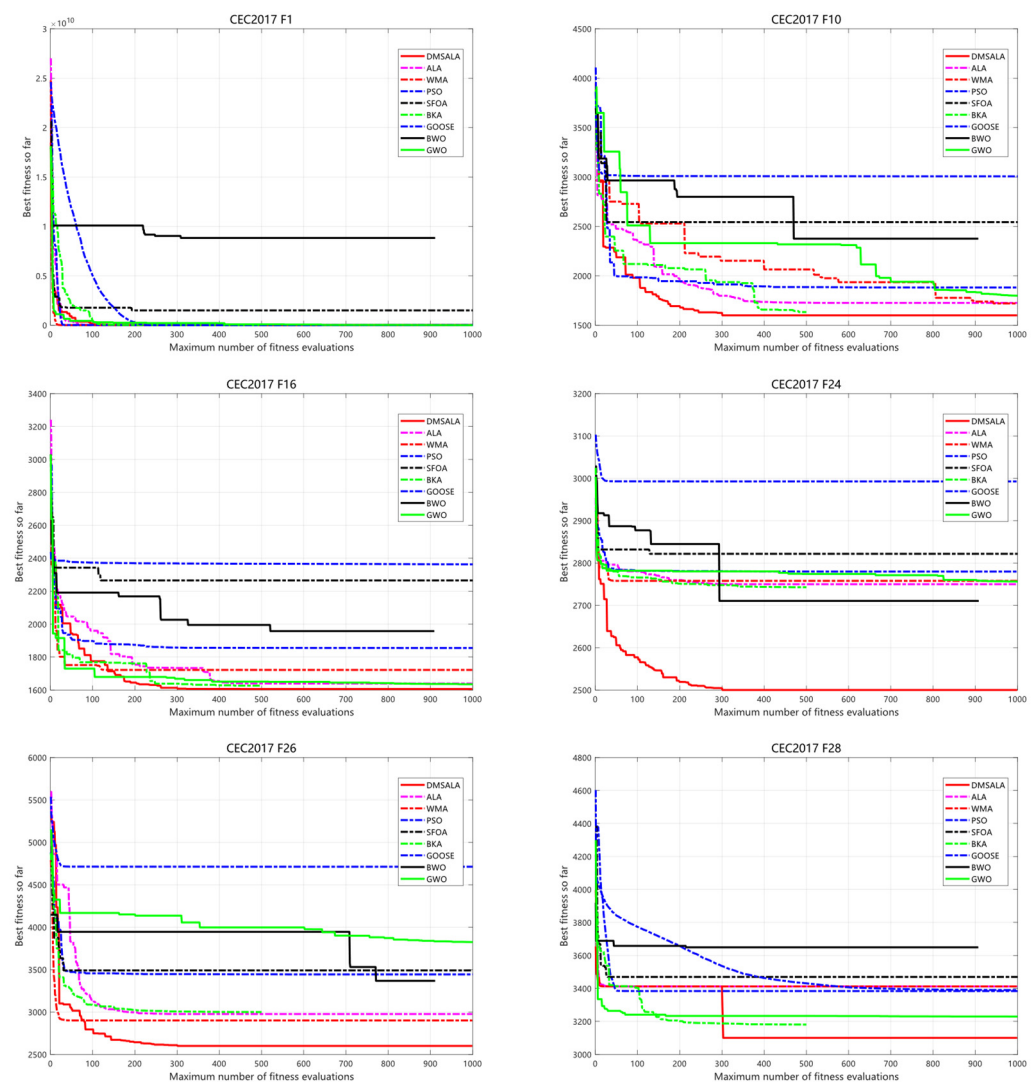


Figure 5. Convergence curve of DMSALAs and 8 advanced algorithms on IEEE CEC 2017.

4.3. Comparison Experiment on IEEE CEC2022

To evaluate DMSALAs performance on the test set, we contrasted it with the eight previously mentioned algorithms using the IEEE CEC 2022 benchmarking framework. The experimental findings are documented in Table 5, while the convergence trajectories of each algorithm across various test functions are illustrated in Figure 6.

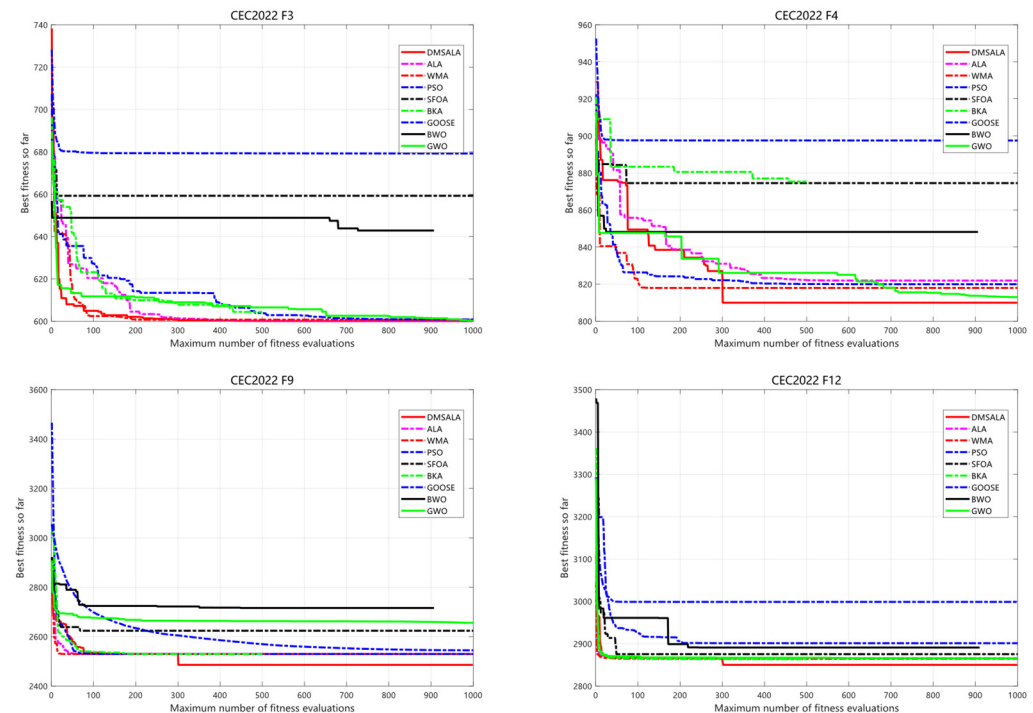


Figure 6. Convergence curve of DMSALAs and 8 advanced algorithms on IEEE CEC 2022.

As presented in Table 5, the results of the Lilliefors normality test are as follows: a total of 108 tests (12 test functions and nine algorithms), with 21 cases where the normal distribution cannot be rejected and 87 cases of non-normal distribution. These results confirm that our data indeed have significant non-parametric characteristics, thereby providing a sufficient statistical basis for the application of the Friedman test. The DMSALAs successfully obtained 10 out of 12 optimal solutions, with a (+/−/=) statistical result of (10/0/2) and a Friedman mean of 1.42, ranking first and significantly outperforming all competing algorithms. ALAs secured seven optimal solutions, verifying that improvements such as dynamic adaptation, hybrid Nelder–Mead integration, and perturbation strategies effectively enhance its performance. With a (+/−/=) statistic of (7/0/5) and a Friedman mean of 1.58, it ranks second. WMAs achieved five optimal solutions, showing competitive yet less stable performance, with a (+/−/=) statistic of (5/0/7) and a Friedman mean of 2.42, ranking third. Both PSOs and GWOs identified only 2 optimal values, sharing a (+/−/=) statistical result of (2/0/10) with Friedman means of 3.08 and 3.50, ranking fourth and sixth, respectively, which reflects their limitations in handling complex high-dimensional problems. BKAs obtained three optimal values, with a (+/−/=) statistic of (3/0/9) and a Friedman mean of 3.17, ranking fifth. Notably, BWOs, SFOAs, and GOOSEs failed to converge to any global optima, indicating flaws in their exploration–exploitation balance mechanisms.

4.4. Wilcoxon Rank Sum Test

The Wilcoxon rank sum test is employed to assess the disparity between DMSALAs and competitor methods. A significance level of 0.05 is set, and (+|−|) denotes superiority, equality, or inferiority of the DMSALAs algorithm compared to its competitors. Table 6 illustrates the notable variances found in the majority of functions between our suggested DMSALAs and alternative algorithms. The results are as follows: 182/0/50 and 84/0/12. These results collectively indicate that our proposed DMSALAs exhibit substantial distinctions when compared with the other ten algorithms.

Table 6. Wilcoxon rank sum test statistical results.

DMSALAs VS.	CECs2017 F1–F30 Dim = 30	CECs2022 F1–F12 Dim = 10	DMSALAs VS.	CECs2017 F1–F30 Dim = 30	CECs2022 F1–F12 Dim = 10
ALAs	19/0/10	10/0/2	BKA	20/0/9	10/0/2
WMAs	17/0/12	9/0/3	GOOSE	28/0/1	12/0/0
PSOs	25/0/4	10/0/2	BWO	27/0/2	11/0/1
SFOAs	28/0/1	12/0/0	GWO	18/0/11	10/0/2
Total (+ = −)	89/0/27	41/0/7		93/0/23	43/0/5

5. Path Planning Problem for Mobile Robots

In this section, we conducted a comprehensive performance evaluation of the DMSALAs algorithm, validating its efficacy in solving the mobile robot path planning problem through comparative analyses with eight benchmark algorithms. The validation was performed across two distinct environments: a simulated testing scenario and the physical operational setting of the RIA-E100 robotic system.

5.1. Simulation Experiment for Mobile Robot Path Planning

To comprehensively evaluate the performance and scalability of the proposed algorithm, experiments were conducted on four grid maps of different sizes: 10×10 , 20×20 , 30×30 , and 40×40 . This size sequence establishes a uniformly increasing gradient of search space complexity, aiming to systematically assess the algorithm's adaptability from simple to complex scenarios. The smaller-scale maps are used to verify the basic correctness of the algorithm, while the medium- and large-scale maps are employed to evaluate its convergence behavior, solution quality, and computational efficiency as the problem size increases—thereby assessing the algorithm's scalability. The evaluated algorithmic parameters consisted of the optimal, average, and standard deviation values of the path length, calculated over 30 independent trials.

The path length in the mobile robot path planning problem is defined as:

$$f(x) = \begin{cases} f(x) + \sum_{i=1}^{\text{size}(\text{path},1)} \sqrt{(\text{path}(i+1,1) - \text{path}(i,1))^2 + (\text{path}(i+1,2) - \text{path}(i,2))^2}, N_b = 0 \\ E(1) \cdot E(2) \cdot N_b, N_b > 0 \end{cases} \quad (13)$$

where *path* represents the generated route, *E*(1) and *E*(2) represent the number of rows and columns in the map, respectively, and *N_b* indicates the number of obstacles encountered along the path. First, a smooth path is generated through a midpoint neighborhood search followed by two smoothing processes. Subsequently, the total path length, denoted as *f*(*x*), is computed by summing the Euclidean distances between each pair of adjacent points along the trajectory.

The DMSALAs algorithms performance evaluation was executed using MATLAB 2023b software. Simulations were run on a desktop system featuring Windows 11 Pro, an Intel(R) Core (TM) i9-14900KF processor (3.20 GHz), and 128 GB of RAM (Hangzhou China).

As depicted in Table 7 and Figure 7, the DMSALAs algorithm efficiently located optimal values across four maps with distinct scales and attained the optimal average performance. The Friedman test ranking is the first. Thereby highlighting its superior optimization capability. While DMSALAs demonstrated stable performance, they did not yield the lowest standard deviation in all cases. Notably, the BKA algorithm achieved the best standard deviation in two scenarios, whereas the BWO and WMA algorithms

each excelled in one case. This suggests that the DMSALAs algorithm has room for stability improvement.

Table 7. DMSALAs performance compared with 8 advanced algorithms on 4 different maps.

Maps		DMSALAs		ALAs	WMAs	PSOs	SFOAs	BKAs	GOOSEs	BWOs	GWOs
10 × 10	Best	12.95742	12.95742	12.95742	12.95742	12.95742	12.95742	12.95742	13.5432	12.95742	12.95742
	mean	13.14049	13.25876	14.87895	13.22565	15.97011	13.68594	26.67653	13.26048	14.48174	
	std	0.645548	0.54497	1.685236	0.537623	3.718501	0.54964	25.11747	0.330406	1.479801	
20 × 20	Best	27.54258	28.93703	30.2661	28.25798	33.43024	30.39802	38.45355	40.90339	34.57773	
	mean	28.74429	29.20669	29.85713	29.89343	45.32093	30.7127	111.5527	48.98892	32.53923	
	std	0.902847	1.230958	1.909247	2.024177	67.05115	0.732436	146.7869	66.32595	4.096773	
30 × 30	Best	47.83477	47.96416	48.87945	48.69489	50.94203	49.83517	61.03903	54.45101	51.95469	
	mean	53.22709	55.40341	81.68151	63.42001	426.8657	60.90314	303.6167	319.7956	69.24881	
	std	6.135755	6.466712	154.7195	7.363908	420.8932	5.838168	435.3804	386.5595	12.41287	
40 × 40	Best	64.9859	66.40731	69.147	70.07844	94.10414	65.58852	91.36641	75.39668	79.70253	
	mean	90.5632	94.50665	98.02464	362.4215	1353.986	99.31272	607.404	2666.667	608.8957	
	std	17.65466	19.12524	9.074725	606.3704	602.6078	20.52429	768.9426	1306.395	767.74	
(+ / − / =) Statistic		(4 0 0)	(1 3 0)	(1 3 0)	(1 3 0)	(1 1 2)	(1 3 0)	(1 2 1)	(1 1 2)	(1 3 0)	
Friedman mean		1	2.25	4.75	3.75	8	4.25	7.75	7.25	6	
Ranking		1	2	5	3	9	4	8	7	6	

The best values are highlighted in bold.

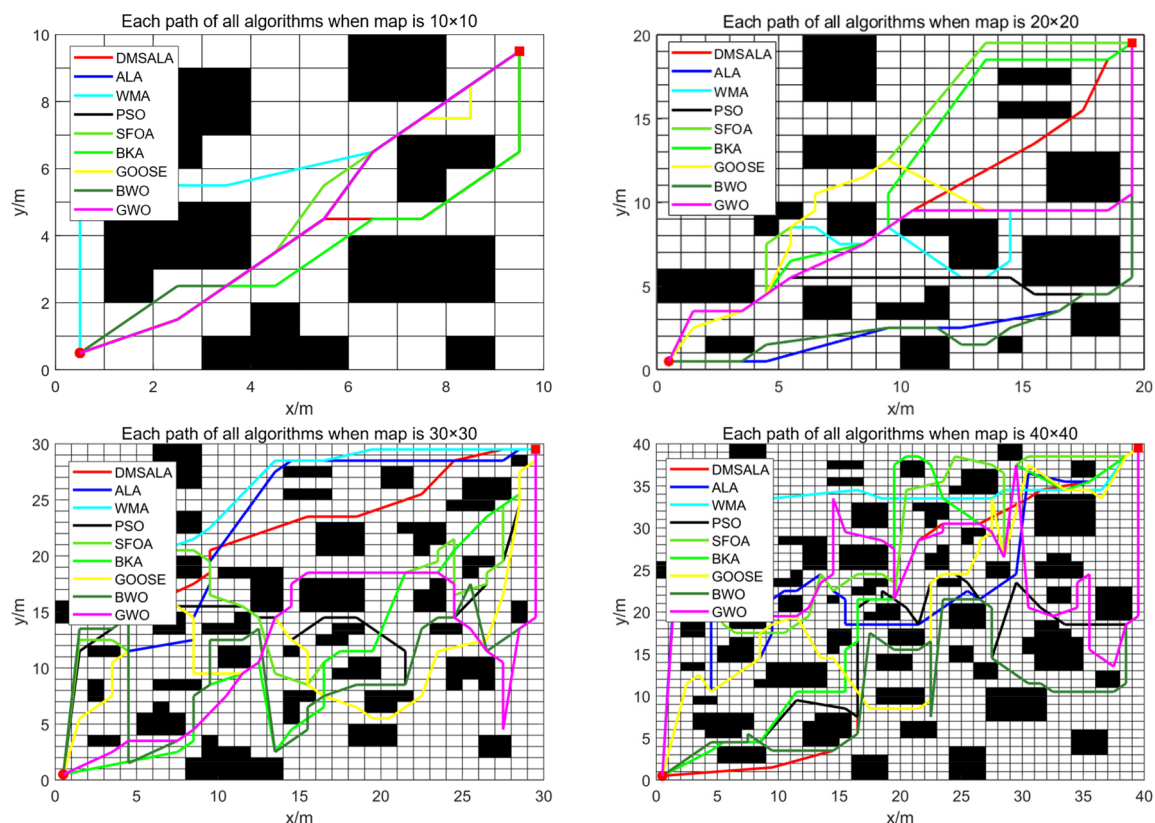


Figure 7. Path found of DMSALAs and 8 advanced algorithms.

DMSALAs consistently achieved the minimum path lengths (optimal values) across all four map scenarios, demonstrating their superior equilibrium between exploration and exploitation capabilities. The algorithm also exhibited the lowest mean path length across multiple runs, underscoring its robust performance. Although DMSALAs standard

deviations remained stable, they were not optimal in all cases (BKAs outperformed in 2 of 4 maps). This minor variability in performance is likely attributed to random perturbations in complex mapping environments. By contrast, BKAs and BWOs occasionally demonstrated better standard deviations, primarily due to their exploitation-oriented strategies. DMSALAs adaptive mechanism prioritizes optimality over strict repeatability, which explains its slightly higher variance trade-off that emphasizes solution quality over absolute consistency.

5.2. Real-World Experimental Verification of DMSALAs for Mobile Robot Path Planning

In this section, the DMSALAs algorithm and the eight previously introduced algorithms were experimentally evaluated on the RIA-E100 mobile robot, a compact modular model from GaiTech Robotics' RIAs series. As illustrated in Figure 8, the robot features a differential-drive configuration with two powered wheels at the front and two passive omnidirectional wheels at the rear, ensuring agile maneuverability. Its computational capabilities are supported by an Intel i5 processor, 8GB of RAM, and a 120GB SSD, enabling real-time data processing. The experimental setup integrates an Astra Orbbec RGBD camera, an Inertial Measurement Unit (IMU), and a 2D LiDAR sensor, facilitating environmental perception. Wireless communication between the control computer and the robot is established via Wi-Fi, ensuring seamless data transfer. The testing environment has a length of 3.2 m and a width of 2.7 m, with obstacles randomly distributed within the area.

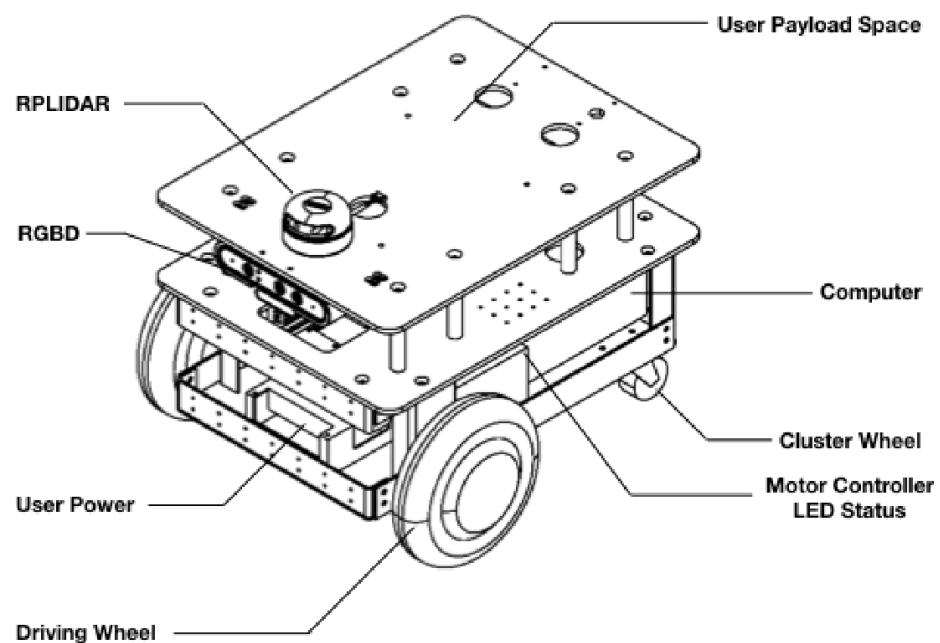


Figure 8. Structural Diagram of the RIA-E100 Mobile Robot.

In Figures 9 and 10, the starting point of the map is located at the lower left corner in the actual map scenario, while the endpoint is positioned at the upper right corner. In contrast, within the Rviz environment, the map starts at the lower right corner and ends at the upper left corner. Cuboid-shaped obstacles are strategically placed to simulate challenging navigation scenarios, with the unobstructed zones designated as navigable regions. To ensure comparability, identical start and end coordinates were assigned across all algorithms. Each algorithm underwent 30 independent runs, during which the minimum path length was recorded. Subsequently, the arithmetic means and variance of these 30 trials were computed to analyze the algorithms' performance stability statistically. Figure 9 captures the robot's navigation process in the physical environment, and Figure 10 illustrates the

corresponding occupancy grid map generated in Rviz, providing a comprehensive view of the experimental outcomes.



Figure 9. The operation of the RIA-E100 mobile robot in the real environment.

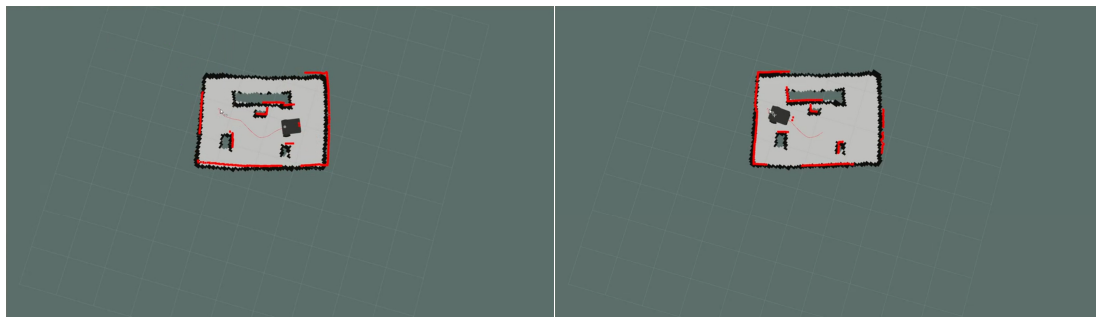


Figure 10. Map of the RIA-E100 mobile robot in Rviz.

Table 8 presents the optimal values, mean values, and variances of path lengths derived from 30 independent runs of the DMSALAs algorithm and eight comparative algorithms. DMSALAs attained an optimal path length of 3.687 m and a mean path length of 3.881 m, both of which ranked top-ranked across all algorithms. Although the standard deviation of DMSALAs over 30 runs was not the minimum, standing in third place, trailing only the WMAs and ALAs algorithms, it still maintained a relatively low magnitude. These findings firmly validate the effectiveness of the DMSALAs algorithm in addressing mobile robot path planning challenges.

Table 8. Results of the DMSALAs and 8 advanced algorithms on the RIA-E100 robot.

	DMSALAs	ALAs	WMAs	PSOs	SFOAs	BKAs	GOOSEs	BWOs	GWOs
Best (m)	3.687	3.941	3.914	4.090	4.651	4.983	4.797	4.486	4.338
Mean (m)	3.881	4.069	4.077	4.368	5.049	5.217	5.268	4.805	4.539
std	0.2587	0.1952	0.1156	0.4897	0.5139	0.5047	0.4928	0.5915	0.3622

The best values are highlighted in bold.

6. Conclusions

By integrating a dynamic adaptive mechanism, a hybrid Nelder–Mead method, and a small-range perturbation strategy into the ALAs algorithm, we developed the DMSALAs algorithm. To assess its efficacy, DMSALAs were first compared with six variant algorithms to validate the proposed strategies, then benchmarked against eight advanced algorithms to demonstrate its superior performance. These enhancements effectively address the original algorithm's core limitations—fixed parameters, weak local search capability, and

insufficient diversity preservation-making it better suited for complex optimization tasks. Specifically, the hybrid Nelder–Mead component plays a pivotal role in refining way-points, minimizing detours, and ensuring smooth trajectories; the dynamic search radius adjustment prevents premature stagnation, and perturbation strategies enable escape from concave obstacle regions.

To further verify DMSALAs applicability in mobile robot path planning, we systematically executed simulation experiments for the DMSALAs algorithm and eight comparative algorithms across four maps with varying scales. Subsequently, we carried out real-world experiments utilizing the RIA-E100 mobile robot within a custom-constructed physical environment. Our dual-approach experimental design, which combines virtual simulations across four maps of varying scales and real-world validations using the RIA-E100 robot, enabled a comprehensive evaluation of the DMSALAs algorithm and eight comparative algorithms. This methodology demonstrated DMSALAs superiority in addressing practical challenges, proving it to be a highly effective optimizer, especially for complex, high-dimensional problems. Its consistent top performance across diverse scenarios highlights its broad applicability in real-world engineering optimizations, such as robotic path planning tasks that demand both solution precision and reliability.

Notably, while DMSALAs demonstrate strong effectiveness, its standard deviation results across various tests indicate room for stability improvements, even after the implemented enhancements. This presents a key direction for future research. Additionally, as this study tested DMSALAs up to 30D problems, exploring its performance in higher-dimensional scenarios (50D and 100D) constitutes another critical research avenue. While this work focuses on mobile robot path planning, future efforts will aim to extend DMSALAs to other practical domains, including industrial robotics and search-and-rescue operations. In this study, high-performance algorithms were not included in the comparative analysis; however, performance comparisons with a broader selection of state-of-the-art algorithms will be conducted in future work and will constitute an important research direction. Furthermore, we plan to carry out rigorous ablation studies to precisely quantify the synergistic effects of the hybrid strategy. Specifically, on the CECs benchmark suite, incremental variants of the algorithm will be compared to disentangle the contribution of the enhanced local search component from the effect of increased function evaluations, thereby providing empirical support for the proposed synergy.

Although current stability and scalability limitations require further investigation, DMSALAs consistent excellence in both benchmark functions and applied path planning solidifies its status as a state-of-the-art optimization tool. Future research will prioritize bridging the gap between theoretical optimization and real-world deployment, ensuring robust performance in dynamic and uncertain environments.

Author Contributions: P.Q.: Conceptualization, Methodology, Writing, Data testing, Reviewing, Software; X.S.: Conceptualization, Funding; Z.Z.: Supervision, Reviewing. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (No. 52065010 and No. 52165063). The Key Laboratory of New Power System Operation Control of Guizhou Province (Qiankehe Platform ZSYS[2025]007).

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

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

References

1. Qu, P.J.; Yuan, Q.N.; Du, F.L.; Gao, Q.Y. An improved manta ray foraging optimization algorithm. *Sci. Rep.* **2024**, *14*, 10301. [\[CrossRef\]](#)
2. Xiao, Y.N.; Sun, X.; Guo, Y.L.; Li, S.P.; Zhang, Y.P.; Wang, Y.W. An Improved Gorilla Troops Optimizer Based on Lens Opposition-Based Learning and Adaptive β -Hill Climbing for Global Optimization. *Cmes-Comput. Model. Eng. Sci.* **2022**, *131*, 815–850. [\[CrossRef\]](#)
3. Miarnaeimi, F.; Azizyan, G.; Rashki, M. Horse herd optimization algorithm: A nature-inspired algorithm for high-dimensional optimization problems. *Knowl.-Based Syst.* **2021**, *213*, 106711. [\[CrossRef\]](#)
4. El-kenawy, E.M.; Khodadadi, N.; Mirjalili, S.; Abdelhamid, A.A.; Eid, M.M.; Ibrahim, A. Greylag Goose Optimization: Nature-inspired optimization algorithm. *Expert Syst. Appl.* **2024**, *238*, 122147. [\[CrossRef\]](#)
5. Zhu, F.F.; Shuai, Z.H.; Lu, Y.R.; Su, H.H.; Yu, R.W.; Li, X.; Zhao, Q.; Shuai, J.W. oBABC: A one-dimensional binary artificial bee colony algorithm for binary optimization. *Swarm Evol. Comput.* **2024**, *87*, 10156. [\[CrossRef\]](#)
6. Ebeed, M.; Abdelmoteleb, M.A.; Khan, N.H.; Jamal, R.; Kamel, S.; Hussien, A.G.; Zawbaa, H.M.; Jurado, F.; Sayed, K. A Modified Artificial Hummingbird Algorithm for solving optimal power flow problem in power systems. *Energy Rep.* **2024**, *11*, 982–1005. [\[CrossRef\]](#)
7. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [\[CrossRef\]](#)
8. Belge, E.; Altan, A.; Hacioglu, R. Metaheuristic Optimization-Based Path Planning and Tracking of Quadcopter for Payload Hold-Release Mission. *Electronics* **2022**, *11*, 1208. [\[CrossRef\]](#)
9. Li, W.Y.; Shi, R.H.; Dong, J. Harris hawks optimizer based on the novice protection tournament for numerical and engineering optimization problems. *Appl. Intell.* **2023**, *53*, 6133–6158. [\[CrossRef\]](#)
10. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [\[CrossRef\]](#)
11. Shao, S.K.; Peng, Y.; He, C.L.; Du, Y. Efficient path planning for UAV formation via comprehensively improved particle swarm optimization. *Isa Trans.* **2020**, *97*, 415–430. [\[CrossRef\]](#)
12. Miao, C.W.; Chen, G.Z.; Yan, C.L.; Wu, Y.Y. Path planning optimization of indoor mobile robot based on adaptive ant colony algorithm. *Comput. Ind. Eng.* **2021**, *156*, 107230. [\[CrossRef\]](#)
13. Mafarja, M.M.; Mirjalili, S. Hybrid Whale Optimization Algorithm with simulated annealing for feature selection. *Neurocomputing* **2017**, *260*, 302–312. [\[CrossRef\]](#)
14. Yu, H.L.; Gao, Y.L.; Wang, L.; Meng, J.T. A Hybrid Particle Swarm Optimization Algorithm Enhanced with Nonlinear Inertial Weight and Gaussian Mutation for Job Shop Scheduling Problems. *Mathematics* **2020**, *8*, 1355. [\[CrossRef\]](#)
15. Li, C.L.; Sun, G.J.; Deng, L.B.; Qiao, L.Y.; Yang, G.Q. A population state evaluation-based improvement framework for differential evolution. *Inf. Sci.* **2023**, *629*, 15–38. [\[CrossRef\]](#)
16. Xiao, Y.N.; Cui, H.; Hussien, A.G.; Hashim, F.A. MSAO: A multi-strategy boosted snow ablation optimizer for global optimization and real-world engineering applications. *Adv. Eng. Inform.* **2024**, *61*, 102464. [\[CrossRef\]](#)
17. Gandomi, A.H. Interior search algorithm (ISA): A novel approach for global optimization. *Isa Trans.* **2014**, *53*, 1168–1183. [\[CrossRef\]](#)
18. Shayanfar, H.; Gharehchopogh, F.S. Farmland fertility: A new metaheuristic algorithm for solving continuous optimization problems. *Appl. Soft Comput.* **2018**, *71*, 728–746. [\[CrossRef\]](#)
19. Padash, A.; Sandev, T.; Kantz, H.; Metzler, R.; Chechkin, A. Asymmetric Levy Flights Are More Efficient in Random Search. *Fractal Fract.* **2022**, *6*, 260. [\[CrossRef\]](#)
20. Pavlik, J.A.; Sewell, E.C.; Jacobson, S.H. Two new bidirectional search algorithms. *Comput. Optim. Appl.* **2021**, *80*, 377–409. [\[CrossRef\]](#)
21. Xu, Y.H.; Zhou, W.N.; Fang, J.A.; Sun, W. Adaptive lag synchronization and parameters adaptive lag identification of chaotic systems. *Phys. Lett. A* **2010**, *374*, 3441–3446. [\[CrossRef\]](#)
22. Xiao, Y.N.; Cui, H.; Abu Khurma, R.; Castillo, P.A. Artificial lemming algorithm: A novel bionic meta-heuristic technique for solving real-world engineering optimization problems. *Artif. Intell. Rev.* **2025**, *58*, 84. [\[CrossRef\]](#)
23. Wei, Q.L.; Liu, D.R.; Lin, H.Q. Value Iteration Adaptive Dynamic Programming for Optimal Control of Discrete-Time Nonlinear Systems. *IEEE Trans. Cybern.* **2016**, *46*, 840–853. [\[CrossRef\]](#) [\[PubMed\]](#)
24. Zhao, Y.W.; Niu, B.; Zong, G.D.; Xu, N.; Ahmad, A.M. Event-triggered optimal decentralized control for stochastic interconnected nonlinear systems via adaptive dynamic programming. *Neurocomputing* **2023**, *539*, 126163. [\[CrossRef\]](#)
25. Liang, H.J.; Liu, G.L.; Zhang, H.G.; Huang, T.W. Neural-Network-Based Event-Triggered Adaptive Control of Nonaffine Nonlinear Multiagent Systems with Dynamic Uncertainties. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 2239–2250. [\[CrossRef\]](#)
26. Mai, C.L.; Zhang, L.X.; Chao, X.W.; Hu, X.; Wei, X.Z.; Li, J. A novel MPPT technology based on dung beetle optimization algorithm for PV systems under complex partial shade conditions. *Sci. Rep.* **2024**, *14*, 6471. [\[CrossRef\]](#)
27. Yin, Z.Y.; Jin, Y.F.; Shen, J.S.; Hicher, P.Y. Optimization techniques for identifying soil parameters in geotechnical engineering: Comparative study and enhancement. *Int. J. Numer. Anal. Methods Geomech.* **2018**, *42*, 70–94. [\[CrossRef\]](#)

28. Ali, A.F.; Tawhid, M.A. A hybrid cuckoo search algorithm with Nelder Mead method for solving global optimization problems. *Springerplus* **2016**, *5*, 473. [[CrossRef](#)]
29. Huang, J.; Biggs, J.D.; Bai, Y.L.; Cui, N.G. Integrated guidance and control for solar sail station-keeping with optical degradation. *Adv. Space Res.* **2021**, *67*, 2823–2833. [[CrossRef](#)]
30. He, J.H.; Zhao, S.J.; Ding, J.Y.; Wang, Y.M. Mirage search optimization: Application to path planning and engineering design problems. *Adv. Eng. Softw.* **2025**, *203*, 103883. [[CrossRef](#)]
31. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948. [[CrossRef](#)]
32. Jia, H.M.; Zhou, X.L.; Zhang, J.R.; Mirjalili, S. Superb Fairy-wren Optimization Algorithm: A novel metaheuristic algorithm for solving feature selection problems. *Clust. Comput.-J. Netw. Softw. Tools Appl.* **2025**, *28*, 246. [[CrossRef](#)]
33. Wang, J.; Wang, W.C.; Hu, X.X.; Qiu, L.; Zang, H.F. Black-winged kite algorithm: A nature-inspired meta-heuristic for solving benchmark functions and engineering problems. *Artif. Intell. Rev.* **2024**, *57*, 98. [[CrossRef](#)]
34. Hamad, R.K.; Rashid, T.A. GOOSE algorithm: A powerful optimization tool for real-world engineering challenges and beyond. *Evol. Syst.* **2024**, *15*, 1249–1274. [[CrossRef](#)]
35. Zhong, C.T.; Li, G.; Meng, Z. Beluga whale optimization: A novel nature-inspired metaheuristic algorithm. *Knowl.-Based Syst.* **2022**, *251*, 109215. [[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.