

A State-Sensing Adaptive Artificial Bee Colony Algorithm with Dynamic Search and Rank-Based Selection for High-Dimensional Complex Optimization

Xinyao Gao

gaoxinyao@mails.neu.edu.cn

Northeastern University

Research Article

Keywords: Artificial Bee Colony Algorithm, State-Sensing Adaptive Mechanism, Dynamic Hybrid Initialization, Elite-Mean-Guided Dimension Selection, Dynamic Dimension Search, Elite Immunity Protection, Gravitational Gradient Selection, Robot Path Planning

Posted Date: September 2nd, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-10274135/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.
[Read Full License](#)

Additional Declarations: No competing interests reported.

A State-Sensing Adaptive Artificial Bee Colony Algorithm with Dynamic Search and Rank-Based Selection for High-Dimensional Complex Optimization

Xinyao Gao¹

¹ School of Resources and Civil Engineering, Northeastern University, Shenyang, Liaoning, 110004, China

Corresponding author: Xinyao Gao (e-mail: gaoxinyao@mails.neu.edu.cn)

Abstract Standard Artificial Bee Colony (ABC) suffers from poor initialization, imbalanced exploration and exploitation in its one-dimensional search mechanism, and weakening selection pressure, limiting convergence speed, accuracy, and stability. This paper proposes State-Sensing Adaptive ABC (SSA-ABC). It senses the problem's dimensionality and uses sigmoid dynamic hybrid initialization combining Latin hypercube sampling, random initialization, and a reflection-hybrid boundary mechanism. It senses progress via function evaluation ratio to dynamically adjust search from one-dimensional exploitation to multi-dimensional exploration, with elite-mean-guided dimension selection. It senses population ranking, applying immune protection to top 10% elites and a two-stage selection model: roulette wheel for early-stage diversity, then a shrinking elite nucleus with rank-based gravitational gradients. Experiments on the CEC2014 benchmark set show the positive contribution of each mechanism in the ablation study. Against six state-of-the-art ABC variants, SSA-ABC achieves the lowest Friedman mean rank, as well as superior accuracy, stability, and convergence speed. In robot path planning, SSA-ABC achieves the best path length and smoothness, outperforming standard ABC, six improved ABCs, and five other swarm intelligence algorithms. SSA-ABC overcomes ABC's three main limitations, offering an efficient solution for high-dimensional complex problems.

Keywords Artificial Bee Colony Algorithm, State-Sensing Adaptive Mechanism, Dynamic Hybrid Initialization, Elite-Mean-Guided Dimension Selection, Dynamic Dimension Search, Elite Immunity Protection, Gravitational Gradient Selection, Robot Path Planning

Mathematics Subject Classification (2020) 90C59 90C26

1 Introduction

Among swarm intelligence algorithms, the Artificial Bee Colony (ABC) algorithm has received widespread attention due to its simple structure, few control parameters, and strong robustness, inspired by the foraging behavior of bee colonies [1,2,3]. Compared with mainstream evolutionary computation algorithms such as Particle Swarm Optimization (PSO) [4,5] and Differential Evolution (DE) [6,7], ABC exhibits competitive performance on various benchmark problems and has been applied to numerous practical problems, including feature selection [8] and flow shop scheduling

[9], etc. Taking robot path planning as an example, this problem requires finding the optimal safe path from the start point to the destination point in an environment with obstacles, featuring high dimensionality, multiple constraints, and multiple objectives. Traditional algorithms face limitations such as high computational complexity and insufficient search capability in high-dimensional complex scenarios [10], while ABC demonstrates application potential in this domain due to its strong global search capability [11,12]. However, ABC suffers from search stagnation when handling complex problems [13,14], primarily because the algorithm has strong exploration capability but insufficient exploitation capability [15,16]. This deficiency is particularly pronounced in high-dimensional and multimodal optimization problems: the one-dimensional search strategy of standard ABC struggles to effectively locate optimal solutions in high-dimensional solution spaces, resulting in slow convergence speed and low solution accuracy.

To enhance the exploitation capability of ABC, researchers have proposed various improved variants, most of which employ global optimal individuals or elite individuals to guide the search. Zhu and Kwong proposed a global best-guided GABC, incorporating the global optimal individual into the solution search equation [17]. Zhou et al. proposed a multi-elite-guided MGABC, selecting outstanding individuals to form an elite population and modifying the solution search equation accordingly [18]. The feasibility of the above methods has been verified in multiple studies. Cui et al. integrated a depth-first search framework with an elite-guided search equation, effectively improving the local exploitation capability of ABC [19]; Gao and Liu enhanced the global search performance of the algorithm by introducing a chaotic system and an opposition-based learning mechanism [20]; Yu et al. designed an adaptive greedy position update strategy that dynamically adjusts search behavior according to the evolutionary state [21]; Zhou et al. proposed a Gaussian bare-bones ABC, replacing the original search equation with Gaussian distribution sampling to improve convergence performance [22]. These studies confirm the positive role of incorporating high-quality individual information in improving ABC performance from different perspectives. On this basis, while focusing on high-quality individual information, further integration of population diversity maintenance mechanisms is advisable, and in addition to utilizing high-quality information, more attention should be paid to the diversity and structural information embodied by individuals within the population. This helps accelerate convergence while maintaining population diversity, thereby enhancing global search capability and robustness.

To alleviate the above issues, neighborhood topology-based ABC variants have emerged in recent years. The core idea of such variants is to guide the search using the best neighbor individual within a predefined neighborhood topology. Peng et al. designed a best neighbor-guided ABC (NABC), constructing a random neighborhood topology for each individual and using the best individual in the neighborhood as the search basis [23]. Wang et al. proposed NSABC based on a ring neighborhood topology, replacing the probabilistic selection in the onlooker bee phase with the best neighbor individual [24]. These methods reduce dependence on the global elite to some extent. Building on this foundation, a dynamic adjustment mechanism can be further introduced to adapt the search mode to changes in the optimization process; simultaneously, when selecting neighborhood individuals, in addition to fitness information, the spatial distribution characteristics reflected by distance information among individuals in the population can also be incorporated to enhance the matching capability between search behavior and the optimization stage.

In recent years, the idea of integrating multiple search strategies with complementary characteristics has gained attention. Most existing multi-strategy ABC methods construct a strategy pool and select search strategies based on historical experience. Wang et al. proposed MEABC, integrating three different solution search equations into a strategy pool and dynamically selecting strategies based on the success experience of the previous round [25]. Xiang et al. developed ABCG, employing multiple solution search equations with different search characteristics and determining strategy selection through newly defined parameters [26]. Xiao et al. proposed ABCNG, integrating three mechanisms: adaptive neighborhood search, global best guidance, and Gaussian perturbation [27]. On this basis, real-time perception capability of the optimization process stage can be further introduced into strategy switching, enabling the algorithm to dynamically adjust search behavior according to whether it is in the early or later stage of optimization, thereby enhancing the matching degree between strategy selection and the evolutionary process.

From a macro perspective, Fitness Landscape Analysis (FLA) has become a tool for understanding problem characteristics and guiding algorithm design [28,29]. Some ABC variants have begun to introduce FLA techniques, selecting search strategies by considering problem characteristics. Zhou et al. proposed FLABC, using online FLA techniques to dynamically identify landscape features such as morphology and ruggedness, adjusting search strategies in real time [30]. Zhou et al. developed ABC-AMP, quantifying landscape features as smooth, rugged, or neutral types using fitness-distance correlation and dynamically adjusting population strategies [31]. These studies demonstrate the potential of state-aware design concepts. On this basis, algorithms should place greater emphasis on perceiving internal states during their own operation such as population convergence degree, diversity level, etc., and use this as the core basis for strategy adjustment to enhance the algorithm's adaptability to its own evolutionary process.

Based on the above analysis, when existing ABC algorithms are applied to high-dimensional, dynamic, and multi-constrained problems, their performance bottlenecks can be attributed to three critical issues, corresponding respectively to the search space characteristics, search process stage, and population convergence state of the algorithm.

First, algorithms should possess the capability to perceive problem dimensionality states. Fitness landscape analysis techniques primarily focus on the external landscape characteristics of the problem, while dimensionality information, as a fundamental attribute of the search space, has not yet had its utilization value fully exploited. During algorithm operation, dynamically adjusting strategic focuses according to dimensionality, maintaining sufficient random exploration for low-dimensional problems and ensuring uniform spatial coverage capability for high-dimensional problems, is an important direction for further enhancing algorithm adaptability.

Second, algorithms should possess the capability to perceive search process states. In existing methods, once the core mode of the search strategy is determined, it often remains unchanged, yet different stages of the optimization process have different requirements for search behavior. The early stage requires sufficient exploitation capability to achieve precise localization, while the late stage requires maintaining a certain exploration capability to escape local optima. Therefore, dynamically adjusting the granularity and range of the search according to process advancement to achieve adaptive matching between search behavior and optimization stage is a key link in enhancing the algorithm's global search capability.

Third, algorithms should possess the capability to perceive population

convergence states. The standard selection mechanism struggles to maintain effective selection pressure after population convergence, and the degree of population convergence is an important basis for determining whether to enter the refined search stage. Dynamically adjusting the selection mechanism according to the actual convergence state of the population and reshaping selection pressure at appropriate opportunities to support the algorithm's refined search near the optimal solution is an effective way to improve the algorithm's convergence performance.

By perceiving and utilizing multi-dimensional state information during its own operation, including dimensionality characteristics, process stage, and convergence degree, the algorithm provides an important basis for adaptively adjusting search behavior. This is a key direction for further improving ABC algorithm performance. Therefore, this paper proposes the State-Sensing Adaptive Artificial Bee Colony (SSA-ABC) algorithm, which adaptively adjusts initialization, search, and selection strategies through three key dimensions of state information, including problem dimensionality state, search process state, and population ranking state, while incorporating the utilization of individual distance information to compensate for the deficiency of existing neighborhood methods that rely only on fitness information.

(1) A dynamic hybrid initialization strategy based on the sigmoid function is designed, enabling the algorithm to perceive the problem dimensionality state and adaptively adjust the mixing ratio of Latin hypercube initialization and random initialization—biasing towards random exploration in low dimensions to maintain diversity and towards Latin hypercube in high dimensions to ensure uniform spatial coverage. Simultaneously, a reflection-hybrid mapping boundary processing method is introduced to eliminate the boundary clustering problem at the source.

(2) A dynamic dimension adjustment mechanism based on the function evaluation ratio is designed, enabling the algorithm to perceive the search process state and gradually increase the search dimension as the process advances—adopting one-dimensional search in the early stage to focus on local refinement and multi-dimensional search in the late stage to expand the exploration range. For multi-dimensional search, an elite-mean-guided dimension selection mechanism is designed: half of the dimensions preferentially select dimensions with large differences from the elite to accelerate convergence, while the other half are randomly selected to maintain diversity.

(3) A two-stage dynamic probability selection mechanism is designed, enabling the algorithm to perceive the population ranking state: roulette wheel selection is used in the early stage to maintain extensive exploration; in the late stage, a linearly shrinking elite nucleus is constructed based on the population ranking state, and arithmetic gravitational gradients distributed according to rank are applied to enhance the selection advantage of high-ranking individuals. Simultaneously, elite immunity protection is provided to ensure that high-quality solution information is not lost.

The above three components together constitute a complete state-aware adaptive optimization framework: dimensionality state determines the initialization method, process state governs search behavior, and population state determines the selection focus. By embedding these three types of key state information into the algorithm's decision-making process, SSA-ABC achieves full-process state awareness and adaptive control covering initialization, search, and selection. The subsequent sections of this paper are arranged as follows: Section 2 elaborates on the principles of the standard ABC algorithm and the mathematical model of robot path planning; Section 3 details the components and complete workflow of the SSA-ABC algorithm; Section 4 verifies

the effectiveness of the algorithm through comparative experiments on standard test functions and path planning scenarios; Section 5 summarizes the paper and discusses future research directions.

2 Standard Algorithm and Problem Model

2.1 Standard ABC Algorithm

The ABC algorithm primarily simulates the foraging process of bees, divided into four phases: initialization phase, employed bee phase, onlooker bee phase, and scout bee phase. The workflow of the standard ABC algorithm is as follows:

1) Initialization Phase

ABC initiates the search process by randomly initializing the population. Each employed bee is assigned to a randomly generated food source. Let the i -th food source be represented as $x_i = (x_{i,1}, x_{i,2}, \dots, x_{i,j}, \dots, x_{i,D})$, $i \in \{1, 2, \dots, SN\}$, $j \in \{1, 2, \dots, D\}$, where SN represents the number of food sources and D represents the optimization problem dimensionality. This food source is generated by (1):

$$x_{i,j} = x_j^{min} + rand(0,1) * (x_j^{max} - x_j^{min}) \quad (1)$$

where x_j^{min} and x_j^{max} represent the lower and upper bounds of the j -th dimension of the optimization problem, respectively, and $rand(0,1)$ represents a uniform random number on $[0,1]$.

2) Employed Bee Phase

After randomly initializing the population, employed bees perform foraging by conducting local searches within the neighborhood of food sources. Let v_i represent the updated food source. Employed bees update food source positions using the search equation shown in (2):

$$v_{i,j} = x_{i,j} + \varphi * (x_{i,j} - x_{k,j}) \quad (2)$$

where φ is a uniform random number on $[-1,1]$, x_k is a randomly selected food source such that $x_k \neq x_i$, and j is a randomly selected dimension.

After completing the local search, the employed bee makes a greedy choice between the current food source and the updated food source, retaining the better of the two.

3) Onlooker Bee Phase

Onlooker bees collect food source information from all employed bees and select promising sources for honey collection based on their probability values. A higher fitness value increases the probability of a source being selected. The fitness of a food source is calculated by (3):

$$fit_i = \begin{cases} \frac{1}{1+f(x_i)} & f(x_i) \geq 0 \\ 1 + abs(f(x_i)) & f(x_i) < 0 \end{cases} \quad (3)$$

where fit_i represents the fitness value of x_i and $f(x_i)$ represents the objective function value.

Onlooker bees perform roulette wheel selection based on food source fitness. Sources with higher nectar yield (i.e., better objective function values) have greater probabilities of being selected. The probability value for each source can be calculated by (4):

$$P_i = \frac{fit_i}{\sum_{j=1}^{SN} fit_j} \quad (4)$$

where P_i represents the selection probability of x_i .

4) Scout Bee Phase

This phase prevents excessive computational resources from being allocated to exhausted food sources, thereby avoiding search stagnation. During the employed bee and onlooker bee phases, a *Trial* counter is assigned to each food source to track successful updates. If successful, the counter resets to 0; if unsuccessful, it increments by 1.

When the counter value exceeds the threshold *limit*, the food source is deemed no longer exploitable. The associated employed bee transforms into a scout bee, which then reinitializes a new food source by (1).

After initialization, the standard ABC algorithm iteratively executes three search phases—employed bee, onlooker bee, and scout bee—until it reaches the maximum function evaluations *Max_FEs* or achieves acceptable solution accuracy.

2.2 Robot Path Planning Problem Model

1) Problem Description

Robot Path Planning (RPP) involves finding an optimal motion path for a robot to travel from its starting position to an end position within a known environment. The objective is to generate a path that satisfies obstacle avoidance constraints while minimizing total cost. The path planning problem addressed in this paper can be formally described in the following aspects:

(1) Robot: Consider a robot represented as R , with starting point $S=(x_s, y_s)$ and end point $E=(x_e, y_e)$. The task is to plan a path P for R to move from S to E .

(2) Environment and Obstacles: The robot navigates a two-dimensional static environment containing obstacles. The environment is modeled as a grid map where the positions, shapes, and dimensions of all obstacles are fully known. Let the environment contain M obstacles, defined as the set $O=\{O_1, O_2, \dots, O_M\}$, where $O_i=(x_i, y_i)$ represents the geometric center coordinates of obstacle O_i .

(3) Distance and Cost: The optimal path P should have the minimum total length, which is typically used to measure key performance indicators such as time cost and energy consumption.

(4) Objective and Constraints: The optimization objective is to generate a collision-free path P that minimizes the total path length while ensuring the robot safely avoids all obstacles.

2) Model Assumptions

To simplify the problem and establish an effective mathematical model, the following assumptions are made:

(1) Global Information Availability: The robot precisely knows its starting point S , end point E , and the complete grid map of the environment, including the geometric information of all static obstacles.

(2) Planning and Optimization Process: Path planning employs a two-stage strategy of initialization followed by optimization. An initial path is generated using a deterministic feasible path initialization method (e.g., vertical offset method). Subsequently, a swarm intelligence optimization algorithm iteratively refines the sequence of path nodes until either a preset maximum number of iterations is reached or no feasible path can be found.

(3) Robot Physical Dimension Modeling: The robot is treated as a grid-based entity with actual physical dimensions rather than an idealized point mass during collision

detection. A safety distance Q (typically defined as the robot's diagonal length) is introduced to ensure path safety.

(4) Static Environment: All obstacles maintain fixed positions and configurations within the workspace, remaining unchanged over time.

3) Objective Function

The robot path planning problem is formalized as a multi-constraint optimization problem simultaneously considering path length and collision avoidance safety. These objectives are unified by defining a composite objective function f , as shown in (5):

$$f = f_{length} + f_{collision} \quad (5)$$

(1) Path Length Cost Function

The path length f_{length} is calculated by summing the Euclidean distances between adjacent nodes, as shown in (6):

$$f_{length} = \sum_{i=1}^{N-1} \sqrt{(P_{i+1,x} - P_{i,x})^2 + (P_{i+1,y} - P_{i,y})^2} \quad (6)$$

where N represents the number of path nodes in a robot path P , $P = \{P_1, P_2, \dots, P_N\}$, where P_i is the i -th path node of the robot, P_i and P_{i+1} are two adjacent nodes in the path, $P_{i,x}$ is the x -coordinate of P_i , $P_{i,y}$ is the y -coordinate of P_i , and f_{length} represents the total path length obtained by summing all line segments.

(2) Collision Penalty Function

To ensure path safety, a collision penalty is applied when the distance between a path node P_i and an obstacle O_j is less than the safety distance Q , defined as the robot's diagonal length, as shown in (7):

$$Penalty_{i,j}(P_i, O_j) = \begin{cases} 2 \times \sqrt{w^2 + h^2} & \text{if } d(P_i, O_j) < Q \\ 0 & \text{else} \end{cases} \quad (7)$$

where $d(P_i, O_j)$ is the Euclidean distance between the i -th node of the robot's path and the j -th obstacle; w and h are the width and height of the map, respectively, ensuring that the penalty function returns a large value, thereby increasing the value of the total objective function. From this, the collision penalty function $f_{collision}$ for the entire path is obtained, as shown in (8):

$$f_{collision} = \sum_{i=1}^N \sum_{j=1}^M Penalty_{i,j}(P_i, O_j) \quad (8)$$

(3) Optimization Objective

The robot path planning problem ultimately boils down to finding the path P^* that minimizes the composite objective function f . This function design ensures the optimization algorithm simultaneously balances path length and safety within the search space: the collision penalty term $f_{collision}$ excludes unsafe paths from the preferred solutions, while the path length term f_{length} drives the selection of shorter paths among all safe paths.

3 State-Sensing Adaptive Artificial Bee Colony Algorithm

3.1 Motivation

The standard artificial bee colony algorithm faces three major challenges when addressing high-dimensional complex optimization problems such as robot path planning: First, random initialization struggles to generate initial populations that balance both quality and diversity within high-dimensional solution spaces, while traditional boundary truncation methods often induce boundary clustering, limiting global exploration potential; Second, the fixed one-dimensional search update strategy fails to adaptively adjust exploration and exploitation intensity based on search progress,

resulting in insufficient depth in the early stage and limited breadth in the later stage; Third, the roulette wheel selection mechanism during the onlooker bee phase diminishes selection pressure as the population converges, significantly slowing the convergence rate in the later stage, and high-quality solutions are prone to being discarded due to stagnation counter misjudgments, further compromising solution accuracy.

To address these issues, this paper proposes SSA-ABC, whose main concept is to endow bees with the ability to sense states and adaptively adjust behaviors: sensing problem dimensionality during initialization, sensing search progress during search, and sensing population rankings during selection. Based on this, a three-stage adaptive optimization framework is constructed, encompassing three key technologies: dimensionality-sensing hybrid initialization and reflection-hybrid mapping boundary processing; process-sensing dynamic dimension search and elite-mean-guided dimension selection; ranking-sensing elite immunity protection and gravity gradient selection model.

3.2 State-Sensing Mechanism I: Dimension-Sensing Hybrid Initialization and Reflection-Hybrid Mapping Boundary Processing

1) Dynamic Hybrid Initialization Strategy Based on Sigmoid Function

In swarm-based optimization algorithms, the quality and diversity of the initial population directly determine the efficiency of subsequent searches and the potential for global optimization. While traditional random initialization methods are simple to implement and facilitate extensive, unbiased exploration in low-dimensional or structurally simple solution spaces due to complete randomness, they often result in uneven distribution of individuals across vast solution spaces when handling high-dimensional, complex path planning problems. This leads to poor initial population quality and slows overall convergence. In contrast, Latin hypercube sampling, as a systematic hierarchical sampling technique, ensures uniform distribution across each dimension. This enables effective coverage of high-dimensional solution spaces with fewer sampling points, providing the algorithm with a more systematic exploration starting point. However, overreliance on this method may weaken the population's inherent random exploration capability.

To address this, a dynamic hybrid initialization strategy based on the sigmoid function is proposed to enable the algorithm to sense the current optimization problem's dimensionality. Based on the dimensionality value, it dynamically and smoothly adjusts the composition proportion of Latin hypercube sampling and random initialization within the population. This achieves an adaptive balance between broad random exploration and systematic uniform coverage, generating an initial population for complex optimization problems that combines high quality and high diversity, as shown in (9):

$$LatinProportion = 0.1 + 0.8 \times \frac{1}{1 + e^{-0.1(D - D_{mid})}} \quad (9)$$

where *LatinProportion* represents the proportion of initial population individuals initialized using Latin hypercube sampling, and D_{mid} represents the geometric center of the problem's dimensional range. To ensure the proportion remains within a reasonable range, the calculation result is constrained: such that $LatinProportion \in [0.1, 0.9]$. Furthermore, when initializing individuals using Latin hypercube sampling, uniformly distributed sequences (center points plus small perturbations) are generated for each

dimension. These sequences are then randomly permuted using the Fisher-Yates shuffling algorithm to ensure the initial population exhibits good spatial filling properties across all dimensions.

The main advantage of this strategy lies in its intelligent dimensionality adaptation: when the algorithm detects relatively low problem dimensionality, the sigmoid function outputs smaller proportional values, causing the algorithm to favor a higher proportion of random initialization. This fully leverages its free diffusion capability within the solution space to enhance population diversity. Conversely, when the algorithm senses a significant increase in problem dimensionality, the proportional value smoothly increases. The algorithm automatically enhances the weight of Latin hypercube sampling, leveraging its multi-dimensional collaborative distribution characteristics to ensure systematic exploration in high-dimensional solution spaces. This effectively counters potential initial distribution imbalances triggered by the curse of dimensionality. This parameter adaptation mechanism, based on problem dimensionality, enables the algorithm to dynamically balance unconstrained global diffusion with structured spatial traversal. It establishes a dynamic balance between breadth exploration and depth coverage in the solution space, laying the groundwork for subsequent refined searches.

Fig.1 illustrates the dynamic variation of *LatinProportion* based on the sigmoid function with problem dimension D , using $D_{mid} = 50$ as an example. The curve exhibits an inflection point at $D = 50$.

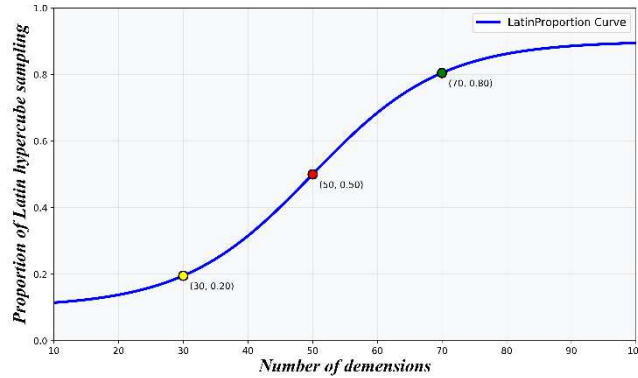


Figure 1 Dynamic variation of *LatinProportion* with D (Using $D_{mid}=50$ as an example)

In low-dimensional regions ($D < 30$), the proportion approaches the lower bound of 0.1, where the algorithm prioritizes random initialization, emphasizing global exploration and maintaining population diversity. In the high-dimensional region ($D > 70$), the proportion approaches the upper bound of 0.9, and the algorithm shifts to prioritize Latin hypercube sampling. Leveraging its hierarchical space-filling properties to address the curse of dimensionality, this method ensures uniform coverage of the initial population across the high-dimensional solution space. The sigmoid function enables a smooth transition, preventing abrupt parameter shifts. This demonstrates the initialization strategy's adaptive capability to problem dimensionality, laying a solid foundation for subsequent optimization through an optimized initial population.

Fig.2 compares the distribution characteristics of the three initialization strategies at $D=50$ within the range $[-100, 100]$, while Table I presents the normalized uniformity metrics.

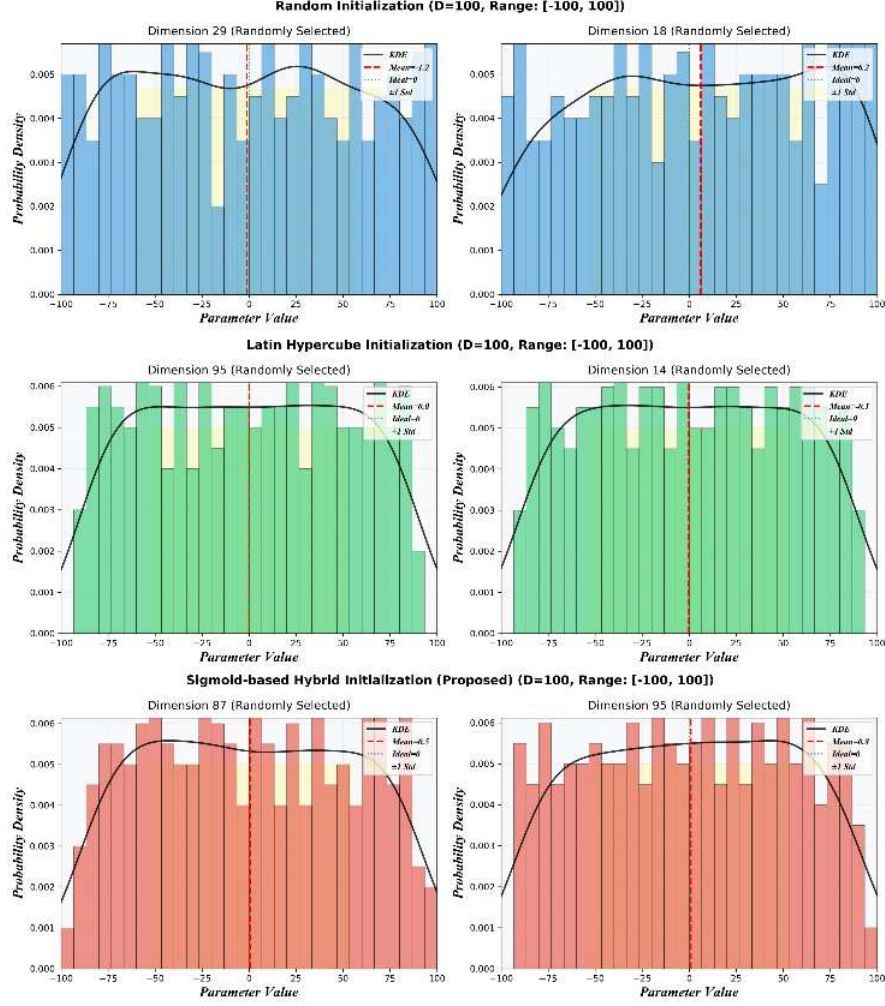


Figure 2 Distribution characteristics of three initialization strategies

TABLE I Comparison of normalized uniformity metrics for three initialization strategies

Initialization Method	Mean Deviation	Normalized Standard Deviation	Normalized Maximum Gap	Entropy Ratio
Random Initialization	2.49	0.2882	0.0209	0.9898
Latin Hypercube Initialization	0.05	0.2606	0.0140	0.9709
Hybrid Initialization (This Paper)	1.12	0.2663	0.0263	0.9822

Table I reveals that random initialization exhibits a mean deviation of 2.49, significantly higher than that of Latin hypercube (0.05) and the proposed method (1.12). Its normalized standard deviation (0.2882) and maximum gap (0.0209) are the highest among all three, confirming clustering and gaps in the dimensional space. Latin hypercube demonstrates optimal performance in uniformity: a mean deviation of 0.05, normalized standard deviation of 0.2606, and maximum gap of 0.0140, demonstrating systematic coverage. However, its entropy ratio of 0.9709 is lower than that of random initialization, suggesting that excessive structure may reduce diversity. The dynamic hybrid strategy proposed in this paper achieves a favorable balance at $D = 50$, with a Latin hypercube ratio of 49.1% and a random ratio of 50.9%. Its mean deviation of 1.12 represents a 55.0% reduction compared to random sampling, while the normalized

standard deviation of 0.2663 approaches that of Latin hypercube, and the entropy ratio of 0.9822 outperforms that of Latin hypercube. This strategy successfully balances uniform coverage with random exploration, providing a stable and diverse initial solution set for optimization problems.

2) Boundary Handling Mechanism for Reflection-Hybrid Mappings

During the search process of swarm intelligence optimization algorithms, newly generated individuals may fall outside the predefined feasible region. Traditional boundary handling methods, such as direct truncation, are simple but tend to cause solutions to cluster near the boundaries, reducing population diversity and potentially leading to premature convergence. Particularly in high-dimensional path planning scenarios, this boundary clustering phenomenon significantly weakens the algorithm's exploration capability.

To address this issue, this paper proposes a reflection-hybrid mapping boundary handling mechanism. This mechanism first maps out-of-bounds individuals to mirrored positions within the feasible region via a symmetric reflection operation, followed by a truncation correction to ensure that all values strictly satisfy the constraint conditions. Specifically, for an out-of-bounds individual x_i , the reflection mapping rule is given by (10):

$$x'_{i,j} = \begin{cases} 2x_j^{min} - x_{i,j} & \text{if } x_{i,j} < x_j^{min} \\ 2x_j^{max} - x_{i,j} & \text{if } x_{i,j} > x_j^{max} \\ x_{i,j} & \text{otherwise} \end{cases} \quad (10)$$

The core idea of the reflection operation is to fold the distance exceeding the boundary back into the feasible region, thereby effectively dispersing individuals that might otherwise cluster near the boundary toward the central region of the solution space, avoiding the boundary clustering effect caused by traditional truncation methods. Subsequently, a truncation correction is applied to ensure all variables remain strictly within bounds.

This two-step strategy of "reflection first, then truncation" not only preserves parameter feasibility but also enhances population diversity near the boundary through mirror reflection. Compared to the direct truncation method, this mechanism more effectively maintains the uniformity of solution distribution, enhances the algorithm's exploration potential near the boundaries of the feasible region, and lays a more balanced foundation for subsequent global search and local optimization.

3) Summary

In summary, the dimensionality-sensing sigmoid dynamic hybrid initialization method adaptively combines Latin hypercube sampling and random initialization based on the problem's dimensionality. It emphasizes random exploration in low-dimensional spaces and shifts toward systematic space-filling in high-dimensional spaces, thereby providing an initial population that balances uniformity and diversity for subsequent optimization. Meanwhile, the reflection-hybrid boundary handling mechanism maps out-of-bounds individuals to mirrored positions within the feasible region via symmetric reflection, followed by truncation correction to ensure strict feasibility. This significantly alleviates the boundary clustering issue caused by traditional direct truncation methods. Addressing population initialization and search process maintenance together synergistically demonstrate the algorithm's capability to perceive and adapt to the problem state, providing a diverse population foundation for solving complex optimization problems.

3.3 State-sensing Mechanism II: Elite-Mean-Guided Dynamic Dimension Search

1) Dynamic Selection Mechanism for Dimension Search Strategies

In swarm intelligence optimization algorithms, the selection of search dimension directly influences the algorithm's exploration breadth and exploitation depth within the solution space. Traditional methods often employ a fixed-dimension search mode, which struggles to adapt to the changing characteristics of the algorithm across different search stages. In the early stage of search, when population diversity is high, low-dimensional search is preferred for focusing on the in-depth exploitation of local regions. In contrast, in the later stage, as the population converges, the search dimension must be increased to expand the exploration range and avoid getting trapped in local optima.

To address this issue, this paper introduces an adaptive dimension adjustment mechanism based on the ratio of function evaluations. The search dimension dn is dynamically adjusted throughout the search process, and its calculation is given by (11):

$$dn = 1 + \left\lceil \left(\frac{D}{10} - 1 \right) \times \sin^2 \left(\frac{FEs}{Max_FEs} \right) \right\rceil \quad (11)$$

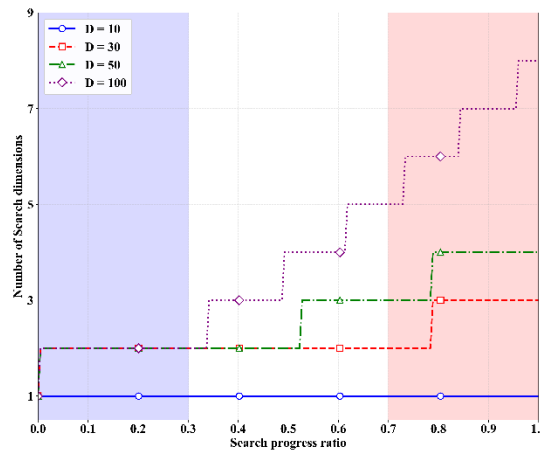
where FEs represents the current number of function evaluations, and Max_FEs represents the maximum number of function evaluations.

Equation (11) quantifies the search process using the ratio FEs/Max_FEs , enabling adaptive adjustment of the search dimension based on the search progress and problem scale. In the early stage of the search, dn is set to its minimum value of 1, meaning a one-dimensional search is performed, focusing on the fine-tuning of each dimension to quickly identify promising regions. As the search progresses, dn gradually increases, reaching on the order of $D/10$ in the later stage. By increasing the number of dimensions updated simultaneously, the search stride and exploration range are expanded, effectively maintaining population diversity and avoiding premature convergence.

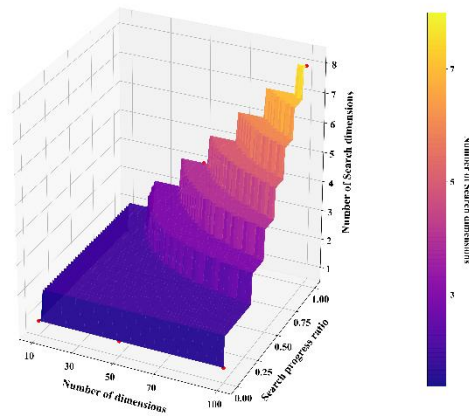
Fig.3(a) shows the changing trend of the search dimension dn as a function of the search progress ratio FEs/Max_FEs under different problem dimensions D . In the initial stage (blue region), all curves have $dn=1$, enabling one-dimensional fine-tuning exploitation to quickly identify promising regions; In the middle stage, dn gradually increases with a smooth transition; In the later stage (red region), dn reaches the order of $D/10$, expanding the exploration region by adding new dimensions to avoid local optima. The family of curves indicates that this mechanism can adaptively adjust search dimension based on problem scale: the larger D is, the more significant the range of dn becomes, satisfying both the fine-tuning requirements of low-dimensional problems and the exploration demands of high-dimensional problems, thereby achieving intelligent matching between search behavior and problem scale.

Fig.3(b) presents the functional relationship between dn , D , and FEs/Max_FEs using a three-dimensional surface. The surface rises monotonically in two directions; in the initial plateau region ($FEs/Max_FEs \approx 0$), $dn=1$, ensuring the algorithm starts with a one-dimensional fine-tuning search; at the later stage endpoint, $dn \approx D/10$, demonstrating adaptability to problem scale. The smooth and continuous surface demonstrates that the $\sin^2$ function achieves gradual adjustment of the dimension, avoiding abrupt changes. Red markers highlight the start and end values at key dimensions ($D = 10, 50, 100$). This surface clearly reveals the core of the strategy: the search dimension increases progressively with the process and scales with the problem dimension, focusing on exploitation in the early stage and strengthening exploration in

the later stage, thereby achieving an adaptive balance between exploration and exploitation.



(a) Analysis of trends in search dimension changes under different problem dimensions



(b) Three-dimensional response surface analysis of search dimension
Figure 3 Analysis of dynamic selection of dimensional search strategies

To accelerate convergence while maintaining sufficient exploration randomness, this paper designs different dimension selection strategies for one-dimensional and multi-dimensional searches.

(1) Dimension Selection Strategy for One-Dimensional Search

In the one-dimensional search mode, to ensure unbiased and comprehensive exploration of all dimensions of an individual, the dimension to be updated is selected using a purely random approach. This strategy ensures that each dimension has an equal opportunity to be thoroughly searched, avoiding the omission of potential optimal regions due to dimensional selection bias, and providing a foundation of uniform dimensional coverage for in-depth local exploitation.

(2) Selection Strategy for Multi-dimensional Search

In the multi-dimensional search mode, to leverage the successful experience of elite individuals to guide the search direction while maintaining population diversity, a hybrid selection mechanism of “elite-mean-guided + random replenishment” is adopted. The specific steps are as follows:

- ① Sort the current population by fitness to identify the global optimal individual

x_{best} ;

② Randomly select two distinct individuals, x_{e1} and x_{e2} , from the top 5% and top 10% elite individuals;

③ Calculate the elite mean vector $x_{\overline{elite}}$, where the values in each dimension are the arithmetic averages of x_{best} , x_{e1} and x_{e2} for that dimension, as shown in (12):

$$x_{\overline{elite},j} = \frac{x_{best,j} + x_{e1,j} + x_{e2,j}}{3} \quad (12)$$

④ For each individual x_i , calculate the Euclidean distance between its dimensions and the elite mean $x_{\overline{elite}}$, as shown in (13):

$$dist_{i,j} = |x_{i,j} - x_{\overline{elite},j}| \quad (13)$$

⑤ Sort all dimensions of x_i in descending order of $dis_{i,j}$;

⑥ Select $[dn/2]$ half dimensions with the largest distances as elite-mean-guided dimensions (i.e., dimensions with significant differences from the elite, which may have greater potential for improvement), and randomly select the remaining half from the rest of the dimensions.

This strategy constructs spatial reference points using the mean of the elite population, enabling the algorithm to sense the dimensional deviation between an individual and the elite region: the larger the distance in a given dimension, the further the individual deviates from the elite population in that dimension, potentially indicating greater room for optimization or a need for adjustment. By prioritizing updates to these dimensions, the algorithm accelerates the convergence of individuals toward high-quality regions; meanwhile, the introduction of random dimensions prevents the search scope from becoming too narrow due to excessive focus, thereby maintaining the necessary degree of exploration freedom. In this way, an effective balance is achieved between convergence speed and global exploration.

2) Capability-Balanced Dimension Search Equations

To complement the exploration and exploitation characteristics of search strategies across different dimensions, this paper designs corresponding search equations for one-dimensional and multi-dimensional searches. This enables the algorithm to recognize the current search mode and invoke the appropriate update strategy, thereby further optimizing overall search performance.

(1) One-Dimensional Search Equation

In one-dimensional search, the search equation of the standard ABC is adopted, as shown in (2). By introducing random perturbations and variation between individuals along a single dimension, this equation exhibits strong exploration characteristics. It facilitates detailed exploration within local regions and helps escape the neighborhood of local extrema, making it suitable for in-depth exploitation along a single dimension.

(2) Multi-dimensional Search Equation

In multi-dimensional search, the Global Optimal-Guided Ant Colony Algorithm (GABC) search equation [17] is adopted, as shown in (14):

$$v_{i,j} = x_{i,j} + \phi_{i,j} * (x_{best,j} - x_{r1,j}) \quad (14)$$

where $\phi_{i,j}$ is a random number on $[0, 1.5]$, and x_{r1} is a randomly selected food source distinct from x_i .

This equation adds a learning term toward the global optimal solution to the standard equation, significantly enhancing the exploration capability. The first term maintains a local perturbation of the current solution, while the second term introduces a directed guidance toward the global optimum, enabling the search to converge simultaneously toward potential optimal regions across multiple dimensions. This

makes it suitable for accelerating optimization and improving convergence accuracy in multi-dimensional spaces.

By pairing the highly explorative one-dimensional search equation with the highly exploitative multi-dimensional search equation alongside their respective search modes, the algorithm achieves effective complementarity between fine-tuning exploitation and global exploration: the one-dimensional search focuses on independent, fine-tuning in each dimension, while the multi-dimensional search emphasizes rapid convergence through multi-dimensional coordination.

3) Summary

In summary, the proposed process-sensing dynamic-dimension search strategy intelligently responds to exploration and exploitation requirements at different search stages through adaptive adjustment of the search dimension. By integrating elite-mean-guided dimension selection with random selection, the algorithm accelerates convergence while maintaining population diversity. By assigning characteristic-matched search equations to different search modes, it further optimizes global and local search capabilities. These three layers of design work synergistically, demonstrating the algorithm's ability to sense and adapt to the optimization process, thereby enhancing search efficiency and solution accuracy in high-dimensional complex spaces.

3.4 State-Sensing Mechanism III: Ranking-sensing Elite Immunity Protection and Gradient-Driven Selection

1) Elite Protection Immunity Mechanism Preventing Loss of Information from High-Quality Solutions

In the standard ABC algorithm, the *Trial* counter is used to identify individuals stuck in local stagnation and triggers a reset operation during the scout phase. However, in the later stage of optimization, some elite individuals may be mistakenly identified as stagnant due to the slow pace of local fine-tuning, resulting in their reinitialization and causing unnecessary loss of the population's core competitiveness. This loss of elite individuals severely weakens the algorithm's convergence capability in the later stage. To address this, this paper proposes an elite immunity protection strategy.

This strategy performs an intervention before each scout phase: first, the population is sorted by fitness, and then the *Trial* counters of the top 10% of individuals are reset to zero. This operation grants elite individuals immunity, shielding them from elimination by the stagnation detection mechanism during the scout bee phase.

The core of this strategy lies in establishing a dual mechanism of elite screening and protective barriers: the elite screening mechanism dynamically identifies elites within each generation through fitness ranking, demonstrating the algorithm's real-time sensing of individual quality; the protective barrier mechanism provides elite individuals with survival guarantees throughout the iteration cycle by resetting their counters, ensuring the continuity of the population's core evolution momentum. This mechanism effectively prevents the unnecessary loss of high-quality solutions at a very low computational cost, providing a stable foundation for the algorithm's convergence in the later stage.

2) Elite-Driven Selection Model: A Dynamic Focusing Mechanism Based on Quality Gradient

Based on elite preservation, this paper proposes a ranking-sensing gradient-based

selection model to further enhance the algorithm's convergence efficiency in the later stage. This model simulates the gravitational field effect observed in physics, enabling the algorithm to sense the ranking of individuals within the population. Based on this, it constructs a probability field that evolves with the optimization process, allowing search resources to intelligently concentrate on high-quality elite individuals and achieving adaptive adjustment of selection pressure.

(1) Phase 1: Early-Stage Exploration Guided by a Uniform Attraction Field

When the algorithm's process $FES/Max_FES \leq 0.6$, a roulette wheel selection mechanism is employed to construct a uniform gravitational field. This phase follows traditional roulette wheel selection logic, where an individual's selection probability is proportional to its fitness value, as shown in (4). This mechanism assigns selection probabilities to each individual based on the magnitude of their fitness value; higher-fitness individuals are more likely to be selected, while lower-fitness individuals retain a non-zero probability of being selected. This design enables the algorithm to maintain sufficient spatial exploration capability during the early search stage, while simultaneously providing a foundational guide for the population to converge toward promising regions.

(2) Phase 2: Later-Stage Elite Guided by a Concentration Gravitational Field

When the process $FES/Max_FES > 0.6$, the algorithm switches to the concentration gravitational field mode. This mode enhances elite guidance through two components:

① Dynamic Formation and Contraction of the Elite Nucleus

The algorithm constructs a dynamically evolving elite subpopulation from the current population, referred to as the elite nucleus. Its size, N_{core} , contracts linearly as the optimization process progresses, as shown in (15):

$$N_{core} = \lfloor SN \times [0.15 + 0.05 \times (1 - FES/Max_FES)] \rfloor \quad (15)$$

The contraction range of N_{core} extends from an initial 15% to a final 10%. This moderate contraction mechanism ensures the concentration of the later-stage search while avoiding the diversity crisis caused by excessive elitism.

② Intelligent Allocation of the Gravitational Gradient

Within the elite core, a ranking-based gravitational gradient allocation mechanism is employed. For the k th-ranked individual in the elite core, the gravitational coefficient G_k is calculated as shown in (16):

$$G_k = m - \frac{(m-1)(k-1)}{N_{core}-1} \quad (16)$$

where m is the selection pressure factor that controls the amplitude of probability differences and is typically set as a constant. This equation ensures the establishment of a clear probability gradient within the core group. Furthermore, a normalization method is used to convert the gravitational coefficient G_k into selection probabilities P_k , as shown in (17):

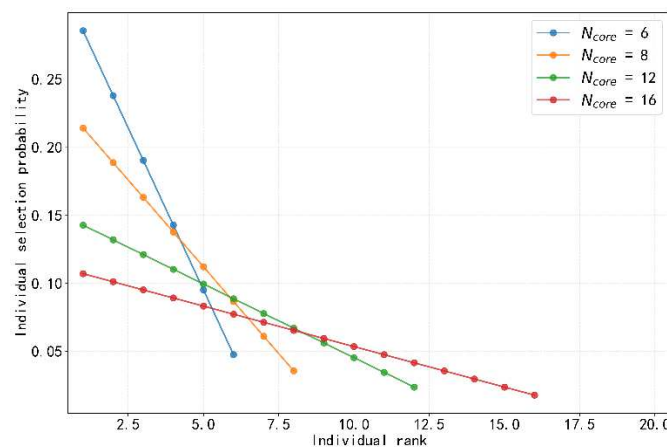
$$P_k = \frac{G_k}{\sum_{k=1}^{N_{core}} G_k} \quad (17)$$

This model introduces an elite core, transforming the later-stage selection process from evaluation of all individuals to elite competition, and systematically amplifies the competitive advantage of top-layer elites through an arithmetic ranking mechanism. This ensures that during the critical convergence phase, the algorithm's search direction is strongly guided by a small number of the most promising solutions, thereby enabling a refined exploration of the neighborhood of the optimal solution. The core driver of the entire mechanism is the algorithm's precise sensing of individual rankings: rankings determine the gravitational coefficients, gravitational coefficients determine the

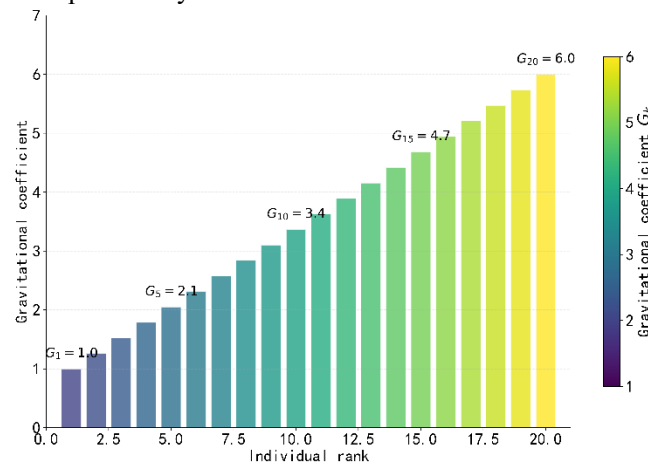
selection probabilities, and selection probabilities determine the allocation of search resources.

Fig.4(a) illustrates the arithmetic decay of the selection probability P_k with respect to rank k under different elite core sizes N_{core} . When $N_{core}=6$, the curve is steepest; the selection probability for the top individual ($k=1$) reaches 0.2857, while that for the lowest-ranked individual ($k=6$) is only 0.0476, demonstrating a strong elite bias. As N_{core} increases to 16, the curve gradually flattens, and P_1 drops to 0.0968, indicating that competitive pressure is relatively dispersed. This arithmetic distribution mechanism allows for fine-tuning of selection pressure by adjusting P_1 : a larger P_1 enhances the ability to focus the search in the later stage, while a smaller P_1 maintains population diversity and prevents the algorithm from converging prematurely.

Fig.4(b) visually illustrates the gravitational coefficient G_k as a heatmap. As the rank k ranges from 1 to 20, G_k uniformly decreases from 6.0 to 1.0. The colors, ranging from dark to light, correspond to gravity decreasing from high to low, forming a clear gradient of competitive advantage. Key G_k values are annotated in the figure, illustrating the arithmetic decay characteristic. This design ensures that search resources in the later stage of the algorithm are concentrated on top-ranking individuals. G_k is normalized to form the selection probability P_k , creating a complete transmission chain where rank k determines G_k , and G_k determines P_k . This achieves efficient elite-driven convergence while maintaining population diversity.



(a) Analysis of the probability distribution of individual selection within the elite core



(b) Schematic analysis of the gravity gradient field

Figure 4 Probability distribution of the elite core and the gravity gradient field

3) Summary

In summary, the two-layer enhancement mechanism—comprising ranking-sensing elite immunity protection and gradient-driven selection—proposed in this paper mitigates the risk of unnecessary loss of high-quality individuals through the elite immunity mechanism, thereby fundamentally safeguarding the survival of the population's core evolutionary momentum. Through a two-stage gradient-guided selection model, it achieves adaptive dynamic regulation of selection pressure throughout the search process, enabling the algorithm to maintain broad exploration in the early stage and focus on the refined exploitation of the core elite group in the later stage. These two mechanisms are closely integrated in timing and are functionally complementary and reinforcing. Their shared core lies in the algorithm's real-time sensing of and intelligent response to individual rankings, which constitutes a foundation for the algorithm's convergence in the later stage, significantly enhancing the stability, convergence speed, and final accuracy of solutions to complex optimization problems.

3.5 Algorithm Pseudocode

Pseudocode of SSA-ABC is shown in Fig.5.

4 Experimental Results and Discussions

4.1 Experimental Design and Test Set

This paper uses the CEC2014 test set [32] and designs four experiments to validate and analyze the effectiveness and superiority of SSA-ABC. Experiment 1 is an ablation study that progressively verifies the effectiveness of the three proposed state-sensing mechanisms. These mechanisms include dimension-sensing hybrid initialization and reflection-hybrid mapping boundary handling, process-sensing elite-mean-guided dynamic dimension search, and ranking-sensing elite immunity protection and gradient-of-gravity selection models. Experiment 2 is a comparative analysis, in which SSA-ABC was compared with the standard ABC and advanced improved ABCs developed domestically and internationally in recent years, including APABC [13], ABCG [26], DFSABC_elite [19], ECABC [33], ARABC [34], and DAABC [35]. Experiment 3 is a parameter sensitivity analysis, examining the impact of the selection pressure factor m in the gradient selection model (within the two-stage selection model) on algorithm performance, and providing recommended parameter settings based on the experimental results. Experiment 4 is a convergence analysis, in which convergence curves were plotted to compare the convergence performance of SSA-ABC with that of other algorithms. In Experiments 1, 2, and 3, the mean and standard deviation of the optimal values obtained from multiple independent runs of each algorithm were calculated to evaluate solution accuracy and stability. The experimental platform used an Intel Core i5-8300H CPU with a clock speed of 2.30 GHz, running Windows 10. The programming language was C++, and the compiler was the Code Blocks GNU GCC Compiler.

```

Algorithm: State-Sensing Adaptive Artificial Bee Colony Algorithm (SSA-ABC)
Input: Problem dimension  $D$ ; Food Number  $SN$ ; Max evaluations  $Max\_FES$ ; Trial counter  $limit$ 
Output: Best solution  $x^*$ 

1 // ===== Mechanism I: Dimensionality-Sensing Hybrid Initialization =====
2 Calculate Latin hypercube initialization proportion by Eq.(9)
3 Initialize population  $P$  with hybrid Latin hypercube and random sampling based on Latin hypercube initialization proportion
4 Apply reflection-hybrid boundary processing to all individuals by Eq.(10)
5 Evaluate  $P$ , identify  $x^*$ , initialize  $trial$  counters
6  $FES = SN$ 
7 While  $FES < Max\_FES$  do
8 // ===== Mechanism II: State-Sensing Elite-Mean-Guided Dynamic Dimension Search =====
9   Calculate search dimension number  $dn$  by Eq.(11)
10  // Employed bee phase
11  For  $i = 1$  to  $SN$  do
12    If  $dn == 1$  then
13      Randomly select a dimension to improve
14      Generate  $v_i$  of  $x_i$  by Eq.(2)
15    Else
16      Identify top 5% and 10% elites
17      Calculate mean of elites by Eq.(12)
18      Select  $\lfloor dn/2 \rfloor$  dimensions to improve with largest distance from mean of elites by Eq.(13)
19      Select remaining dimensions randomly
20      Generate  $v_i$  of  $x_i$  by Eq.(14)
21    End If
22    Apply reflection-hybrid boundary processing to  $v_i$  by Eq.(10)
23    Greedy selection between  $v_i$  and  $x_i$ 
24     $FES = FES + SN$ 
25    Update  $x^*$  and  $trial_i$  if necessary
26  End For
27 // ===== Mechanism III: Ranking-Sensing Elite Immunity Protection and Gradient-Driven Selection =====
28  If  $FES/Max\_FES \leq 0.6$  then
29    Calculate selection probabilities of  $P$  by Eq.(4)
30  Else
31    Calculate size of elite nucleus by Eq.(15)
32    Calculate gravitational coefficient by Eq.(16)
33    Normalize to selection probabilities of  $P$  by Eq.(17)
34  End If
35  // Onlooker bee phase
36  For  $i = 1$  to  $SN$  do
37    Select  $x_i$  according to selection probabilities
38    If  $dn == 1$  then
39      Randomly select a dimension to improve
40      Generate  $v_i$  of  $x_i$  by Eq.(2)
41    Else
42      Identify top 5% and 10% elites
43      Calculate mean of elites by Eq.(12)
44      Select  $\lfloor dn/2 \rfloor$  dimensions to improve with largest distance from mean of elites by Eq.(13)
45      Select remaining dimensions randomly
46      Generate  $v_i$  of  $x_i$  by Eq.(14)
47    End If
48    Apply reflection-hybrid boundary processing to  $v_i$  by Eq.(10)
49    Greedy selection between  $v_i$  and  $x_i$ 
50     $FES = FES + SN$ 
51    Update  $x^*$  and  $trial_i$  if necessary
52  End For
53  Sort population by fitness
54  For each  $x_i$  of top 10% elites do
55     $trial_i = 0$ 
56  End For
57  // Scout bee phase
58  For  $i = 1$  to  $SN$  do
59    If  $trial_i > limit$  then
60      Reinitialize and evaluate  $x_i$  by Eq.(1)
61       $FES = FES + 1$ 
62    End If
63  End For
64 End While
65 Return  $x^*$ 

```

Figure 5 Pseudocode of SSA-ABC

The CEC2014 test set is a benchmark function set used for testing optimization algorithms, comprising a total of 30 functions. Among these, f1–f3 are unimodal functions designed to verify the algorithm's exploitation capabilities, while f4–f16 are simple multimodal functions intended to assess the algorithm's exploration capabilities and determine whether it can effectively escape local optima. Functions f17–f22 are hybrid functions designed to evaluate the algorithm's comprehensive optimization capabilities, and functions f23–f30 are composite functions that combine multiple test problems into a complex scenario, enabling a comprehensive and objective evaluation of the algorithm's performance. During the experiments, the difference between the optimal solution obtained by the algorithm and the optimal value of each function was calculated. Since the optimal value for all functions is 0, a difference closer to 0 is clearly better

4.2 Ablation Experiment Results and Analysis

To objectively analyze the contribution of each proposed strategy to the improvement in algorithm performance, the three state-sensing mechanisms were embedded into the standard ABC algorithm, resulting in the following algorithms for experimental analysis: ABC (standard ABC), ABC-I (standard ABC + state-sensing mechanism I), ABC-II (standard ABC + state-sensing mechanism II), ABC-III (standard ABC + state-sensing mechanism III), and SSA-ABC (standard ABC + state-sensing mechanisms I, II, and III). Each algorithm was independently run 30 times for $D = 50$, and the mean and standard deviation of the optimal values obtained were statistically analyzed to evaluate solution accuracy and stability. The common parameter settings were as follows: $Max_FEs = 10,000 \times D$, $SN = 50$, and $limit = SN \times D$.

Table II presents the experimental results for the standard ABC algorithm and the ABC algorithms incorporating various improvement strategies. The best results among all algorithms for each function are highlighted in bold, and the last row of the table shows the results of the Friedman nonparametric test.

In terms of the number of functions on which each algorithm achieved the best performance, SSA-ABC obtained the optimal mean and standard deviation on 17 functions, outperforming all other algorithms. ABC, ABC-I, ABC-II, and ABC-III achieved the best results on 3, 5, 4, and 5 functions, respectively. Compared with ABC, ABC-I, ABC-II, and ABC-III, SSA-ABC performed better on 21, 21, 22, and 24 functions, respectively.

These results demonstrate the effectiveness of the proposed improvement strategies, each of which contributes to performance enhancement. SSA-ABC, integrating all three strategies, exhibits the greatest performance improvement. The Friedman test further validates this conclusion. The mean rank order of the algorithms is: $ABC > ABC-II > ABC-I > ABC-III > SSA-ABC$, and the p -value of 0.000076 is significantly less than 0.05, indicating that the performance differences among the algorithms are statistically significant.

4.3 Comparative Experiment Results and Analysis

1) Comparative Algorithms and Parameter Settings

To further validate the superiority of the proposed algorithm, outstanding improved ABC algorithms from recent domestic and international research were selected for comparison. These include APABC with adaptive population size, which shares the concept of dynamic adjustment with SSA-ABC; ABCG based on the gravity

model, which aligns with the gradient-of-gravity selection mechanism; DFSABC_elite featuring elite-mean-guided search, which corresponds to the elite-mean-guided dimension search; ECABC designed for enhanced convergence, which serves as a baseline for convergence performance; as well as ARABC with ranking-based probabilistic selection, which relates to the ranking-sensing elite protection, and DAABC being dimension-adaptive, which matches the dimension-sensing hybrid initialization and dynamic dimension search. The parameter settings for the compared algorithms were adopted from their original literature, as shown in Table III. The common parameter settings were as follows: $D = 50$, and $Max_FEs = 10,000 \times D$. Each algorithm was independently run 30 times. The mean and standard deviation of the optimal values obtained were calculated to compare and analyze solution accuracy and stability. Furthermore, convergence curves were generated to compare and analyze convergence behavior.

2) Comparative Analysis of Solution Accuracy and Stability

Table IV presents the experimental comparison results of SSA-ABC against APABC, ABCG, ARABC, DFSABC_elite, DAABC, and ECABC. Nonparametric tests, including those for two-sample and multiple-sample comparisons, were conducted on the mean solution values of the algorithms using SPSS. The last three rows of the table show the results of the Wilcoxon signed-rank test for pairwise comparisons between SSA-ABC and the other algorithms, as well as the results of the Friedman nonparametric test for comparison among all algorithms.

In terms of the number of functions where each algorithm achieved the optimal mean and standard deviation, APABC, DFSABC_elite, ARABC, ABCG, DAABC, ECABC, and SSA-ABC ranked 1, 5, 0, 6, 2, 4, and 16, respectively, with SSA-ABC performing the best. Moreover, SSA-ABC outperformed APABC, DFSABC_elite, ARABC, ABCG, DAABC, and ECABC on 21, 19, 19, 21, 23, and 21 functions, respectively. Overall, SSA-ABC demonstrates superior solution accuracy and stability compared to the other improved algorithms.

The Wilcoxon signed-rank test results indicate that the p -values for all pairwise comparisons between SSA-ABC and the other six algorithms are below 0.05, suggesting that SSA-ABC is significantly superior to each compared algorithm. The Friedman test results show the following rank order: ARABC > ABCG > APABC > DAABC > DFSABC_elite > ECABC > SSA-ABC. SSA-ABC achieves the lowest mean rank among the seven algorithms, indicating the best overall performance. Furthermore, the p -value=0.006963 of the Friedman test is far below 0.05, confirming that the performance differences among the algorithms are statistically significant.

3) Comparative Analysis of Convergence

To further verify the convergence of SSA-ABC, two functions with identical or similar solution accuracy and stability were selected from each of the four function types in CEC2014. Their convergence curves were plotted in Fig.6, where the horizontal axis represents the number of function evaluations and the vertical axis represents the logarithm (base 10) of the function values. The common parameter settings are as follows: $D=50$, $Max_FEs = 10,000 \times D$.

TABLE II Ablation experiment results

Func	ABC	ABC-I	ABC-II	ABC-III	SSA-ABC
f1	1.12E+07	9.20E+06	8.17E+06	7.26E+06	6.65E+06
	3.35E+06	2.93E+06	2.10E+06	1.64E+06	1.51E+06
f2	7.53E+02	8.44E+02	1.51E+03	9.49E+02	1.08E+03
	5.65E+02	7.25E+02	1.80E+03	5.77E+02	1.49E+03
f3	9.66E+03	6.90E+03	9.31E+03	9.08E+03	7.37E+03
	3.18E+03	2.30E+03	2.83E+03	3.38E+03	2.25E+03
f4	6.07E+01	6.21E+01	5.67E+01	5.19E+01	5.10E+01
	2.44E+01	2.53E+01	2.90E+01	3.24E+01	3.02E+01
f5	2.02E+01	2.01E+01	2.00E+01	2.00E+01	2.00E+01
	2.18E-02	1.65E-02	1.50E-02	1.31E-02	8.17E-03
f6	3.07E+01	3.04E+01	2.97E+01	2.95E+01	2.94E+01
	2.19E+00	2.03E+00	2.39E+00	2.04E+00	2.30E+00
f7	1.44E-03	1.40E-03	4.23E-03	1.79E-05	4.47E-03
	9.80E-04	1.18E-03	7.92E-03	1.57E-05	7.59E-03
8	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f9	1.89E+02	1.91E+02	2.06E+02	1.85E+02	2.19E+02
	2.20E+01	1.92E+01	2.81E+01	1.83E+01	2.89E+01
f10	5.26E-01	6.69E-01	2.70E-01	3.02E-01	1.19E-01
	4.65E-01	6.67E-01	2.93E-01	2.69E-01	1.33E-01
f11	4.25E+03	4.11E+03	4.30E+03	4.20E+03	4.10E+03
	3.35E+02	3.95E+02	3.81E+02	3.37E+02	2.82E+02
f12	1.62E-01	1.58E-01	1.10E-01	1.01E-01	9.67E-02
	2.21E-02	1.57E-02	1.98E-02	2.10E-02	1.39E-02
f13	3.03E-01	3.31E-01	4.01E-01	3.05E-01	3.94E-01
	4.08E-02	2.58E-02	4.68E-02	3.91E-02	5.03E-02
f14	2.56E-01	2.45E-01	2.31E-01	2.59E-01	2.25E-01
	2.08E-02	2.22E-02	1.59E-02	2.60E-02	1.58E-02
f15	1.95E+01	1.78E+01	1.80E+01	1.63E+01	1.91E+01
	1.89E+00	2.29E+00	3.04E+00	2.09E+00	3.01E+00
f16	1.82E+01	1.79E+01	1.76E+01	1.79E+01	1.76E+01
	3.93E-01	3.84E-01	6.29E-01	5.70E-01	5.25E-01
f17	3.00E+06	2.22E+06	2.31E+06	2.07E+06	1.46E+06
	1.12E+06	8.85E+05	1.24E+06	8.85E+05	6.44E+05
f18	1.09E+03	6.73E+02	1.46E+03	1.32E+03	1.05E+03
	7.52E+02	3.85E+02	6.39E+02	1.15E+03	8.37E+02
f19	1.56E+01	1.47E+01	1.60E+01	1.56E+01	1.48E+01
	1.48E+00	2.03E+00	1.95E+00	1.84E+00	1.85E+00
f20	3.11E+04	3.05E+04	2.54E+04	3.19E+04	2.87E+04
	1.17E+04	9.08E+03	7.68E+03	8.10E+03	7.30E+03
f21	1.63E+06	1.28E+06	1.52E+06	1.35E+06	1.22E+06
	7.44E+05	6.11E+05	5.02E+05	7.94E+05	6.30E+05
f22	7.67E+02	7.35E+02	7.63E+02	7.77E+02	7.53E+02
	1.59E+02	1.57E+02	1.76E+02	1.39E+02	1.58E+02
f23	3.27E+02	3.27E+02	3.27E+02	3.27E+02	3.27E+02
	1.43E-02	8.09E-03	2.70E-03	6.42E-03	4.04E-03
f24	2.05E+02	2.05E+02	2.05E+02	2.04E+02	2.04E+02
	2.64E-01	2.82E-01	2.81E-01	2.45E-01	3.46E-01
f25	2.16E+02	2.15E+02	2.13E+02	2.13E+02	2.12E+02
	2.15E+00	1.80E+00	1.87E+00	2.46E+00	1.18E+00
f26	1.00E+02	1.00E+02	1.00E+02	1.00E+02	1.00E+02
	3.44E-02	3.62E-02	3.98E-02	2.58E-02	1.70E-02
f27	4.33E+02	4.28E+02	4.24E+02	4.26E+02	4.24E+02
	4.07E+00	3.51E+00	3.63E+00	3.29E+00	2.58E+00
f28	5.66E+02	5.85E+02	5.60E+02	5.62E+02	5.86E+02
	2.26E+01	2.31E+01	2.58E+01	2.24E+01	2.99E+01
f29	2.58E+03	2.41E+03	2.47E+03	2.46E+03	2.35E+03
	9.57E+02	5.99E+02	9.54E+02	7.68E+02	6.81E+02
f30	1.06E+04	9.18E+03	8.88E+03	8.50E+03	8.16E+03
	1.68E+03	1.38E+03	1.47E+03	1.36E+03	1.23E+03
rank	3.95	2.98	3.22	2.70	2.15

As shown in Fig.6, SSA-ABC performs well on the CEC2014 test functions. On f2, despite a relatively high initial value, SSA-ABC converges rapidly, and its final convergence accuracy ranks among the best of the compared algorithms. On f3 and f8, SSA-ABC exhibits good convergence rates and final accuracy, with particularly leading performance on f8. On high-dimensional complex functions such as f19 and f22, SSA-ABC maintains a stable convergence trend, achieving final accuracy comparable to or slightly better than other algorithms. On functions including f5, f23, and f26, SSA-ABC also attains satisfactory convergence results. Overall, SSA-ABC demonstrates stable performance in both convergence speed and final accuracy. This is attributed to its synergistic mechanism combining dimension-adaptive initialization, process-adaptive dimensional search, and population-adaptive selection, which effectively balances exploration and exploitation, thereby achieving efficient convergence and high-precision optimization.

TABLE III Algorithm parameter settings

Algorithm	Parameter setting
APABC	$SN=35, limit=SN*D, T=20, A=\emptyset, SN_{max}=35, SN_{min}=20$
DFSABC_elite	$SN=50, limit=SN*D, p=0.1, r=1/p$
ARABC	$SN=50, limit=SN*D, \Delta=0.01, \alpha_{min}=0, \alpha_{max}=5$
ABCG	$SN=50, limit=SN*D$
DAABC	$SN=150, limit=SN*D$
ECABC	$SN=50, limit=SN*D, p=0.1$
SSA-ABC	$SN=50, limit=SN*D, m=6$

4) Summary

SSA-ABC was comprehensively evaluated using the CEC2014 test function set. The algorithm was compared against six mainstream improved ABC algorithms in terms of mean, standard deviation, and nonparametric test results. SSA-ABC achieved the optimal mean and standard deviation on 16 out of 30 test functions, significantly outperforming the other algorithms. The Wilcoxon rank-sum test revealed significant differences between SSA-ABC and each compared algorithm. The Friedman test further confirmed that SSA-ABC ranked first, with a p-value far below 0.05, indicating that its performance advantage is statistically significant. Convergence analysis showed that SSA-ABC exhibited more persistent and stable convergence capabilities on multimodal, hybrid, and composite functions. In summary, SSA-ABC demonstrates significant advantages in solution accuracy, stability, and convergence.

4.4 Parameter Sensitivity Experiments and Results Analysis

To conduct a sensitivity analysis of parameter m , 11 representative benchmark functions were selected from the CEC2014 test suite, covering four function types: unimodal, multimodal, hybrid, and composite. Specifically, from the unimodal functions, f1 was selected to evaluate high-precision convergence, f2 for local exploitation capability, and f3 for adaptability to dimensional heterogeneity. From the multimodal functions, f5 was used to test global exploration capability, f7 to assess the ability to escape local optima, and f9 to evaluate diversity maintenance under complex landscapes. From the hybrid functions, f12 and f14 were selected to comprehensively evaluate robustness and adaptability in handling heterogeneous optimization problems. From the composite functions, f20, f22, and f25 were selected to test generalization ability and ultimate convergence performance under highly complex scenarios. These

11 functions enable a multi-dimensional evaluation of the sensitivity of parameter m in terms of convergence accuracy, exploration capability, robustness, and generalization capability.

TABLE IV Comparative Experimental Results

Func	APABC	DFSABC elite	ARABC	ABCG	DAABC	ECABC	SSA-ABC
f1	1.51E+07 2.82E+06	8.67E+06 2.90E+06	1.87E+07 5.54E+06	2.32E+07 5.75E+06	1.33E+07 3.71E+06	4.64E+06 1.14E+06	6.65E+06 1.51E+06
f2	1.12E+04 5.22E+03	4.80E+03 5.07E+03	2.65E+03 2.25E+03	1.51E+03 1.31E+03	3.48E+03 3.26E+03	3.23E+03 2.84E+03	1.08E+03 1.49E+03
f3	8.79E+03 3.88E+03	9.98E+03 2.23E+03	1.02E+04 3.16E+03	2.65E+03 1.25E+03	9.43E+03 7.34E+03	5.30E+03 1.77E+03	7.37E+03 2.25E+03
f4	7.31E+01 2.07E+01	7.60E+01 2.71E+01	8.83E+01 7.45E+00	9.74E+01 1.75E+01	6.40E+01 3.26E+01	1.05E+02 2.04E+01	5.10E+01 3.02E+01
f5	2.02E+01 3.42E-02	2.02E+01 3.09E-02	2.03E+01 1.91E-02	2.04E+01 2.10E-02	2.03E+01 2.33E-02	2.04E+01 3.76E-02	2.00E+01 8.17E-03
f6	2.75E+01 3.66E+00	2.61E+01 1.14E+00	2.67E+01 1.23E+00	3.72E+01 1.56E+00	2.81E+01 1.64E+00	2.99E+01 2.17E+00	2.94E+01 2.30E+00
f7	2.26E-03 2.89E-03	2.38E-03 3.93E-03	7.66E-04 2.01E-03	1.24E-08 2.73E-08	2.49E-03 2.35E-03	4.57E-03 5.60E-03	4.47E-03 7.59E-03
f8	0.00E+00 0.00E+00	0.00E+00 0.00E+00	8.37E-09 1.49E-09	0.00E+00 0.00E+00	0.00E+00 0.00E+00	6.96E-01 4.56E-01	0.00E+00 0.00E+00
f9	1.16E+02 1.56E+01	9.22E+01 1.20E+01	1.01E+02 1.14E+01	1.71E+02 2.51E+01	9.50E+01 1.01E+01	6.42E+01 9.97E+00	2.19E+02 2.89E+01
f10	1.61E-01 2.08E-01	3.15E+00 8.67E-01	4.19E-01 4.38E-01	1.08E+01 4.03E+00	4.86E-01 4.42E-01	7.55E+00 2.40E+00	1.19E-01 1.33E-01
f11	3.73E+03 4.60E+02	3.49E+03 3.12E+02	3.96E+03 2.86E+02	6.59E+03 3.43E+02	3.81E+03 2.99E+02	4.46E+03 3.48E+02	4.10E+03 2.82E+02
f12	1.63E-01 2.03E-02	1.72E-01 3.05E-02	2.66E-01 2.16E-02	5.63E-01 3.76E-02	2.68E-01 1.91E-02	4.91E-01 6.21E-02	9.67E-02 1.39E-02
f13	3.46E-01 5.05E-02	2.66E-01 1.37E-02	3.34E-01 3.11E-02	4.31E-01 3.70E-02	2.44E-01 3.52E-02	3.78E-01 4.22E-02	3.94E-01 5.03E-02
f14	3.16E-01 2.27E-02	2.79E-01 2.17E-02	2.93E-01 3.55E-02	3.25E-01 1.99E-02	2.68E-01 2.26E-02	2.84E-01 2.06E-02	2.25E-01 1.58E-02
f15	1.46E+01 1.73E+00	9.33E+00 1.23E+00	1.32E+01 1.64E+00	1.83E+01 1.23E+00	2.03E+01 4.68E+00	2.05E+01 9.64E-01	1.91E+01 3.01E+00
f16	1.73E+01 4.45E-01	1.68E+01 2.76E-01	1.74E+01 3.91E-01	1.88E+01 4.08E-01	1.73E+01 3.55E-01	1.74E+01 3.43E-01	1.76E+01 5.25E-01
f17	4.54E+06 1.42E+06	3.93E+06 1.43E+06	6.84E+06 1.91E+06	2.06E+06 7.11E+05	3.42E+06 1.02E+06	2.40E+06 5.23E+05	1.46E+06 6.44E+05
f18	1.98E+03 5.87E+02	1.31E+03 8.03E+02	8.33E+03 4.11E+03	1.25E+03 6.49E+02	7.94E+06 2.38E+07	1.94E+03 4.94E+02	1.05E+03 8.37E+02
f19	2.00E+01 7.73E+00	1.71E+01 1.85E+00	2.36E+01 5.12E+00	4.25E+01 1.43E+01	1.98E+01 6.43E+00	2.75E+01 1.04E+01	1.48E+01 1.85E+00
f20	3.49E+04 1.27E+04	2.92E+04 6.74E+03	2.75E+04 7.15E+03	9.54E+03 2.89E+03	2.77E+04 8.95E+03	3.18E+04 2.99E+03	2.87E+04 7.30E+03
f21	4.15E+06 1.40E+06	2.45E+06 8.24E+05	3.33E+06 1.17E+06	1.61E+06 8.85E+05	2.57E+06 8.89E+05	1.60E+06 6.09E+05	1.22E+06 6.30E+05
f22	7.53E+02 2.06E+02	5.96E+02 1.24E+02	7.92E+02 2.12E+02	4.99E+02 1.41E+02	9.82E+02 1.52E+02	5.63E+02 1.70E+02	7.53E+02 1.58E+02
f23	3.27E+02 6.58E-02	3.27E+02 6.25E-03	3.27E+02 7.83E-02	3.27E+02 2.22E-10	3.27E+02 5.42E-02	3.27E+02 5.20E-04	3.27E+02 4.04E-03
f24	2.04E+02 2.86E-01	2.04E+02 2.17E-01	2.04E+02 2.55E-01	2.03E+02 3.27E-01	2.04E+02 3.97E-01	2.03E+02 2.60E-01	2.04E+02 3.46E-01
f25	2.17E+02 2.36E+00	2.16E+02 2.98E+00	2.20E+02 2.37E+00	2.16E+02 2.19E+00	2.18E+02 2.36E+00	2.12E+02 1.26E+00	2.12E+02 1.18E+00
f26	1.00E+02 3.18E-02	1.00E+02 2.69E-02	1.00E+02 3.09E-02	1.00E+02 3.44E-02	1.00E+02 2.09E-02	1.00E+02 2.59E-02	1.00E+02 1.70E-02
f27	4.39E+02 4.49E+00	4.29E+02 4.70E+00	4.45E+02 6.81E+00	4.60E+02 6.66E+00	4.39E+02 6.48E+00	4.34E+02 2.68E+00	4.24E+02 2.58E+00
f28	5.66E+02 1.39E+01	5.63E+02 1.73E+01	5.72E+02 2.07E+01	6.35E+02 2.45E+01	5.72E+02 2.22E+01	5.36E+02 1.80E+01	5.86E+02 2.99E+01
f29	3.57E+03 9.80E+02	3.56E+03 6.46E+02	4.26E+03 7.64E+02	6.27E+03 1.70E+03	7.36E+03 3.50E+03	4.08E+03 9.22E+02	2.35E+03 6.81E+02
f30	1.42E+04 2.16E+03	9.74E+03 1.63E+03	1.27E+04 3.48E+03	9.42E+03 1.11E+03	1.12E+04 3.39E+03	8.85E+03 1.60E+03	8.16E+03 1.23E+03
R+ R- p	253 72 0.014889	256 95 0.040901	281 97 0.027079	284 94 0.022467	269.5 81.5 0.016963	271 107 0.048833	-
rank	4.20	3.20	4.62	4.58	4.27	4.13	3.00

In the experiments, the dimensionality was set to $D=50$, and five gradient values of m ($m = 2, 4, 6, 8, 10$) were tested. Each configuration was run independently 30 times, and the mean and standard deviation of the convergence accuracy were calculated. The results are presented in Table V.

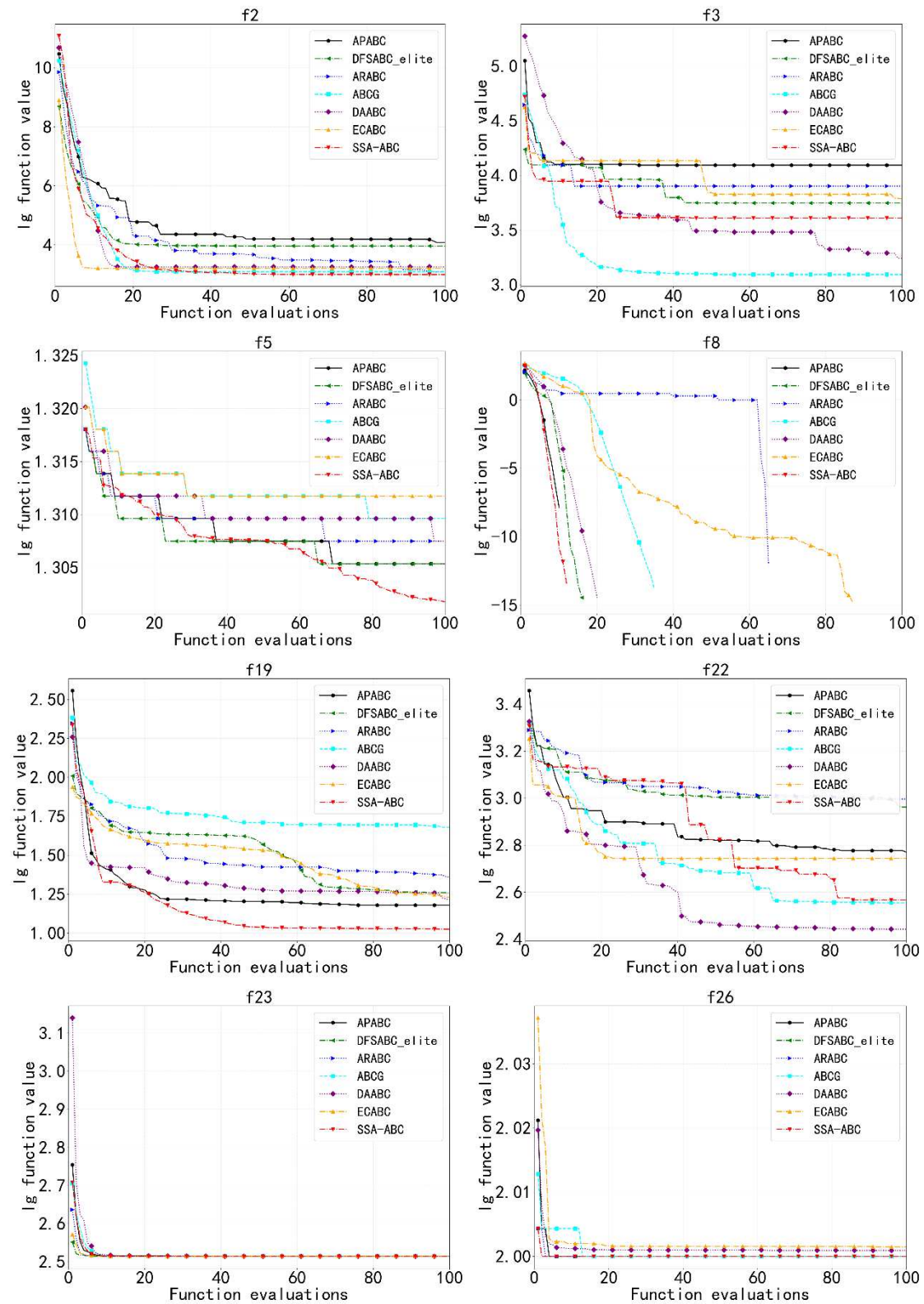


Figure 6 Convergence curves of the comparison algorithms on functions with similar performance

TABLE V Parameter sensitivity experimental results

Func	$m=2$	$m=4$	$m=6$	$m=8$	$m=10$
f1	7.67e+006	7.20e+006	6.65E+06	7.57e+006	6.79e+006
	1.74e+006	2.12e+006	1.51E+06	2.34e+006	1.70e+006
f2	9.61e+002	1.22e+003	1.08E+03	8.60e+002	8.52e+002
	1.24e+003	1.33e+003	1.49E+03	1.45e+003	6.77e+002
f3	8.44e+003	8.99e+003	7.37E+03	8.93e+003	8.91e+003
	2.81e+003	3.04e+003	2.25E+03	3.83e+003	2.78e+003
f5	2.00e+001	2.00e+001	2.00E+01	2.00e+001	2.00e+001
	9.96e-003	1.53e-002	8.17E-03	1.11e-002	1.23e-002
f7	2.09e-005	8.53e-004	4.47E-03	2.69e-005	8.50e-004
	1.49e-005	2.52e-003	7.59E-03	3.76e-005	3.34e-003
f9	1.87e+002	1.90e+002	2.19E+02	1.88e+002	1.87e+002
	2.40e+001	2.35e+001	2.89E+01	1.84e+001	1.89e+001
f12	1.02e-001	1.01e-001	9.67E-02	1.03e-001	1.05e-001
	2.06e-002	2.36e-002	1.39E-02	2.24e-002	1.87e-002
f14	2.55e-001	2.56e-001	2.25E-01	2.62e-001	2.49e-001
	2.21e-002	2.61e-002	1.58E-02	2.24e-002	2.42e-002
f20	3.06e+004	3.12e+004	2.87E+04	2.98e+004	3.37e+004
	8.85e+003	7.76e+003	7.30E+03	6.71e+003	8.32e+003
f22	7.86e+002	8.05e+002	7.53E+02	7.81e+002	7.60e+002
	1.63e+002	1.43e+002	1.58E+02	1.48e+002	1.76e+002
f25	2.13e+002	2.13e+002	2.12E+02	2.13e+002	2.13e+002
	2.16e+000	1.96e+000	1.26E+00	1.94e+000	2.50e+000
rank	4.20	3.20	4.62	4.58	4.27

As shown in Table V, the performance of different m values fluctuates across functions without exhibiting a monotonic trend. The Friedman test was employed to assess whether the effect of m on overall performance is statistically significant. The resulting $p_value=0.102193>0.05$ indicates no significant difference among the tested m values. Overall, the algorithm is insensitive to the parameter m within the range [2,10], demonstrating good parameter robustness. Considering that $m=6$ achieves a high rank on most functions and yields the optimal mean in several cases, it is recommended as the default parameter configuration.

4.5 Experimental Results and Analysis of the Robot Path Planning Problem

1) Design of the SSA-ABC Algorithm for Solving the Robot Path Planning Problem

(1) Solution Encoding

In the robot path planning problem addressed in this paper, a single robot path constitutes a solution. A path consists of N nodes, each defined by its x and y coordinates. Accordingly, the solution dimension is $D=2N$. Each solution is represented as a D -dimensional vector, where each pair of consecutive dimensions corresponds to the coordinates of a single path node, as shown in (18):

$$P = \{x_1, y_1, \dots, x_N, y_N\} \quad (18)$$

The starting point (x_1, y_1) and end point (x_N, y_N) of the path are fixed and are not treated as optimization variables. Therefore, path optimization is primarily achieved by adjusting the coordinates of the intermediate nodes $(x_2, y_2), (x_3, y_3), \dots, (x_{N-1}, y_{N-1})$ during the optimization process.

(2) Population Initialization

Several candidate paths from the starting point to the destination are randomly generated as initial solutions. In this paper, each path node in every initial solution is generated using vertical offsets, ensuring that the resulting path allows the robot to move continuously from the start to the destination without colliding with obstacles. The specific initialization procedure is as follows: First, connect the robot's starting point S and destination point E to form the line segment SE , which represents the base path direction. Next, divide SE into $N-1$ equal segments, yielding $N-2$ intermediate division points. At each division point P_i , a virtual line direction vector perpendicular to SE is constructed. An illustration of the virtual line direction vector at a division point is shown in Fig.7.

In Fig.7, the blue point in the lower-left corner is the starting point S , and the red point in the upper-right corner is the endpoint E . The red solid line represents the line segment SE connecting the two points. The green points on SE are the $N-2$ intermediate points that evenly divide SE . The blue dashed line indicates the direction vector perpendicular to SE . At each segment point, a random offset in the vertical direction (i.e., the direction of the dashed line) is added to the x and y coordinates of the segment point, while ensuring that the resulting coordinates remain centered within the grid. This ensures that the path nodes lie near the perpendicular line of the segment point. Finally, connecting the segment points obtained after applying the vertical offset yields an initial path P for the robot.

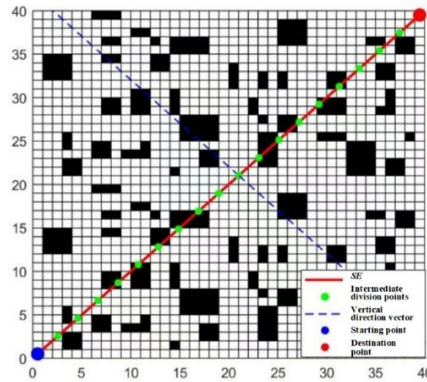


Figure 7 Vertical vector diagram

It is worth noting that if the random offset is set too large (e.g., close to the map width), the generated path will cover an excessively wide search space, which may degrade the solution quality. By restricting the random offset to a small range, a certain degree of randomness can be maintained while preventing the path from deviating excessively from the direction of the SE . In this paper, to optimize the efficiency of pathfinding, the maximum random offset, denoted as max_offset , is set to one-fifth of the grid map's dimension. Random offsets are generated within the range $[-max_offset, max_offset]$. If no feasible path node is found within this range, the random offset is gradually increased until a feasible node is generated.

(3) SSA-ABC Iterative Optimization and Smoothing

For the initialized population, the SSA-ABC algorithm proposed in this paper is used for iterative optimization until the maximum number of iterations, G_{max} , is reached, at which point the optimal solution found is output. It is worth noting that the smoothness of the robot's planned path is a key consideration in path optimization. While ensuring the path's continuity and feasibility, excessive twists and turns should be avoided, as this is crucial for reducing execution time and improving the robot's motion efficiency.

Therefore, a path smoothing function f_{smooth} is designed to optimize the shape of the robot's path, making it as smooth as possible.

Specifically, it determines whether to process intermediate nodes by checking whether two adjacent nodes on the path can be directly connected (i.e., there are no obstacles in the intermediate region), thereby making the path more intuitive and efficient. The specific formulation of f_{smooth} is as follows: For a feasible path P , where let path node i be represented as P_i . For any three adjacent points P_{i-1} , P_i , and P_{i+1} on the path, checks whether the intermediate point P_i can be simplified. The calculation is performed as follows: Perform obstacle detection on the line segment connecting path nodes P_{i-1} and P_{i+1} . If the straight-line path between P_{i-1} and P_{i+1} does not intersect any obstacles, it is considered that P_{i-1} and P_{i+1} are directly connected. In this case, P_i is redefined as the midpoint of the line connecting P_{i-1} and P_{i+1} , thereby simplifying the path.

Furthermore, to compare the smoothness of paths obtained by various algorithms, this paper employs an inflection point calculation function to analyze corners or inflection points in the path. For any three consecutive nodes P_{i-1} , P_i , and P_{i+1} in the path, the following operations are performed

- ① Calculate the two vectors v_1 and v_2 according to (19) and (20):

$$v_1 = P_i - P_{i-1} = (x_i - x_{i-1}, y_i - y_{i-1}) \quad (19)$$

$$v_2 = P_{i+1} - P_i = (x_{i+1} - x_i, y_{i+1} - y_i) \quad (20)$$

- ② Calculate the angle θ_i between the two vectors v_1 and v_2 using (21):

$$\cos(\theta_i) = \left[\frac{v_1 \cdot v_2}{\|v_1\| \cdot \|v_2\|} \right] \quad (21)$$

where $v_1 \cdot v_2$ is the dot product of the two vectors, and $\|v_1\|$ and $\|v_2\|$ are the magnitudes of the vectors.

- ③ Calculate the angle θ_i using the inverse cosine function according to (22):

$$\theta_i = \cos^{-1} \left[\frac{v_1 \cdot v_2}{\|v_1\| \cdot \|v_2\|} \right] \quad (22)$$

- ④ Convert the angle to degrees using (23):

$$\theta_i = \theta_i \times \frac{180^\circ}{\pi} \quad (23)$$

If θ_i exceeds a given angle threshold, the path is considered to have an inflection point at node P_i .

2) Grid-Based Environmental Modeling

In robot path planning, grid maps serve as a fundamental method for environmental modeling. By discretizing a continuous physical space into regular grid cells, environmental information is simplified and represented. The principle can be briefly described as follows: Assuming the robot's working environment is a horizontal two-dimensional plane, this plane is divided into rectangle cells. The working environment is then abstracted into a two-dimensional binary matrix, where grid cells without obstacles are marked as 0 and those filled with obstacles are marked as 1, thereby mapping the real physical environment into a digital matrix containing only 0s and 1s. The grid cells containing obstacles are marked in black to indicate impassable areas, while the free areas are marked in white to indicate passable areas, forming the final grid map used for robot path planning.

Following the grid map generation method described in Reference [36] of this paper, a 40×40 grid map $G(i,j)$ is constructed. Here, $G(i,j)$ represents the grid cell at row i and column j . If $G(i,j) = 0$, the cell represents free space (passable); if $G(i,j) = 1$, the cell

represents an obstacle (impassable). The matrix of the grid map G can be expressed as shown in (24):

$$G = \begin{pmatrix} G(1,1) & \cdots & G(1,j) \\ \vdots & \ddots & \vdots \\ G(i,1) & \cdots & G(i,j) \end{pmatrix} \quad (24)$$

Additionally, to accurately plot the grid map and the path, the grid indices are converted to actual two-dimensional. The coordinates of the center point of each grid cell are used to represent the positions of obstacle grids, free space, and path nodes. Assume that the side length of each grid cell is 1 unit. The conversion method is shown in (25):

$$\begin{cases} x = i - 0.5 \\ y = j - 0.5 \end{cases} \quad (25)$$

where i is the row index and j is the column index.

3) Simulation Environment and Parameter Settings

This simulation study used MATLAB R2025a to investigate grid preprocessing methods. The simulation experiments were conducted on an Intel Core i5-8300H CPU with a base clock speed of 2.30 GHz, running Windows 10.

First, an environmental model diagram, as shown in Fig.8, was established in MATLAB. The experimental grid size was 40×40 . Based on the specific environmental model, the parameters for SSA-ABC were set as follows: $SN = 50$; $D = 40$; $N = D/2$; $limit = 30$; $m = 6$; $G_{max} = 500$. On the grid map, the starting point S was located at the center of the grid with coordinates $(0.5, 0.5)$, and the destination point E was located at the center of the grid with coordinates $(39.5, 39.5)$. This paper assumed that each sub-grid was 1×1 ; therefore, Q was the diagonal length of the grid, i.e., $\sqrt{2}$. The corner threshold was set to 45° [37].

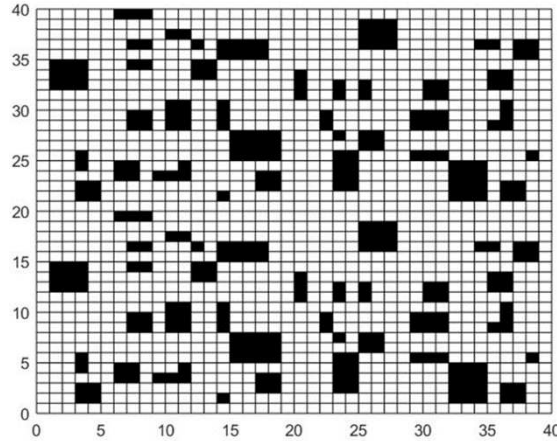


Figure 8 Grid map model

To minimize randomness, the experiment compared the average path length, optimal path length, worst path length, average runtime, and standard deviation obtained from multiple solutions. Among these, path length reflected the accuracy of the solution: a shorter optimal path length indicated that the algorithm had found a better solution during the robot path planning process. Standard deviation reflected the degree of variation among solutions: a smaller standard deviation indicated that the path lengths were more consistent across multiple experiments, suggesting a more stable algorithm. Average runtime reflected the computational resources and time required by the algorithm during path planning: the shorter the time taken to obtain a path, the higher the solution efficiency.

In addition, the optimal path map was plotted, and the number of inflection points in the optimal path was calculated. The optimal path diagram illustrated the optimal path obtained by the algorithm, verifying the path's quality and rationality while visually presenting the path planning results. The number of inflection points reflected the smoothness of the solution path: the fewer the inflection points, the smoother the path.

4) Comparative Analysis with ABC and Its Improved Algorithms

To further validate the superiority of SSA-ABC for complex problems such as robot path planning, simulation experiments were conducted using SSA-ABC alongside the standard ABC algorithm and six classical improved ABC algorithms, as discussed in the experimental validation section. In the experiments, the parameter settings for the six classical improved algorithms were as described earlier. All algorithms were run independently for 20 times. The experimental results are shown in Table VI.

TABLE VI Experimental results of SSA-ABC, standard ABC, and 6 classic improved algorithms

Metrics	SSA-ABC	ABC	APABC	DFSABC elite	ARABC	ABCG	DAABC	ECABC
Best Path	57.475	60.867	59.222	57.668	58.002	58.223	60.017	59.468
Worst Path	63.348	68.823	69.813	73.954	63.841	64.645	79.502	69.188
Average Path	60.222	62.866	62.304	62.120	60.740	60.619	67.671	62.505
Standard Deviation	1.765	1.824	1.655	2.733	1.960	1.6122	3.463	1.615
Number of Inflection Points	1	5	2	2	2	2	5	2
Average Runtime (s)	4.649	2.350	3.968	5.080	10.725	9.878	8.310	9.011

As shown in Table VI, SSA-ABC achieves superior performance across all evaluation metrics compared to the seven other algorithms, with an average path length of 60.2215, optimal path length of 57.4754, worst path length of 63.3478, only one inflection point, and a standard deviation of 1.7654. The optimal path length of SSA-ABC was 57.4754, indicating a better ability to find the optimal solution than the basic ABC algorithm at 60.8665 and the other improved algorithms. SSA-ABC also produced the shortest worst path length and the smallest standard deviation, demonstrating minimal solution variability and high stability across multiple runs. In contrast, other algorithms such as DFSABC and DAABC exhibited significantly larger variability, with worst path lengths of 73.9544 and 79.5018 and standard deviations of 2.7334 and 3.4630, respectively. The average runtime of SSA-ABC was 4.6491 seconds, which is slightly higher than that of Standard ABC at 2.3503 seconds and APABC at 3.9677 seconds but remains comparable. Other algorithms, including ARABC, ECABC, DAABC, and ABCG, required longer execution times, indicating that SSA-ABC is more computationally efficient. In summary, SSA-ABC outperforms the standard ABC algorithm and its classical improved variants in solving robot path planning problems. The optimal paths obtained by the eight algorithms are shown in Fig.9.

As shown in Fig.9 and based on the calculated number of inflection points, the SSA-ABC, Standard ABC, APABC, DFSABC_elite, ARABC, ABCG, DAABC, and ECABC algorithms generated paths with 1, 5, 2, 2, 2, 2, 5, and 2 inflection points, respectively.

SSA-ABC demonstrates superior path performance compared to the other algorithms, generating paths that are short and smooth while exhibiting strong optimization and local adjustment capabilities. The vertical offset strategy employed by SSA-ABC effectively avoids excessive detours or winding paths.

Overall, SSA-ABC offers a balanced trade-off between path quality and computational efficiency, as well as superior search stability. Compared to the standard ABC algorithm and its six classical improved variants, SSA-ABC has significant advantages in path optimization and stability, demonstrating its efficiency and superiority in complex robot path planning scenarios.

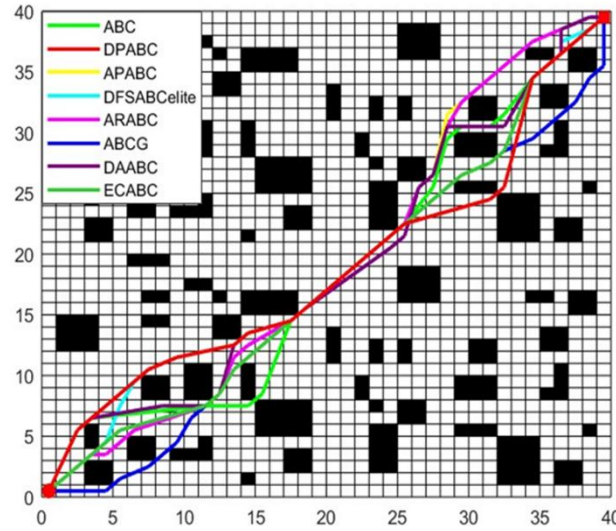


Figure 9 Path planning results of SSA-ABC, standard ABC, and improved ABCs

A comparison with other algorithms further validates the superiority of SSA-ABC. Using the experimental environment and parameters set in the previous section, path planning was performed by the Genetic Algorithm (GA) [38], Sparrow Search Algorithm (SSA) [39], Particle Swarm Optimization (PSO) [4], Grey Wolf Optimizer (GWO) [40], Differential Evolution (DE) [41], and SSA-ABC. The common parameter SN was set to 50 with 500 iterations; all other parameters followed the original algorithm specifications. By analyzing the results of 20 runs, the path planning performance of different algorithms was compared. The specific experimental results are shown in Table VII.

As shown in Table VII, SSA-ABC outperforms the other five swarm intelligence algorithms across all evaluation metrics, achieving an average path length of 60.643, an optimal path length of 57.3876, a worst path length of 64.2549, and a standard deviation of 1.1658. Its optimal path length is significantly better than that of GA at 59.1015, SSA at 62.5273, PSO at 58.0391, DE at 59.9979, and GWO at 58.5196. In terms of worst path length and standard deviation, SSA-ABC also far surpasses GA with 78.1436 and 3.8778 respectively and PSO with 5.6355, demonstrating superior stability across multiple runs. Furthermore, the number of inflection points in the optimal path generated by SSA-ABC is either better than or equal to those of the other five algorithms, indicating better path smoothness. The average runtime of SSA-ABC is 4.3365 seconds, which is significantly better than that of the other compared algorithms, meeting the efficiency requirements for path planning. All six algorithms were optimized under identical initial conditions, and their optimal paths are shown in Fig.10.

TABLE VII Experimental results of SSA-ABC and other swarm intelligence optimization algorithms

Metrics	SSA-ABC	GA	SSA	PSO	DE	GWO	SSA-ABC	GA
Best Path	57.388	59.102	62.527	58.039	59.998	58.520	57.388	59.102
Worst Path	64.255	78.144	76.828	65.540	66.300	67.846	64.255	78.144
Average Path	60.643	64.264	68.460	64.328	62.779	63.961	60.643	64.264
Standard Deviation	1.166	3.878	1.903	5.636	4.012	4.6419	1.166	3.878
Number of Inflection Points	2	3	2	2	4	2	2	3
Average Runtime (s)	4.337	16.994	11.998	18.723	18.099	11.335	4.337	16.994

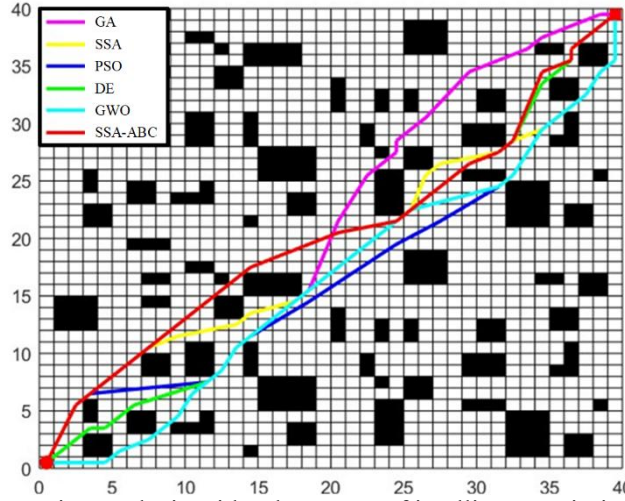


Figure 10 Comparative analysis with other types of intelligent optimization algorithms

Based on the path diagrams and inflection point counts calculated using the inflection-point calculation function, SSA-ABC, GA, SSA, PSO, GWO, and DE algorithms yielded 2, 3, 2, 2, 4, and 2 inflection points, respectively. These results indicate that SSA-ABC achieves better path smoothness and lower tortuosity compared to GA and GWO, while performing on par with SSA, PSO, and DE.

Overall, SSA-ABC demonstrated a strong convergence rate and robust path optimization performance throughout the iterative process. Its optimal path was significantly better than those of the other five algorithms, suggesting that SSA-ABC possesses effective path planning capabilities and is capable of finding the global optimal solution efficiently.

4.6 Summary

Through ablation experiments, comparative experiments, parameter sensitivity experiments, and robot path planning application experiments, the effectiveness and superiority of SSA-ABC have been systematically verified.

(1) The results of the ablation experiments indicate that the three state-sensing mechanisms proposed in this paper all contribute positively to the algorithm's performance. SSA-ABC, which integrates these three mechanisms, demonstrates the best overall performance across 30 CEC2014 test functions. Friedman test results confirm the effectiveness of each improvement strategy and the significant advantages of the overall algorithm.

(2) In the comparative experiments, SSA-ABC was compared with six typical improved ABC algorithms on the CEC2014 test set. It achieved the best results on 16

functions, significantly outperforming the other algorithms. Both the Wilcoxon and Friedman tests indicate that SSA-ABC possesses statistically significant advantages in terms of solution accuracy, stability, and overall performance. Convergence analysis further validates its excellent optimization capability on complex functions.

(3) Parameter sensitivity experiments show that the selection pressure factor m in the gradient selection model has no significant impact on algorithm performance within the range $[2, 10]$, demonstrating SSA-ABC's good parameter robustness. Accordingly, $m = 6$ is recommended as the default parameter.

(4) In robot path planning applications within a grid map environment, SSA-ABC was compared with the standard ABC algorithm, six improved ABC algorithms, and five typical algorithms including GA, PSO, SSA, DE, and GWO. SSA-ABC achieved the best performance across all metrics, including optimal path length, average path length, standard deviation, number of inflection points, and runtime. It also exhibited the smoothest paths, confirming its practicality and efficiency in complex engineering problems.

In summary, the SSA-ABC algorithm demonstrates excellent convergence, strong stability, and good parameter robustness in both benchmark function optimization and robot path planning, making it an efficient and reliable intelligent optimization algorithm.

Conclusions

This paper addresses three issues associated with the standard ABC algorithm for high-dimensional complex optimization problems: poor initialization quality, difficulty in balancing exploration and exploitation during one-dimensional search, and the attenuation of selection pressure in the later stage. To address these issues, this paper proposes SSA-ABC, which senses the problem dimensionality, search progress, and population ranking, and adaptively adjusts initialization, search, and selection strategies accordingly.

The main contributions of this paper are follows: First, a dimensionality-sensing hybrid initialization mechanism with reflection-hybrid mapping for out-of-bounds handling is proposed. By dynamically adjusting the mixing proportion of Latin hypercube and random initialization via the sigmoid function, the algorithm adaptively balances systemic coverage and random exploration according to the problem dimensionality, while handling out-of-bounds individuals through reflection mapping. Second, a process-sensing elite-mean-guided dynamic-dimension search strategy is designed. This strategy adaptively adjusts the search dimension based on the proportion of function evaluations, facilitating a transition from one-dimensional search in the early stage to multi-dimensional search in the later stage. An elite-mean-guided dimension selection mechanism is also proposed, where half of the dimensions are selected based on the dimension with the greatest difference from the elite individual, while the other half are selected randomly, to match the adaptive search equation. Third, a ranking-sensing elite immunity protection and gradient selection model is proposed. The elite immunity mechanism prevents excellent individuals from being mistakenly reset. A two-stage selection model is constructed, employing roulette wheel selection in the early stage and constructing an elite core using ranking-based arithmetic gradients in the later stage.

Ablation experiments, comparative tests, and convergence analyses on the CEC2014 benchmark functions demonstrate that SSA-ABC outperforms the standard ABC algorithm and six improved ABC algorithms in terms of solution accuracy, convergence speed, and stability. Ablation results indicate that all three state-sensing

mechanisms contribute positively to algorithm performance. Parameter sensitivity experiments show that the selection pressure factor m has no significant impact on algorithm performance. In robot path planning applications, SSA-ABC was compared with the standard ABC algorithm, six improved ABC algorithms, and other typical algorithms such as GA, PSO, SSA, DE, and GWO. Experimental results show that SSA-ABC outperforms the compared algorithms in terms of optimal path length, average path length, standard deviation, number of inflection points, and runtime.

In summary, by introducing state-sensing and adaptive control mechanisms, the SSA-ABC algorithm demonstrates strong convergence, stability, and parameter robustness in benchmark function optimization and robot path planning. Future research may extend the state-sensing mechanism to other swarm intelligence algorithms and explore its applications in scenarios such as multi-robot collaboration and planning in dynamic environments.

Acknowledgements

No funding was received for this work.

REFERENCES

- [1] D. Karaboga, B. Gorkemli, C. Ozturk, et al., “A comprehensive survey: artificial bee colony (ABC) algorithm and applications,” *Artif. Intell. Rev.*, vol. 42, pp. 21–57, 2014, doi: 10.1007/s10462-012-9328-0.
- [2] D. Karaboga and B. Akay, “A comparative study of artificial bee colony algorithm,” *Appl. Math. Comput.*, vol. 214, no. 1, pp. 108–132, 2009, doi: 10.1016/j.amc.2009.03.090.
- [3] D. Karaboga and B. Basturk, “On the performance of artificial bee colony (ABC) algorithm,” *Appl. Soft Comput.*, vol. 8, no. 1, pp. 687–697, 2008, doi: 10.1016/j.asoc.2007.05.007.
- [4] H. Moazen, S. Molaei, L. Farzinvash, et al., “PSO-ELPM: PSO with elite learning, enhanced parameter updating, and exponential mutation operator,” *Inf. Sci.*, vol. 628, pp. 70–91, 2021, doi: 10.1016/j.ins.2023.01.103.
- [5] D. Yousri, D. Allam, M. Eteiba, et al., “Static and dynamic photovoltaic models’ parameters identification using chaotic comprehensive learning particle swarm optimizer variants,” *Energy Convers. Manage.*, vol. 182, pp. 546–563, 2019, doi: 10.1016/j.enconman.2018.12.022.
- [6] N. H. Awad, M. Z. Ali, P. N. Suganthan, et al., “An ensemble sinusoidal parameter adaptation incorporated with L-SHADE for solving CEC2014 benchmark problems,” in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2016, pp. 2958–2965, doi: 10.1109/CEC.2016.7744163.
- [7] X. Zhang, Q. Liu, and Y. Qu, “An adaptive differential evolution algorithm with population size reduction strategy for unconstrained optimization problem,” *Appl. Soft Comput.*, vol. 138, p. 110209, 2023, doi: 10.1016/j.asoc.2023.110209.
- [8] C. Zhong, G. Li, Z. Meng, et al., “A self-adaptive quantum equilibrium optimizer with artificial bee colony for feature selection,” *Comput. Biol. Med.*, vol. 153, p. 106520, 2023, doi: 10.1016/j.combiomed.2022.106520.
- [9] Z. Wang, Q. Pan, L. Gao, et al., “A variable-representation discrete artificial bee colony algorithm for a constrained hybrid flow shop,” *Expert Syst. Appl.*, vol. 254, p.

124349, 2024, doi: 10.1016/j.eswa.2024.124349.

[10] J. Cui, L. Wu, X. Huang, et al., “Multi-strategy adaptable ant colony optimization algorithm and its application in robot path planning,” *Knowl.-Based Syst.*, vol. 288, p. 111459, 2024, doi: 10.1016/j.knosys.2024.111459.

[11] X. Xia, L. Gui, and Z.-H. Zhan, “A multi-swarm particle swarm optimization algorithm based on dynamical topology and purposeful detecting,” *Appl. Soft Comput.*, vol. 67, pp. 126–140, 2018, doi: 10.1016/j.asoc.2018.03.016.

[12] Y. Cui, W. Hu, and A. Rahmani, “A reinforcement learning based artificial bee colony algorithm with application in robot path planning,” *Expert Syst. Appl.*, vol. 203, p. 117389, 2022, doi: 10.1016/j.eswa.2022.117389.

[13] L. Cui, G. Li, Z. Zhu, et al., “A novel artificial bee colony algorithm with an adaptive population size for numerical function optimization,” *Inf. Sci.*, vol. 414, pp. 53–67, 2017, doi: 10.1016/j.ins.2017.05.054.

[14] X. Song, M. Zhao, and S. Xing, “A multi-strategy fusion artificial bee colony algorithm with small population,” *Expert Syst. Appl.*, vol. 142, p. 112921, 2020, doi: 10.1016/j.eswa.2019.112921.

[15] P. Moradi, N. Imanian, N. N. Qader, et al., “Improving exploration property of velocity-based artificial bee colony algorithm using chaotic systems,” *Inf. Sci.*, vol. 465, pp. 130–143, 2018, doi: 10.1016/j.ins.2018.06.059.

[16] A. Singh and K. Deep, “Exploration-exploitation balance in Artificial Bee Colony algorithm: a critical analysis,” *Soft Comput.*, vol. 23, pp. 9525–9536, 2019, doi: 10.1007/s00500-018-3515-0.

[17] G. Zhu and S. Kwong, “Gbest-guided artificial bee colony algorithm for numerical function optimization,” *Appl. Math. Comput.*, vol. 217, no. 7, pp. 3166–3173, 2010, doi: 10.1016/j.amc.2010.08.049.

[18] X. Zhou, J. Lu, J. Huang, et al., “Enhancing artificial bee colony algorithm with multi-elite guidance,” *Inf. Sci.*, vol. 543, pp. 242–258, 2021, doi: 10.1016/j.ins.2020.07.037.

[19] L. Cui, G. Li, Q. Lin, et al., “A novel artificial bee colony algorithm with depth-first search framework and elite-guided search equation,” *Inf. Sci.*, vol. 367, pp. 1012–1044, 2016, doi: 10.1016/j.ins.2016.07.022.

[20] W.-L. Gao and S. Liu, “A modified artificial bee colony algorithm,” *Comput. Oper. Res.*, vol. 39, no. 3, pp. 687–697, 2012, doi: 10.1016/j.cor.2011.06.007.

[21] W. Yu, Z. Zhan, and J. Zhang, “Artificial bee colony algorithm with an adaptive greedy position update strategy,” *Soft Comput.*, vol. 22, pp. 437–451, 2018, doi: 10.1007/s00500-016-2334-4.

[22] X. Zhou, Z. Wu, H. Wang, et al., “Gaussian bare-bones artificial bee colony algorithm,” *Soft Comput.*, vol. 20, pp. 907–924, 2016, doi: 10.1007/s00500-014-1549-5.

[23] H. Peng, C. Deng, and Z. Wu, “Best neighbor-guided artificial bee colony algorithm for continuous optimization problems,” *Soft Comput.*, vol. 23, pp. 8723–8740, 2019, doi: 10.1007/s00500-018-3473-6.

[24] H. Wang, W. Wang, S. Xiao, et al., “Improving artificial bee colony algorithm using a new neighborhood selection mechanism,” *Inf. Sci.*, vol. 527, pp. 227–240, 2020, doi: 10.1016/j.ins.2020.03.064.

[25] W. Wang, Z. Wu, S. Rahnamayan, et al., “Multi-strategy ensemble artificial bee colony algorithm,” *Inf. Sci.*, vol. 279, pp. 587–603, 2014, doi: 10.1016/j.ins.2014.04.013.

[26] W. Xiang, X. Meng, Y. Li, et al., “An improved artificial bee colony algorithm

- based on the gravity model,” *Inf. Sci.*, vol. 429, pp. 49–71, 2018, doi: 10.1016/j.ins.2017.11.007.
- [27] S. Xiao, H. Wang, W. Wang, et al., “Artificial bee colony algorithm based on adaptive neighborhood search and Gaussian perturbation,” *Appl. Soft Comput.*, vol. 100, p. 106955, 2021, doi: 10.1016/j.asoc.2020.106955.
- [28] F. Zou, D. Chen, H. Liu, et al., “A survey of fitness landscape analysis for optimization,” *Neurocomputing*, vol. 503, pp. 129–139, 2022, doi: 10.1016/j.neucom.2022.06.084.
- [29] Y. Borenstein and R. Poli, “Information landscapes and problem hardness,” in *Proc. 7th Annu. Conf. Genet. Evol. Comput. (GECCO)*, 2005, pp. 1425–1431, doi: 10.1145/1068009.1068236.
- [30] X. Zhou, J. Song, S. Wu, et al., “Artificial bee colony algorithm based on online fitness landscape analysis,” *Inf. Sci.*, vol. 619, pp. 603–629, 2022, doi: 10.1016/j.ins.2022.11.056.
- [31] X. Zhou, X. Zhang, W. Gao, et al., “Adaptive multi-population artificial bee colony algorithm based on fitness landscape analysis,” *Appl. Soft Comput.*, vol. 164, p. 111952, 2024, doi: 10.1016/j.asoc.2024.111952.
- [32] J. J. Liang, B. Y. Qu, and P. N. Suganthan, “Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization,” *Zhengzhou Univ., Zhengzhou, China, and Nanyang Technol. Univ., Singapore, Tech. Rep.*, 2013.
- [33] D. Kong, T. Chang, W. Dai, et al., “An improved artificial bee colony algorithm based on elite group guidance and combined breadth-depth search strategy,” *Inf. Sci.*, vols. 442–443, pp. 54–71, 2018, doi: 10.1016/j.ins.2018.02.025.
- [34] L. Cui, G. Li, X. Wang, et al., “A ranking-based adaptive artificial bee colony algorithm for global numerical optimization,” *Inf. Sci.*, vol. 417, pp. 169–185, 2017, doi: 10.1016/j.ins.2017.07.015.
- [35] M. Zhao, J. R. Jiao, X. Y. Song, et al., “Research on dimension-adaptive artificial bee colony algorithm,” *J. Chin. Comput. Syst.*, vol. 45, no. 3, pp. 562–569, 2024.
- [36] G. C. Zeng, J. Ye, S. Y. Song, et al., “Application of an improved whale optimization algorithm in robot path planning,” *J. Gun Launch Control*, vol. 9, pp. 1–9, 2025.
- [37] B. Fang, C. Hu, J. Wan, et al., “Intelligent planning method for UAV inspection path of long-distance transmission lines,” *J. Autom. Technol. Appl.*, vol. 45, no. 1, pp. 28–32, 2026, doi: 10.20033/j.1003-7241.
- [38] E. Bonabeau, *Swarm Intelligence: From Natural to Artificial Systems*. Oxford, U.K.: Oxford Univ. Press, 1999.
- [39] J. Xue and B. Shen, “A novel swarm intelligence optimization approach: sparrow search algorithm,” *Syst. Sci. Control Eng.*, vol. 8, no. 1, pp. 22–34, 2020, doi: 10.1080/21642583.2019.1708830.
- [40] S. Mirjalili, S. M. Mirjalili, and A. Lewis, “Grey wolf optimizer,” *Adv. Eng. Softw.*, vol. 69, pp. 46–61, 2014, doi: 10.1016/j.advengsoft.2013.12.007.
- [41] R. Storn and K. Price, “Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces,” *J. Global Optim.*, vol. 11, pp. 341–359, 1997, doi: 10.1023/A:1008202821328.B.

