Mathematics

https://www.aimspress.com/journal/Math

AIMS Mathematics, 11(2): 3534–3559.
DOI:10.3934/math.2026144
Received: 08 December 2025
Revised: 22 January 2026
Accepted: 02 February 2026
Published: 05 February 2026

Research article

An improved aphid optimization algorithm with soft thresholding

Renyun Liu[1], **Helei Kang**[2], **Rui Bao**[1], **Siyi Gong**[1], **Yifei Yao**[3] **and Yang Wu**[1,*]

[1] Department of Mathematics, Changchun Normal University, Jilin 130032, China
[2] Key Laboratory for Applied Statistics of MOE, School of Mathematics and Statistics, Northeast Normal University, Changchun 130024, China
[3] Department of Computer Science, Changchun Normal University, Jilin 130032, China

* **Correspondence:** Email: 758652790@qq.com; Tel: 17543067492.

Abstract: To address excessive randomness, low late-stage convergence efficiency, and premature convergence in the aphid optimization algorithm (AOA), this study proposes a soft-threshold aphid optimization algorithm (STAOA) from a search-dynamics regulation perspective. The soft-threshold function nonlinearly controls update amplitudes to adaptively suppress or release step sizes, enhancing local exploitation while preserving global exploration and achieving a dynamic balance between them. A soft-threshold-based perturbation strategy further improves the ability to escape local optima, forming a hierarchical search regulation framework. Experiments on 23 benchmark functions, the CEC2019 test suite, and agricultural unmanned aerial vehicle (UAV) path planning tasks show that the STAOA outperforms several representative metaheuristic algorithms in accuracy, stability, and convergence speed, verifying the effectiveness of the soft-threshold mechanism in search-dynamics regulation and UAV path planning optimization.

Keywords: soft-threshold function; aphid optimization algorithm; swarm intelligence optimization; mean Latin hypercube sampling; UAV path optimization algorithm
Mathematics Subject Classification: 90C59, 90C90

1. Introduction

Swarm intelligence optimization algorithms [1] have demonstrated strong adaptability and stability in solving complex optimization problems, and they are generally effective in balancing global exploration and local exploitation. Representative algorithms include particle the swarm optimization algorithm (PSO) [2], the asexual reproduction optimization algorithm (ARO) [3], and the genetic algorithm (GA) [4]. The AOA [5], inspired by aphid foraging behavior, is a recently proposed metaheuristic method that utilizes information exchange among individuals to enhance search efficiency and has shown promising performance across various optimization tasks [6].

Single-objective optimization problems [7] are widely encountered in resource scheduling, engineering design, and hyperparameter tuning in machine learning. Their core task is to identify the optimal solution of a single objective function within a complex search space. In high-dimensional, nonlinear, and highly constrained scenarios, traditional optimization methods often exhibit limited performance. In contrast, metaheuristic algorithms, owing to their low dependence on problem structure, strong global search capability, and high robustness, have become mainstream tools for solving such problems [8]. In recent years, extensive efforts have been devoted to improving algorithmic performance, leading to numerous enhanced strategies and novel algorithms. Nevertheless, the fundamental challenge remains on how to strengthen global exploration while ensuring satisfactory convergence.

To enhance convergence performance, various improvement strategies have been proposed. Chu et al. [9] designed an evolutionary search strategy centered on a slope-based search mechanism, while retaining the global exploration ability of particle swarms, thereby achieving simultaneous improvements in search efficiency and convergence behavior. Chen et al. [10] proposed a dimension-enhanced cuckoo search algorithm, which improves solution quality and convergence in high-dimensional spaces through dimension-independent updating and integration mechanisms. Meng et al. [11] developed a hybrid paradigm-sorting particle swarm variant, in which adaptive parameter control enhances convergence performance and optimization speed. Kang et al. [12] introduced a Brownian motion mutation strategy into the Harris hawks optimization algorithm, using stochastic perturbations to escape local optima and strengthen global search capability. These studies provide important references for iterative improvements of metaheuristic algorithms.

Researchers have also proposed various strategies to enhance population diversity and further improve the exploration–exploitation balance. Mokabberi et al. [13] presented a trend-improved Grey wolf optimizer by adjusting control coefficients to accelerate convergence while enhancing both exploration and exploitation, thus avoiding entrapment in local optima. Adegboye et al. [14] improved the exponential distribution optimizer by combining the sand worm swarm algorithm (SSA) with quadratic interpolation (QI), where QI enhances local search efficiency and accuracy, and SSA provides a global migration mechanism to maintain population diversity. Wang et al. [15] proposed a reinforcement learning-based the PSO, which adopts a hierarchical population structure and uses reinforcement learning strategies to control different levels, thereby improving search efficiency and diversity. Zhu et al. [16] developed a hybrid Black widow optimization that combines the PSO with differential mutation, integrating global exploration and local refinement while maintaining population diversity and preventing premature convergence. Wang et al. [17] proposed a multi-strategy enhanced Chernobyl disaster optimizer, which further strengthens global search ability and population diversity through radial propagation factors, improved position update formulas, chaos-based elite opposition learning, and dimensional search mechanisms. Although these methods improve performance to varying degrees, achieving an optimal balance between convergence speed and population diversity in complex multimodal problems remains challenging.

Despite the remarkable progress of single-objective metaheuristic algorithms, several bottlenecks still exist. Their adaptability to high-dimensional problems is insufficient; affected by the "curse of dimensionality", computational complexity increases rapidly, and algorithms are prone to being trapped in local optima [18]. It is difficult to maintain a balance between exploration and convergence, as dynamic adjustment strategies are limited and imbalance often occurs when optimizing complex objective functions. Algorithms are sensitive to parameters and lack robustness; empirically set parameters lead to large performance fluctuations and limited adaptability [19]. Moreover, compatibility with real-world application scenarios is weak: Most algorithms algorithms are designed based on idealized models and require extensive manual tuning, and there is a lack of unified evaluation and safety analysis frameworks [20].

In response to these issues, the development of single-objective optimization algorithms shows several clear trends. Adaptive intelligent parameter control can dynamically adjust parameters according to objective function characteristics and search states, thus balancing exploration and convergence [21]. Hybrid algorithms and cross-mechanism fusion improve search efficiency by integrating multiple advantageous mechanisms or stochastic strategies [22]. Integration with emerging technologies, such as machine learning–guided search, parallel computing, or quantum computing, can potentially break through efficiency bottlenecks [23]. Algorithm design oriented toward practical scenarios can promote engineering applications through dedicated strategies and evaluation systems. Combinatorial optimization strategies enhance solution stability and accuracy through multiple trials and multi-perspective solving approaches [24].

Although single-objective metaheuristic algorithms have achieved notable progress, issues such as convergence speed, population diversity, and parameter sensitivity still constrain performance improvement in complex multimodal and high-dimensional scenarios [25]. These bottlenecks indicate that relying solely on improvements at the algorithmic framework level can hardly yield sustained significant gains. Research on optimization algorithms is gradually shifting from macroscopic structural design to the fine-grained construction of key operators and local mechanisms.

Against this background, introducing self-regulating mechanisms that can dynamically adapt to search states has become an important direction for improving algorithm stability and robustness [26]. Adaptive parameter control, by dynamically adjusting control parameters according to iteration stages, population distribution, or fitness changes, helps alleviate the contradiction between exploration and convergence and reduces dependence on empirically chosen parameters [27].

Meanwhile, guiding the search process using function forms with nonlinear regulation capability has also attracted increasing attention. Soft-thresholding functions [28], originally widely used in signal processing and sparse optimization, possess adaptive nonlinear characteristics for noise suppression and sparsity regulation [29]. Introducing them into optimization algorithms enables dynamic adjustment of perturbation amplitude and search scale during the search process: Individuals maintain large exploration steps when far from promising regions, and automatically shrink the search range when approaching potential optima, thereby improving local exploitation accuracy and enhancing convergence stability. Therefore, combining soft-thresholding functions with adaptive parameter mechanisms to dynamically regulate the search process at the level of key operators provides a new approach to alleviating the trade-off among convergence speed, population diversity, and parameter sensitivity, and constitutes the core motivation for the method proposed in this study.

Guided by this idea, this paper introduces soft-thresholding functions into AOA and constructs a soft-threshold aphid optimization algorithm (STAOA). Through the soft-threshold mechanism, the search radius is adaptively shrunk when approaching potential optimal solutions, enabling more refined local exploitation; meanwhile, a large search scale is maintained when far from promising regions to preserve global exploration capability. This adaptive regulation process effectively improves convergence accuracy and stability, accelerates convergence while ensuring global search ability, and enhances the overall optimization performance of the algorithm.

The remainder of this paper is organized as follows: Section 2 describes the basic AOA; Section 3 introduces the proposed STAOA; Section 4 presents experimental results; Section 5 concludes the study and discusses future work.

2. AOA

The AOA is a swarm-intelligence metaheuristic algorithm consisting of three phases: Winged-aphid generation, migration, and attack. The transition from migration to attack is determined by the food source stimulus intensity. During migration, aphids perform active or passive flight depending on individual energy levels. The overall framework of the AOA is shown in Figure 1.

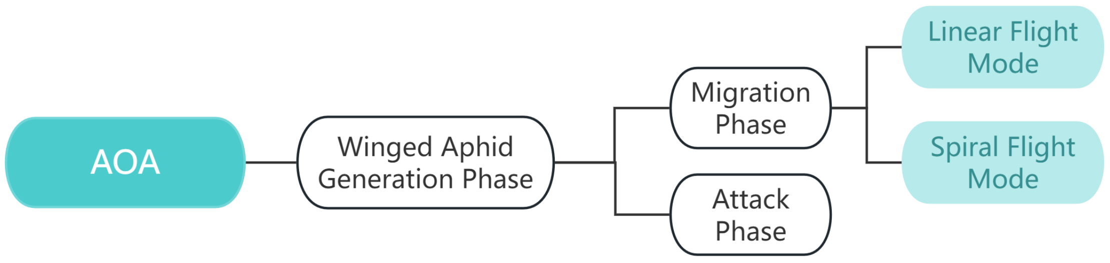

Figure 1. Structure of the AOA.

2.1. *Winged aphid generation phase*

In this stage, aphids form the initial population and begin foraging and migration, corresponding to the algorithm's initialization process. To simulate the division of aphids into small groups and their natural selection, the algorithm employs k-means clustering to partition the population.

2.2. *Migration phase*

Influenced by stored energy, aphids can perform either active or wind-driven passive flight. In the optimization algorithm, this behavior corresponds to particle updating. As migration continues, aphid energy decreases, which is modeled by Eq (2.1).

$$S = 1 - \frac{t}{\max t}, \tag{2.1}$$

where S is the current stored energy, t is the iteration number, and $\max t$ is the maximum number of iterations.

When an aphid has sufficient energy $S/\mu \geq 1$, where μ is a wind influence factor randomly generated in $(-1, 1)$, it performs spiral flight to expand its foraging range. To avoid crowding, particles tend to explore sparser regions. If a particle's crowding distance $D_{dis}(X_n)$ is smaller than the population average $D_{avg\_dis}(X)$, a randomly selected particle X_i is used to guide its search; otherwise, the particle with the largest crowding distance, denoted as X_d, is selected as the guide. The corresponding update rule is given in Eq (2.2).

$$X_n(t+1) = \begin{cases} X_n(t) + \mu e^r \cos(2\pi r)(X_i(t) - X_n(t)) + \mu e^r \sin(2\pi r)(X_i(t) - X_n(t)), & \text{if } D_{dis}(X_n) < D_{avg\_dis}(X), \\ X_n(t) + \mu e^r \cos(2\pi r)(X_d(t) - X_n(t)) + \mu e^r \sin(2\pi r)(X_d(t) - X_n(t)), & \text{otherwise,} \end{cases} \tag{2.2}$$

where X_n represents the n-th particle, X_i is a randomly selected particle, X_d is the particle with the largest crowding distance , and r is the spiral radius (random in $(0, 0.1)$). The crowding distance of particle X_n is defined by Eq (2.3).

$$D_{dis}(X_n) = \sum_{j=1}^{N} d_{n,j}(j \neq n), \tag{2.3}$$

where N is the total number of individuals in the population and $d_{n,j}$ denotes the Euclidean distance between the n-th individual and the j-th individual. A smaller crowding distance indicates that the current individual is located in a more crowded state; conversely, a larger crowding distance implies that the individual lies in a sparser region.

When the energy becomes insufficient $S/\mu < 1$, aphids switch to wind-driven linear flight, modeled by Eq (2.4).

$$X_n(t+1) = \mu * (X_i(t) - X_n(t)), \tag{2.4}$$

where X_i is a randomly selected particle.

2.3. Attack phase

In this phase, aphids approach the food source using olfactory and gustatory cues. By integrating these cues, the algorithm guides particles toward the optimal solution, as defined in Eq (2.5).

$$X_n(t+1) = X_A(t) + r * (X_A(t) - X_n(t) + (X_B(t) - X_n(t))), \tag{2.5}$$

where X_A denotes the global best solution and X_B represents the particle's historical best. The aphid movement trajectory is shown in Figure 2.

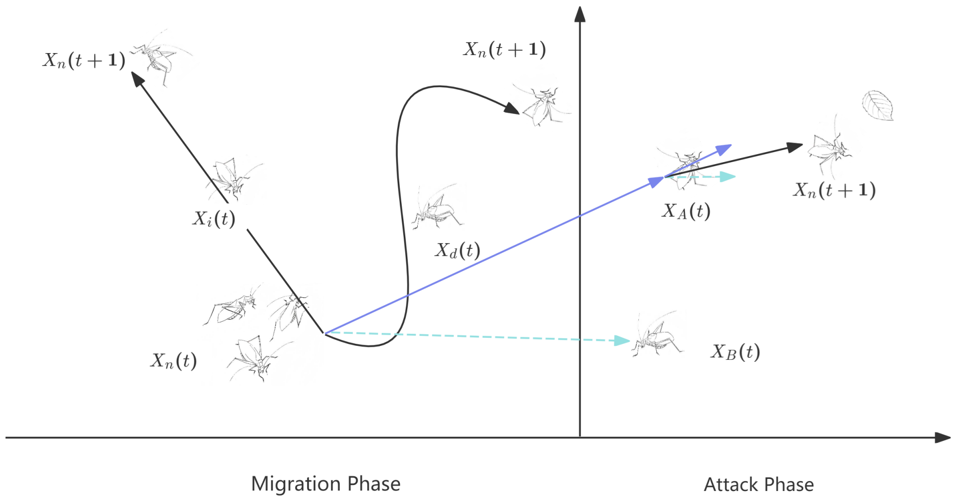

Figure 2. Aphid movement trajectory.

The transition of aphids from the migration stage to the attack stage is induced by their preference for long-wave light in plants. In the proposed algorithm, Eq (2.6) quantifies the light stimulation intensity a, whereas Eq (2.7) models the corresponding preference A of the aphids.

$$a = \frac{1}{1 + e^{-\frac{1}{v+1}}}, \tag{2.6}$$

where v is the Euclidean distance between the population's average position and the food source. A smaller distance indicates stronger long-wave light stimulation, resulting in a larger value of a.

$$A = a * (1 - \frac{t}{\max t}), \tag{2.7}$$

where A denotes the aphid's preference for long-wave light in plants, t is the current iteration number, and max t is the maximum number of iterations. Let rand be a uniformly distributed random number in $(0, 1)$. When $A \geq$ rand, the algorithm enters the attack stage; otherwise, it continues the migration stage.

The AOA achieves efficient global optimization through a multi-energy particle mechanism and multi-level information fusion, and its flowchart is shown in Figure 3. The algorithm is an intelligent optimization method that integrates K-means clustering for initialization with a multi-strategy flight mechanism. Its core procedure is as follows: First, the solution space is pre-partitioned using K-means clustering, which improves the population distribution during the initialization phase. In the main iterative loop, the iteration count serves as the termination criterion, and the algorithm dynamically switches between global exploration and local exploitation modes based on a comparison between parameter A and a random value. When the exploration condition is satisfied, The strategy further selects between linear flight and two spiral flight modes, randomly guided spiral and best particle guided spiral, based on the ratio S/μ and a distance threshold, enabling either broad spatial expansion or fine-grained local search. If the exploration condition is not met, the algorithm directly enters the exploitation phase to intensively search high-quality regions. After each flight, the fitness of the solutions is evaluated, and the optimal solution is output upon termination of the iterations. However, the algorithm still has certain limitations: In high-dimensional problems, random initialization may reduce population diversity and increase the risk of premature convergence; stochastic stage transitions can weaken early-stage search capability and slow late-stage convergence; and relying solely on a single global best particle during the attack phase may be insufficient for complex search spaces. Therefore, improvements in initialization strategies and the balance between exploration and exploitation are required to enhance the algorithm's robustness and performance.

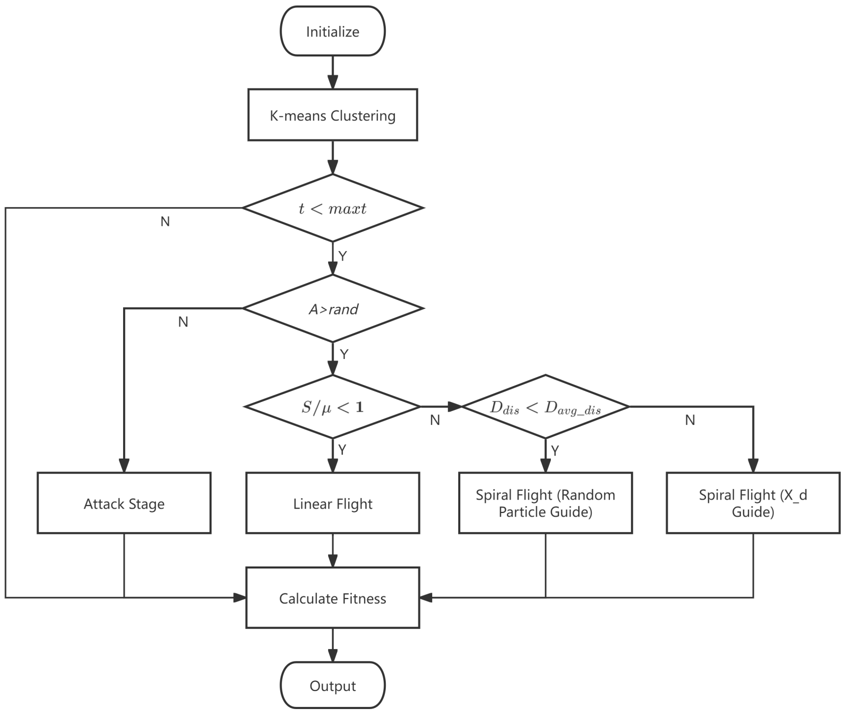

Figure 3. AOA flowchart.

3. The proposed method

3.1. Mean-based Latin hypercube sampling

Latin hypercube sampling (LHS) [31] generates uniformly distributed points in a given space. In this method, each dimension $j \in 1, 2, \ldots, d$ of the d-dimensional input space is divided into n non-overlapping subintervals within $[a, b]$. A random permutation Θ_j is then generated for each dimension to ensure that all n samples cover every subinterval, as expressed in Eq (3.1).

$$x_{ij} = a_j + (b_j - a_j)\left(\frac{\Theta_j(i) - 1 + u_{ij}}{n}\right), u_{ij} \sim U(0,1), \tag{3.1}$$

where, x_{ij} denotes the coordinate of the i-th sample in the j-th dimension, and u_{ij} is a uniformly distributed random perturbation.

Mean-based Latin hypercube sampling (MBLHS) extends the conventional LHS by maximizing the minimum pairwise distance between samples while preserving the stratification and non-repetition constraints in each dimension, as shown in Eq (3.2).

$$\max_{\mathbf{X}} \min_{i \neq j} \|\mathbf{x}_i - \mathbf{x}_j\|_2, \tag{3.2}$$

where x_i and x_j denote two sampling points in the hypercube sampling process. Compared to conventional LHS, MBLHS generates more uniformly distributed initial positions in high-dimensional spaces. Figure 4 presents a comparison of algorithm initialization methods. It depicts the decision space initialization in three-dimensional space under three approaches: random initialization, LHS initialization, and MBLHS initialization. As shown, the MBLHS method produces a notably more uniform distribution across the space compared to the other methods.

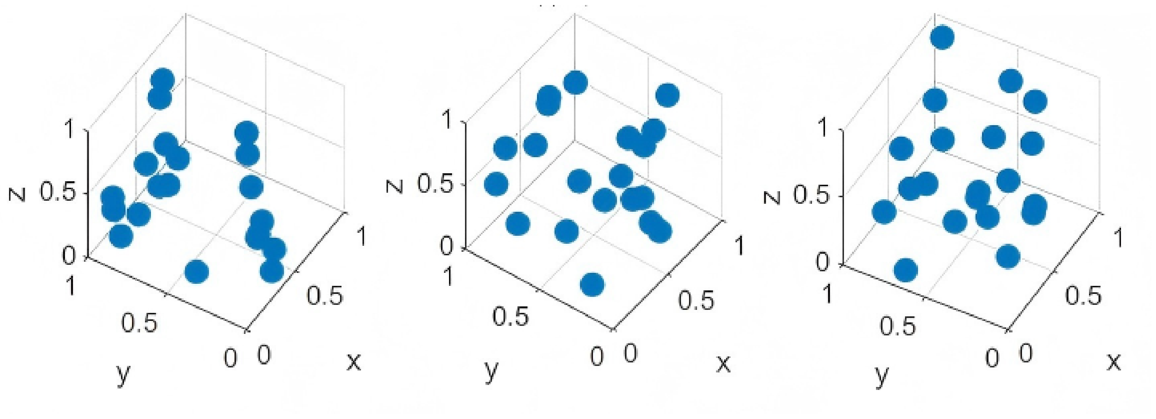

Figure 4. Random initialization, LHS initialization, MBLHS initialization.

3.2. *Adaptive soft-thresholding function*

The threshold function is a type of nonlinear mapping used for coefficient shrinkage. Its basic principle is to set coefficients with magnitudes smaller than a given threshold to zero, while compressing larger coefficients toward zero according to a specific rule. It can be regarded as a smooth gating mechanism. This function has been widely applied in wavelet denoising, Lasso regression, sparse coding, and compressive sensing. The commonly used soft-thresholding function is defined in Eq (3.3).

$$T_\tau(x) = \operatorname{sign}(x) \cdot \max(|x| - \tau, 0), \tag{3.3}$$

where x denotes the original coefficient, τ is the threshold, $T_\tau(x)$ is the processed coefficient, and sign(x) is the sign function. Specifically, when $|x| \leq \tau$, the output is zero, which suppresses small-amplitude noise; when $x < -\tau$, the output is $x + \tau$, shrinking large negative values; when $x > \tau$, the output is $x - \tau$, shrinking large positive values.

On this basis, the traditional soft-thresholding function is improved by introducing a dynamically adaptive threshold instead of a fixed constant. Compared with the rigid nonzero-or-zero behavior of conventional soft thresholding, a dynamic threshold allows the shrinkage intensity to be adjusted during the iterative process. The magnitude of τ reflects the conservativeness of the algorithm: A larger τ leads to stronger shrinkage and more cautious local exploitation around the current optimum, while a smaller τ results in weaker shrinkage and stronger global exploration capability.

After generating a new population, to improve convergence efficiency and reduce the risk of being trapped in local optima, elite particles near the global best solution are subjected to both the soft-thresholding constraint and soft-thresholding perturbation. These two mechanisms cooperate to balance convergence accuracy and search diversity. The update rule under the soft-thresholding constraint is given in Eq (3.4), while the update rule under the soft-thresholding perturbation is given in Eq (3.6). Both mechanisms share the same dynamic threshold parameter τ, forming a unified collaborative control framework.

$$x_{n,\text{new}}^{t} = X_A + T_\tau(\text{Dev}), \tag{3.4}$$

where $x_{n,\text{new}}^{t}$ denotes the particle position after the soft-thresholding constraint, X_A is the global best particle, and $T_\tau(\text{Dev})$ represents applying the soft-thresholding operator in Eq (3.3) to each dimension of the deviation vector. The deviation vector is defined as

$$\text{Dev} = x_n^t - X_A, \tag{3.5}$$

where x_n^t is the position of the n-th particle at iteration t. In the following, x denotes a scalar component of the position vector, while $\mathbf{x}$ denotes the d-dimensional position vector; they represent the same physical quantity in scalar and vector forms, respectively.

From the perspective of function, Eq (3.4) serves as a soft-thresholding constraint whose core role is shrinkage calibration: By applying soft-thresholding to the deviation between particles and the global optimum, redundant deviations are weakened, and elite particles are guided rapidly toward the optimum, reducing ineffective oscillations and improving convergence accuracy and stability. Eq (3.6) acts as a soft-thresholding perturbation whose core role is dynamic exploration: A variable-radius hyperspherical neighborhood is constructed around the global optimum, and a random perturbation term $\boldsymbol{\epsilon}$ is introduced so that particles can deviate moderately within a controlled range, thereby escaping local optima and maintaining global search capability.

$$\mathbf{x}' = \mathbf{x} + \frac{\tau(t)}{\sqrt{-2\ln(p)}} \cdot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d), \tag{3.6}$$

where $\mathbf{x}'$ is the updated particle, $\mathbf{x}$ is the original particle, p denotes the probability that a particle falls outside the perturbation neighborhood (set to 0.2 as the mutation probability in this algorithm), and $\mathbf{I}_d$ is the d-dimensional identity matrix. The parameter $\tau(t)$ is the perturbation radius, defining a d-dimensional hypersphere centered at the global best particle with radius $\tau(t)$. It varies dynamically with the iteration number as Eq (3.7).

$$\tau(t) = \tau_{\max}\left(\frac{t_{\max} - t}{t_{\max}}\right)^k, \tag{3.7}$$

where $\tau_{\max}$ is the maximum threshold, t is the current iteration, $t_{\max}$ is the maximum number of iterations, and k is the decay exponent. The decay exponent k is set to 1.3 and kept constant in all experiments.

The relationship between the two mechanisms is mainly reflected in three aspects. First, at the parameter level, both operations rely on dynamically varying threshold parameters (Eq (3.4) uses τ, while Eq (3.6) uses $\tau(t)$), through which the constraint strength and perturbation range are synchronously regulated. Second, at the objective level, both mechanisms act on elite particles near the global optimum: One is responsible for tightening convergence, while the other is responsible for spreading exploration, jointly enhancing the overall optimization performance. Finally, at the process level, the constraint operation lays the foundation for convergence toward the optimum, and the perturbation operation supplements exploration on this basis, preventing the algorithm from being trapped in local optima due to excessive shrinkage. Together, they form a closed-loop optimization process of constraint-perturbation.

3.3. AOA based on a soft-thresholding function (STAOA)

The AOA often struggles to maintain a stable balance between convergence speed and population diversity, particularly in high-dimensional search spaces. Random stage transitions can reduce global exploration in the early iterations or cause unnecessary oscillations in the later stages. To address these issues, the STAOA incorporates soft-threshold constraints and perturbation mechanisms into its evolutionary process, improving both exploration and convergence performance.

The main idea of the STAOA is to regulate particle movement adaptively around the global best solution by combining deterministic shrinkage and stochastic perturbation. When particles approach promising regions, their movement is gradually compressed to enhance exploitation; meanwhile, controlled perturbations are introduced to prevent premature convergence and preserve diversity. The overall framework is shown in Figure 5.

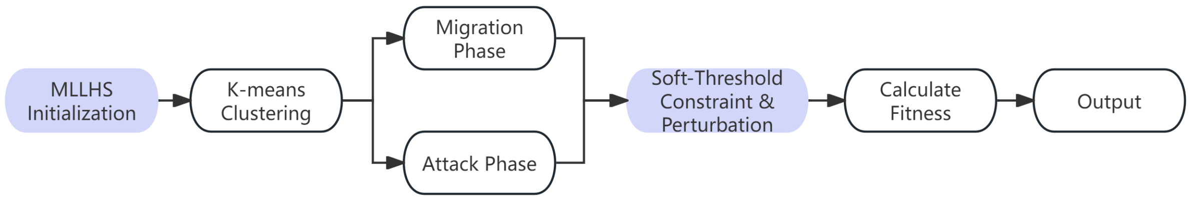

Figure 5. Structure of STAOA.

The proposed STAOA improves upon the standard AOA by addressing the unstable trade-off between convergence speed and population diversity, particularly in high-dimensional search spaces. The algorithm employs MLLHS for population initialization and introduces soft-threshold constraints and perturbation mechanisms into the evolutionary process, thus forming an adaptive shrink-explore search strategy. The pseudocode of the STAOA is presented in Algorithm 1.

Algorithm 1: Aphid optimization algorithm based on a soft-thresholding function

```
Number of aphid individuals $N$; number of clusters $n_c$; search space dimension $M$;
maximum number of iterations $t_{\max}$; current iteration $t = 1$;
initialize population $X = \{X_1, \dots, X_N\}$ randomly;
evaluate fitness of all individuals and obtain initial global best $X_A$;
while $t \le t_{\max}$ do
    for $i = 1 : N$ do
        Calculate the fitness of aphid $X_i$;
        if $A <$ rand then
            Update $X_i$ according to Eqs. (2.2) and (2.4);
        else
            Update $X_i$ according to Eq. (2.5);
    Evaluate fitness of all updated aphids and update $X_A$;
    Select elite set $\mathcal{E}$ from population according to fitness to $X_A$;
    for each $X_n \in \mathcal{E}$ do
        Compute deviation vector Dev $= X_n - X_A$;
        Apply soft-threshold operator to Dev and update $X_n$ using Eq. (3.4);
    Compute dynamic threshold $\tau(t)$ according to Eq. (3.7);
    for each $X_n \in \mathcal{E}$ do
        Generate $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_M)$;
        Update $X_n$ using soft-threshold perturbation in Eq. (3.6);
    Recalculate fitness of particles in $\mathcal{E}$ and update $X_A$;
    $t = t + 1$;
return $X_A$;
```

Algorithm framework: In each iteration, the STAOA consists of three main stages:

1) Standard aphid position update;
2) Soft-threshold constraint applied to elite individuals;
3) Soft-threshold perturbation applied to elite individuals.

Population initialization: The initial population

$$X = \{X_1, X_2, \dots, X_N\} \tag{3.8}$$

is generated using the MBLHS strategy (Eqs (3.1)-(3.2)), which ensures a more uniform distribution of individuals across the decision space and enhances initial diversity, especially in high-dimensional problems. The fitness of all individuals is evaluated to determine the initial global best solution X_A.

Standard position update: At iteration t, all aphids update their positions according to the original AOA movement rules:

- If $A <$ rand, positions are updated using Eqs (2.2) and (2.4);
- Otherwise, positions are updated using Eq (2.5).

After the standard update, the fitness of all individuals is re-evaluated, and X_A is updated if a better solution is found.

Elite selection: To focus refinement on promising individuals, an elite set E is selected from the population based on proximity to X_A. Only individuals in E participate in the subsequent soft-threshold operations, reducing computational cost and preventing excessive disturbance of the entire population.

Soft-threshold constraint: For each elite particle $X_n \in E$, the deviation vector from the global best is computed as Eq (3.5).

A soft-threshold operator is applied to each dimension of Dev as Eq (3.3), where deviations smaller than the threshold τ are set to zero, and larger deviations are shrunk toward zero. The particle is then updated as Eq (3.4).

This operation eliminates minor and ineffective deviations, aligns particles with the global best in insignificant dimensions, suppresses redundant oscillations near the optimum, and accelerates deterministic local convergence.

Dynamic threshold: The threshold τ is dynamically adjusted according to the iteration number as Eq (3.7), where $\tau_{\max}$ is the maximum threshold and k is the decay exponent. This design allows a large search radius in early iterations for global exploration and a gradually shrinking radius in later iterations for fine exploitation.

Soft-threshold perturbation: To prevent premature convergence caused by excessive shrinkage, a soft-threshold perturbation is applied to elite particles. For each elite particle $X_n \in E$, with probability p, a Gaussian random vector is generated with probability p, and the particle is updated according to the soft-threshold perturbation rule (see Eq (3.6)).

This perturbation generates candidate solutions within a dynamic hyperspherical neighborhood centered at the global best solution, with a radius controlled by $\tau(t)$. It enables elite particles to escape local optima while remaining in promising regions, thereby maintaining population diversity without compromising convergence stability. After applying the perturbation, the fitness of updated particles is re-evaluated, and the global best solution X_A is updated accordingly.

After perturbation, the fitness of elite particles is recalculated, and the global best solution X_A is updated again.

Cooperative mechanism: The soft-threshold constraint and perturbation operate cooperatively to form a shrink-explore mechanism:

1) The constraint step enforces deterministic shrinkage toward the global best, ensuring fast and stable convergence.
2) The perturbation step introduces controlled randomness, enabling exploration around the current optimum and preventing premature convergence.
3) Both mechanisms share the dynamic threshold $\tau(t)$, allowing adaptive balance between exploitation and exploration throughout the evolutionary process.

These characteristics constitute the core innovations of the STAOA and significantly enhance the performance and robustness of the original AOA, particularly in complex and high-dimensional optimization problems.

4. Results

4.1. Benchmark functions and experimental design

To evaluate the performance of the STAOA, this study conducts experiments on 23 benchmark functions and the CEC 2019 test functions [30, 31]. The inclusion of the CEC 2019 test set aims to assess the algorithm's capability in solving non-convex optimization problems. The dimensional settings of the test functions are listed in Table 1.

Table 1. Dimensional settings of the test functions.

Test functions		Dimensional
23 benchmark functions	f_1-f_{12}	30
	f_{13},f_{14},f_{16}-f_{18}	2
	f_{15},f_{19},f_{21}-f_{23}	4
	f_{20}	6
CEC 2019 test functions	$CEC01$	9
	$CEC02$	16
	$CEC03$	18
	$CEC04$-$CEC10$	10

Table 2. Parameter setting of various optimization algorithms.

Algorithm	Parameter	Value
STAOA	Adaptive Gaussian mutation probability	0.2
AOA	NC	500
	n_c	30
NRBO	Trap-avoidance operator	0.6
MShOA	Polarization angle range parameter	10
CPO	Control of odor concentration intensity	100
	Diffusion coefficient	0.6
	$beta$	1.5
DBO	$P_{percent}$	0.2
SHO	omega	1.5
	u	0.05
	v	0.05
	l	0.05
RRTO	C	10

The experiments consist of two parts. First, the performance of the STAOA is compared with several recent algorithms, including the Newton-Raphson-based optimizer algorithm (NRBO) [32], the modified spotted hyena optimization algorithm (MShOA) [33], the crested porcupine optimizer algorithm (CPO) [34], the dung beetle optimizer algorithm (DBO) [35], the sea-horse optimizer algorithm (SHO) [36], and the RRT-based optimizer algorithm (RRTO) [37]. The parameter settings of the comparison algorithms are consistent with those reported in the original paper; some of the parameter configurations are shown in the table 2. Second, the STAOA is applied to the path planning problem of plant-protection unmanned aerial vehicles (UAVs) and is compared experimentally with the GA [38] and PSO [39]. To ensure the statistical reliability of the results, all algorithms were independently executed 30 times under identical parameter settings, and the mean value of the results was used as the performance evaluation metric. The population size of each algorithm was uniformly set to 30, and the maximum number of iterations was 500. All simulation experiments were conducted on the MATLAB R2024b platform, and the hardware environment consisted of a Windows 11 64-bit Professional operating system with 16 GB of memory.

4.2. Performance comparison with algorithms

4.2.1. Experimental results

The experimental results are summarized in Tables 3–5, are analyzed using standard statistical indicators, including the mean (Mean), best value (Min), and variance (Var). In addition, the average rank value (ARV) obtained from Friedman ranking and the +/–/= win–loss–tie statistics, based on Wilcoxon signed-rank tests, are reported. Here, + indicates that the compared algorithm outperforms the STAOA, – indicates that it underperforms, and = denotes no significant difference.

On the set of 23 benchmark functions, the STAOA achieves the best or tied-best mean results on the majority of test functions. In particular, for unimodal functions and some low- or medium-difficulty multimodal functions (e.g., f_1–f_4 and f_8–f_{11}), the STAOA consistently converges to the theoretical optimum or values extremely close to it. The mean values are often zero or of a very small magnitude, and the variances are near zero, indicating exceptional convergence accuracy and stability. In contrast, other algorithms exhibit larger performance fluctuations across functions, highlighting their stronger problem dependency.

For more complex multimodal and combinatorial functions (e.g., f_5–f_7 and f_{12}–f_{14}), the STAOA maintains strong competitiveness. Its mean values are generally better than or comparable to most competing algorithms, and its variances are consistently smaller. This demonstrates that the STAOA effectively balances global exploration in complex search spaces with stable local exploitation in later iterations, thereby mitigating the risk of being trapped in local optima.

Across all 23 benchmark functions, the STAOA achieves an ARV of 2.65, the lowest among all algorithms, indicating first-place overall ranking. Furthermore, the Wilcoxon-based +/–/= statistics confirm that the STAOA achieves significantly more wins than losses against most comparative algorithms, with a proportion of ties, demonstrating stable and generalizable superiority.

On the CEC2019 test suite, characterized by high dimensionality, strong multimodality, variable interaction, and complex landscapes, the STAOA achieves the best or near-best mean results on several functions, including CEC01, CEC02, CEC04, CEC05, and CEC07, with relatively small variances. Even on functions where it does not attain the absolute best, the STAOA generally remains within the top performance tier, outperforming or at least performing on par with most competing algorithms.

Table 3. Results of 23 benchmark functions.

Function		Algorithm							
		STAOA	AOA	NRBO	MShOA	CPO	DBO	SHO	RRTO
	Mean	0.00E+00	0.00E+00=	1.35E-280-	0.00E+00=	3.74E-107-	3.51E-116-	4.58E-140-	3.89E-60-
f_1	Min	0.00E+00	0.00E+00	5.30E-299	0.00E+00	6.08E-231	1.81E-165	4.83E-147	1.66E-87
	Var	0.00E+00	0.00E+00	0.00E+00	0.00E+00	4.19E-212	3.70E-230	5.38E-278	4.41E-118
	Mean	0.00E+00	0.00E+00=	8.00E-142-	4.93E-210-	1.69E-61-	1.44E-53-	3.08E-78-	2.10E-13-
f_2	Min	0.00E+00	0.00E+00	1.45E-148	0.00E+00	1.07E-97	2.44E-80	5.26E-81	1.96E-39
	var	0.00E+00	0.00E+00	1.72E-281	0.00E+00	2.80E-121	6.22E-105	3.02E-155	1.33E-24
	Mean	0.00E+00	0.00E+00=	3.58E-264-	0.00E+00=	5.93E-92-	1.56E-70-	3.85E-97-	1.94E-08-
f_3	Min	0.00E+00	0.00E+00	5.32E-284	0.00E+00	3.98E-208	3.81E-139	1.50E-104	6.52E-41
	var	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.05E-181	7.28E-139	4.09E-192	1.10E-14
	Mean	0.00E+00	3.77E-320-	8.16E-140-	1.21E-235-	5.37E-58-	2.44E-55-	1.03E-56-	2.13E-40-
f_4	Min	0.00E+00	0.00E+00	3.76E-145	0.00E+00	3.66E-107	7.72E-79	3.90E-60	6.92E-48
	var	0.00E+00	0.00E+00	1.20E-277	0.00E+00	8.64E-114	9.72E-109	6.82E-112	1.33E-78
	Mean	1.90E-10	3.82E-12+	2.79E+01-	2.90E+01-	2.69E+01-	2.57E+01-	2.82E+01-	1.82E+01-
f_5	Min	0.00E+00	0.00E+00	2.63E+01	2.89E+01	1.02E-02	2.53E+01	2.73E+01	0.00E+00
	var	7.22E-19	4.36E-22	6.51E-01	3.68E-04	2.59E+01	5.02E-02	2.35E-01	1.98E+02
	Mean	2.73E-07	2.43E-13-	2.81E+00-	7.04E+00-	2.24E-01-	2.59E-03-	3.14E+00-	3.09E-23+
f_6	Min	0.00E+00	0.00E+00	2.03E+00	6.02E+00	6.37E-05	1.56E-05	2.25E+00	0.00E+00
	var	1.49E-12	1.78E-24	2.31E-01	1.76E-01	1.46E-01	6.33E-05	3.41E-01	2.86E-44
	Mean	2.13E-02	2.65E-02+	3.73E-04+	1.24E-04+	1.36E-04+	1.50E-03+	1.14E-04+	6.04E-04+
f_7	Min	2.66E-04	1.09E-03	2.57E-05	7.97E-06	2.26E-06	1.24E-04	5.59E-07	6.48E-05
	var	3.80E-04	5.14E-04	1.39E-07	1.48E-08	1.42E-08	1.61E-06	1.28E-08	2.66E-07
	Mean	-1.26E+04	-1.26E+04=	-4.79E+03-	-4.52E+03-	-4.25E+03-	-7.98E+03-	-6.05E+03-	-1.26E+04=
f_8	Min	-1.26E+04	-1.26E+04	-6.17E+03	-5.83E+03	-8.18E+03	-1.24E+04	-7.21E+03	-1.26E+04
	var	6.57E-23	6.73E-21	3.73E+05	3.51E+05	3.42E+06	1.90E+06	3.35E+05	7.40E-22
	Mean	0.00E+00=	0.00E+00=	0.00E+00=	0.00E+00=	0.00E+00=	1.39E+00-	0.00E+00=	9.95E+00-
f_9	Min	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	var	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	3.79E+01	0.00E+00	2.05E+02
	Mean	4.44E-16	4.44E-16=	4.44E-16=	4.44E-16=	4.44E-16=	4.44E-16=	3.88E-15-	4.44E-16=
f_{10}	Min	4.44E-16	4.44E-16	4.44E-16	4.44E-16	4.44E-16	4.44E-16	4.44E-16	0.00E+00
	var	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	4.21E-31	4.44E-16
	Mean	0.00E+00	0.00E+00=	0.00E+00=	0.00E+00=	0.00E+00=	3.10E-03-	0.00E+00=	1.44E-16-
f_{11}	Min	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	var	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	2.87E-04	0.00E+00	6.25E-31
	Mean	7.19E-20	9.90E-19-	2.27E-01-	1.27E+00-	8.29E-04-	3.57E-04-	2.62E-01-	6.11E-19-
f_{12}	Min	9.42E-34	1.57E-32	1.05E-01	5.14E-01	1.40E-10	2.55E-07	1.32E-01	1.57E-32
	var	1.02E-37	1.72E-35	4.11E-03	7.36E-02	5.70E-06	1.56E-06	3.74E-03	1.12E-35
	Mean	1.47E-29	7.00E-18-	2.19E+00-	3.00E+00-	4.73E-03-	6.32E-01-	1.98E+00-	8.45E-06-
f_{13}	Min	1.35E-32	1.35E-32	1.51E+00	2.95E+00	5.14E-08	1.26E-02	1.33E+00	1.35E-32
	var	2.10E-57	1.47E-33	1.64E-01	7.58E-05	3.71E-04	1.80E-01	1.07E-01	2.14E-09
	Mean	9.98E-01	9.98E-01=	2.44E+00-	4.97E+00-	1.06E+01-	1.56E+00-	5.17E+00-	9.98E-01=
f_{14}	Min	9.98E-01	9.98E-01	9.98E-01	9.98E-01	1.99E+00	9.98E-01	9.98E-01	9.98E-01
	var	3.37E-32	1.02E-32	8.67E+00	2.30E+01	1.44E+01	3.34E+00	2.06E+01	6.38E-32
	Mean	4.61E-04	1.64E-03-	1.77E-03-	2.10E-03-	5.52E-04-	7.71E-04-	1.77E-03-	3.66E-04+
f_{15}	Min	3.08E-04	8.34E-04	3.07E-04	3.41E-04	3.09E-04	3.07E-04	3.08E-04	3.13E-04
	var	1.11E-07	2.44E-08	2.56E-05	8.93E-06	5.91E-07	1.04E-07	2.60E-05	5.94E-09
	Mean	-1.03E+00	-1.03E+00=	-1.03E+00=	-1.03E+00=	-1.03E+00=	-1.03E+00=	-1.03E+00=	-1.03E+00=
f_{16}	Min	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00
	var	5.23E-12	2.43E-07	2.84E-31	1.75E-10	2.13E-05	3.66E-31	3.66E-17	6.89E-13
	Mean	3.98E-01	3.99E-01-	3.98E-01=	3.99E-01-	4.03E-01-	3.98E-01=	7.10E-01-	3.98E-01=
f_{17}	Min	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01
	var	1.66E-06	8.30E-06	0.00E+00	2.00E-06	8.81E-04	0.00E+00	1.39E+00	5.24E-13
	Mean	3.00E+00	2.38E+01-	5.70E+00-	3.82E+00-	3.00E+00=	3.00E+00=	3.00E+00=	3.00E+00=
f_{18}	Min	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00
	var	1.12E-06	1.91E+02	2.19E+02	2.01E+01	4.08E-06	3.51E-30	7.47E-16	1.45E-09

The overall ARV based on Friedman ranking is 2.90, still the lowest among all algorithms, confirming that the STAOA maintains the leading rank even on high-difficulty problems. The Wilcoxon-based +/–/= results further indicate that the STAOA consistently occupies a superior or non-inferior position in pairwise comparisons with other algorithms.

To validate these observations statistically, Friedman nonparametric tests and Nemenyi post-hoc comparisons were applied. On the 23 benchmark functions, the Friedman test yields $p = 4.49E - 05 < 0.05$, confirming significant performance differences among algorithms. The corresponding Nemenyi critical difference is $CD = 2.1893$. Comparing average ranks, the STAOA exceeds this threshold relative to several competitors, demonstrating statistically significant superiority. On the CEC2019 suite, the Friedman test produces $p = 1.20E - 5 \ll 0.01$, with a Nemenyi critical difference of $CD = 3.3203$. Despite the larger critical difference, the STAOA maintains the best or near-best average rank, showing statistically significant or near-significant advantages over several algorithms.

In summary, combining basic numerical results, ARV and Wilcoxon-based +/–/= statistics, and statistical analyses via Friedman tests and Nemenyi post-hoc comparisons, the STAOA demonstrates superior or at least competitive performance compared to many state-of-the-art algorithms in terms of convergence accuracy, stability, robustness, and overall effectiveness. These advantages are consistent across different benchmark types and high-difficulty CEC2019 problems, supported by statistically significant evidence, verifying the effectiveness and reliability of the proposed algorithmic improvements.

Table 4. Results of 23 benchmark functions.

Function		Algorithm							
		STAOA	AOA	NRBO	MShOA	CPO	DBO	SHO	RRTO
f_{19}	Mean	-3.86E+00	-3.85E+00-	-3.86E+00=	-3.86E+00=	-3.86E+00=	-3.86E+00=	-3.86E+00=	-3.86E+00=
	Min	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00
	var	2.38E-10	1.90E-02	5.33E-30	4.34E-05	2.11E-06	7.43E-06	1.34E-05	1.00E-12
f_{20}	Mean	-3.27E+00	-2.87E+00-	-3.25E+00-	-3.16E+00-	-3.25E+00-	-3.24E+00-	-3.05E+00-	-3.32E+00+
	Min	-3.32E+00	-3.23E+00	-3.32E+00	-3.27E+00	-3.32E+00	-3.32E+00	-3.32E+00	-3.32E+00
	var	3.68E-03	3.98E-02	4.94E-03	6.06E-03	4.27E-02	7.72E-03	4.78E-02	8.01E-09
f_{21}	Mean	-1.02E+01	-1.02E+01=	-8.18E+00-	4.58E+00-	-1.02E+01=	-7.02E+00-	-6.13E+00-	-8.73E+00-
	Min	-1.02E+01	-1.02E+01	-1.02E+01	-5.52E+00	-1.02E+01	-1.02E+01	-1.01E+01	-1.02E+01
	var	3.90E-17	7.38E-28	7.61E+00	1.46E-01	1.11E-08	6.95E+00	5.25E+00	7.22E+00
f_{22}	Mean	-1.04E+01	-1.04E+01=	-7.72E+00-	-4.89E+00-	-1.04E+01=	-7.47E+00-	-6.14E+00-	-1.00E+01-
	Min	-1.04E+01	-1.04E+01	-1.04E+01	-9.56E+00	-1.04E+01	-1.04E+01	-1.03E+01	-1.04E+01
	var	2.80E-17	5.49E-27	9.08E+00	8.87E-01	2.46E-08	8.13E+00	4.41E+00	1.80E+00
f_{23}	Mean	-1.05E+01	-1.05E+01=	-8.05E+00-	-4.86E+00-	-1.05E+01=	-8.92E+00-	-5.76E+00-	-9.46E+00-
	Min	-1.05E+01	-1.05E+01	-1.05E+01	-1.04E+01	-1.05E+01	-1.05E+01	-1.03E+01	-1.05E+01
	var	2.39E-17	1.61E-28	7.51E+00	1.56E+00	2.55E-09	6.29E+00	2.79E+00	4.80E+00
+/-/=			(2/9/12)	(1/16/6)	(1/15/7)	(1/13/9)	(1/17/5)	(1/17/5)	(4/12/7)
ARV		2.65	3.63	4.80	5.46	4.50	5.07	5.70	4.22

Table 5. Results of CEC2019 test functions.

Function		Algorithm							
		STAOA	AOA	NRBO	MShOA	CPO	DBO	SHO	RRTO
CEC01	Mean	4.44E+04	6.47E+05-	4.69E+03-	2.16E+05-	5.75E+04-	1.44E+09-	4.70E+04-	6.86E+04-
	Min	4.08E+04	6.22E+05	3.76E+04	7.32E+04	4.23E+04	3.96E+04	4.05E+04	4.33E+03
	var	6.95+E06	2.17E+07	1.54E+08	2.15E+10	8.29E+07	1.57E+19	9.44E+06	1.41E+09
CEC02	Mean	1.73E+01	1.84E+01-	1.74E+01-	1.91E+01-	1.74E+01-	1.74E+01-	1.74E+01-	1.80E+01-
	Min	1.73E+01	1.79E+01	1.73E+01	1.78E+01	1.73E+01	1.73E+01	1.73E+01	1.76E+01
	var	8.35E-07	1.01E-02	1.22E-02	5.15E-01	9.76E-05	4.18E-29	6.00E-03	5.48E-02
CEC03	Mean	1.27E+01=	1.27E+01=	1.27E+01=	1.27E+01=	1.27E+01=	1.27E+01=	1.27E+01=	1.27E+01=
	Min	1.27E+01	1.27E+01	1.27E+01	1.27E+01	1.27E+01	1.27E+01	1.27E+01	1.27E+01
	var	1.76E-15	5.66E-07	8.31E-10	3.57E-07	1.96E-13	7.61E-11	1.00E-11	3.78E-11
CEC04	Mean	1.00E+02	7.97E+03-	1.39E+03-	8.24E+03-	2.13E+02-	2.30E+02-	1.06E+03-	9.31E+01+
	Min	5.32E+01	3.83E+03	3.67E+02	6.89E+02	2.07E+01	1.59E+01	8.08E+01	6.83E+01
	var	7.20E+02	5.70E+06	1.07E+06	2.79E+07	5.19E+04	5.58E+04	1.52E+06	2.78E+02
CEC05	Mean	1.26E+00	3.79E+00-	2.06E+00-	3.60E+00-	1.94E+00-	1.44E+00-	1.84E+00-	1.51E+00-
	Min	1.08E+00	2.57E+00	1.63E+00	2.02E+00	1.10E+00	1.09E+00	1.37E+00	1.35E+00
	var	5.17E-02	4.25E-01	4.51E-02	1.10E+00	5.91E-01	1.10E-01	1.29E-01	9.19E-03
CEC06	Mean	1.12E+01	1.30E+01-	9.71E+00+	1.09E+01+	1.03E+01+	1.03E+01+	7.53E+00+	1.03E+01+
	Min	5.67E+00	1.02E+01	7.66E+00	8.75E+00	2.91E+00	6.81E+00	5.02E+00	8.09E+00
	var	7.67E+00	4.54E+00	1.01E+00	8.11E-01	5.17E+00	1.56E+00	1.34E+00	1.02E+00
CEC07	Mean	1.15E+02	1.07E+03-	4.20E+02-	8.62E+02-	7.45E+02-	5.33E+02-	2.60E+02-	4.59E+02-
	Min	3.79E-01	5.88E+02	1.22E+01	3.69E+02	-7.42E+01	8.91E+01	-1.09E+01	1.51E+02
	var	1.38E+04	8.41E+04	3.75E+04	8.21E+04	4.17E+05	5.80E+04	1.87E+04	1.63E+04
CEC08	Mean	5.15E+00	6.91E+00-	5.19E+00-	6.18E+00-	6.63E+00-	5.87E+00-	5.01E+00+	5.69E+00-
	Min	3.33E+00	6.15E+00	4.07E+00	5.11E+00	4.82E+00	4.55E+00	3.12E+00	3.65E+00
	var	6.924E-01	1.21E-01	3.14E-01	2.24E-01	3.22E-01	4.51E-01	4.69E-01	3.39E-01
CEC09	Mean	2.84E+00	1.63E+03-	3.43E+01-	1.23E+03-	3.10E+00-	2.72E+00+	7.13E+01-	3.39E+00-
	Min	2.50E+00	2.74E+02	4.50E+00	1.63E+02	2.62E+00	2.39E+00	4.66E+00	2.76E+00
	var	1.03E-01	2.09E+05	4.93E+02	7.20E+05	1.64E-01	8.21E-02	1.81E+04	1.39E-01
CEC10	Mean	2.05E+01	2.08E+01-	1.97E+01+	2.05E+01=	2.04E+01+	2.04E+01+	2.00E+01+	2.00E+01+
	Min	2.01E+01	2.04E+01	9.27E+00	2.02E+01	2.00E+01	2.02E+01	1.68E+01	7.78E+00
	var	8.35E-02	7.19E-02	5.27E+00	1.33E-02	5.08E-02	1.23E-02	3.77E-01	5.34E+00
+/-/=			(0/9/1)	(2/7/1)	(1/7/2)	(2/7/1)	(3/6/1)	(3/6/1)	(3/6/1)
ARV		2.90	7.35	3.5	6.60	4.60	4.15	3.40	3.40

4.2.2. Algorithm stability analysis

Stability reflects an algorithm's consistency and reliability across multiple independent runs under identical experimental conditions. A stable algorithm produces similar results repeatedly and is not highly sensitive to random initialization. In this study, the stability of the STAOA is assessed using three statistical indicators: the mean (Mean), minimum (Min), and variance (Var) of the best fitness values from multiple independent runs. Smaller variances and smaller gaps between mean and minimum values indicate higher stability and stronger robustness.

All algorithms were executed independently multiple times for each benchmark function under the same parameter settings. For each function, the Mean, Min, and Var of the results are reported in Tables 3–5.

On the 23 benchmark functions, the STAOA demonstrates excellent stability. For functions f_1–f_4, f_9–f_{11}, and f_{16}, the variance is zero or near-zero, indicating almost identical solutions across independent runs. Even on complex functions such as f_5, f_6, f_7, and f_{15}, the variance remains extremely low (e.g., on f_5 and f_6, the variance is on the order of 10^{-12}–10^{-19}), significantly lower than most comparative algorithms. This confirms the STAOA's robustness under varying random seeds.

Moreover, the STAOA's mean values are highly consistent with the corresponding minimum values across most benchmark functions. In many cases, they are identical or differ only slightly, indicating that the algorithm does not rely on occasional successful runs but consistently achieves near-optimal or optimal solutions. In contrast, several comparative algorithms show large variances and notable gaps between mean and minimum values, reflecting unstable search behavior and sensitivity to initialization.

Similar trends are observed in the CEC2019 suite. The STAOA maintains low variances on most functions, whereas some competitors exhibit fluctuations spanning several orders of magnitude. The consistency between mean and minimum values further confirms the algorithm's reliability in challenging optimization environments.

Overall, the results clearly demonstrate that the STAOA exhibits outstanding stability. Its extremely low variance and high mean-minimum consistency indicate robustness to random initialization, delivering reliable and repeatable solutions. Therefore, the STAOA is not only competitive in solution quality but also highly stable, making it suitable for practical optimization scenarios where reliability and robustness are critical.

4.2.3. Parameter stability analysis

To evaluate the impact of parameter P on algorithmic performance, a sensitivity analysis was performed for $P \in \{0.1, 0.2, \ldots, 0.9\}$, as shown in Table 6. Parameter sensitivity was assessed using the Friedman test (23 benchmark functions: $p = 0.611$; CEC2019 functions: $p = 0.266$) followed by the Nemenyi post-hoc test (with critical difference values of 2.505 and 3.799, respectively). The statistical results indicate that variations in P do not lead to significant differences in algorithmic performance ($p > 0.05$), although minor numerical fluctuations are observed. For the 23 benchmark functions, the best average rank (4.478) is obtained at $P = 0.2$, and similarly, $P = 0.2$ also yields the optimal average rank (3.3) on the CEC2019 functions. Overall, the algorithm demonstrates strong robustness with respect to parameter P, and thus setting $P = 0.2$ is recommended for practical engineering applications.

Table 6. Sensitivity analysis of parameter P.

Parameter P	ARV on 23 Benchmark functions	Average rank (23 Benchmarks)	ARV on CEC2019 functions	Average rank (CEC2019)
0.1	4.6	3	4.3	3
0.2	4.5	1	3.3	1
0.3	5.5	9	5.5	6
0.4	5.0	5	4.2	2
0.5	4.5	2	5.1	5
0.6	5.0	4	4.9	4
0.7	5.3	8	5.8	7
0.8	5.3	7	5.9	8
0.9	5.2	6	6.0	9

4.3. Application of STAOA to plant-protection UAV path planning

4.3.1. Problem introduction

This section addresses the following problem: In a working area consisting of multiple non-contiguous rectangular plots, a plant-protection UAV [40] must depart from the base, perform pesticide spraying over all plots, and return to the base. During operation, the UAV is required to minimize its flight distance to enhance operational efficiency and conserve resources. Each plot is treated with a back-and-forth spraying pattern [41], with designated entry and exit points as shown in Figure 6.

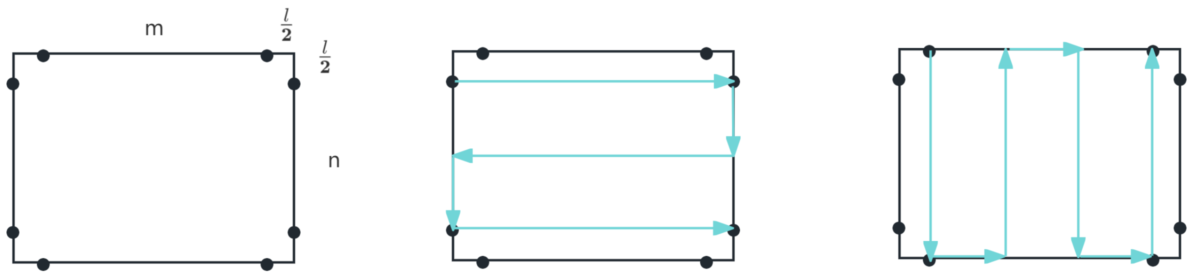

Figure 6. Distribution of entry/exit points and path coverage.

Under these conditions, the problem can be formulated as a UAV path optimization task that incorporates plot-specific coverage patterns. The primary objective is to plan a flight trajectory that minimizes total distance and energy consumption while ensuring full coverage and complying with operational constraints.

The objective function of the model is defined by Eq (4.1).

$$\min \sum_{i=1}^{y}\sum_{j=1}^{y}\sum_{v=1}^{8}\sum_{w=1}^{8} a_{ij}b_v^i c_w^i d(x_{qv}^i, x_{pw}^i) + \sum_{i=1}^{y}\sum_{v=1}^{8} b_v^i d(x_{pv}^i, x_{qv}^i). \tag{4.1}$$

Here, y is the total number of target points, including w departure points and $y-1$ plots. a_{ij} represents the UAV moving from plot i to plot j. b_v^i indicates that the UAV covers plot i along the v-th path, and c_w^i indicates that the UAV covers plot j along the w-th path. $d(x_{qv}^i, x_{pw}^i)$ denotes the Euclidean distance between the endpoint of plot i along the v-th path and the start point of plot j along the w-th path. $d(x_{pv}^i, x_{qv}^i)$ represents the flight distance when the UAV covers plot i along the v-th path.

The constraints are given by Eqs (4.2)–(4.4).

$$\sum_{i=1}^{y} a_{ij} = 1, \quad \sum_{j=1}^{y} a_{ij} = 1, \qquad i,j = 1,2,\cdots,y, \tag{4.2}$$

$$\sum_{v=1}^{8} b_v^i = 1, \quad \sum_{w=1}^{8} c_w^i = 1, \qquad i = 1,2,\cdots,8, \tag{4.3}$$

$$u_i - u_j + y a_{ij} \le y - 1 \quad (2 \le i \neq j \le y), \tag{4.4}$$

where u_i is an auxiliary variable representing the visitation order of plot i in the UAV tour (taking values $2, 3, \ldots, y$). This constraint ensures that if the UAV moves from plot i to plot j, the visitation order increases accordingly, thereby eliminating potential subtours. In Eqs (4.2) and (4.3), a variable value of 1 indicates "yes", while 0 indicates "no". Specifically, Eq (4.2) constrains each plot to be visited only once; Eq (4.3) ensures that each plot is covered using only one path; and the constraint in Eq (4.4) is used to eliminate subtours while preserving the overall main tour structure.

The simulated plots were generated as shown in Figure 7. The plot generation area is a 150×150 rectangular region, and each plot is rectangular. The vertex coordinates of each plot are listed in Table 7.

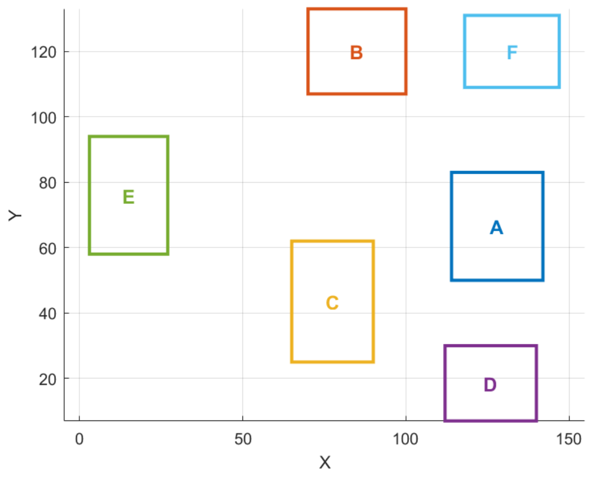

Figure 7. Plot distribution map.

Table 7. Vertex coordinates of the plot.

Plot ID	Vertex coordinates of the plot	Centroid coordinates of the plot
A	(114, 50), (114, 83), (142, 50), (142, 83)	(128, 66.5)
B	(70, 107), (70, 133), (100, 107), (100, 133)	(85, 120)
C	(65, 25), (65, 62), (90, 25), (90, 62)	(77.5, 43.5)
D	(112, 7), (112, 30), (140, 7), (140, 30)	(126, 18.5)
E	(3, 58), (3, 94), (27, 58), (27, 94)	(15, 76)
F	(118, 109), (118, 131), (147, 109), (147, 131)	(132.5, 120)

4.3.2. Path optimization results

STAOA was applied to the target problem to validate its effectiveness. Since the algorithm is designed for continuous problems, discrete variables were encoded into a continuous search space and decoded after optimization, enabling a transition from continuous to discrete solutions.

Its performance was compared with the PSO and GA, with parameters listed in Table 8. Results in Table 9 show that the STAOA outperforms both the PSO and GA in solution quality, convergence speed, and stability.

Table 8. Algorithm parameter settings.

Algorithm	Parameter settings
STAOA	$\tau_{\max}$=0.2,N=30,$maxt$=200,l=5
PSO	$c1$=2,$c2$=2,w=0.8,N=30,$maxt$=200,l=5
GA	c=0.8,m=0.3,N=30,$maxt$=200,l=5

Table 9. Comparison of algorithm performance.

Algorithm	Plot traversal sequence	Shortest path distance
STAOA	C(P5) → D(P1)→ A(P1) → F(P6)→ B(P6)→ E(P6)	1130.62 m
PSO	E(P5)→ C(P5)→ B(P6)→ F(P5)→ A(P2)→ D(P2)	1566.39 m
GA	D(P1) → A(P1) → F(P1) → B(P6) → C(P2) → E(P6)	1200.27 m

In the parameter settings, N denotes the number of particles in the population, max t represents the maximum number of iterations, and l is the spraying radius of the UAV. The experimental results in Table 8 indicate that the STAOA can effectively find more optimal flight paths. The UAV flight route maps generated by the three algorithms are shown in Figure 8, and the convergence curves are presented in Figure 9.

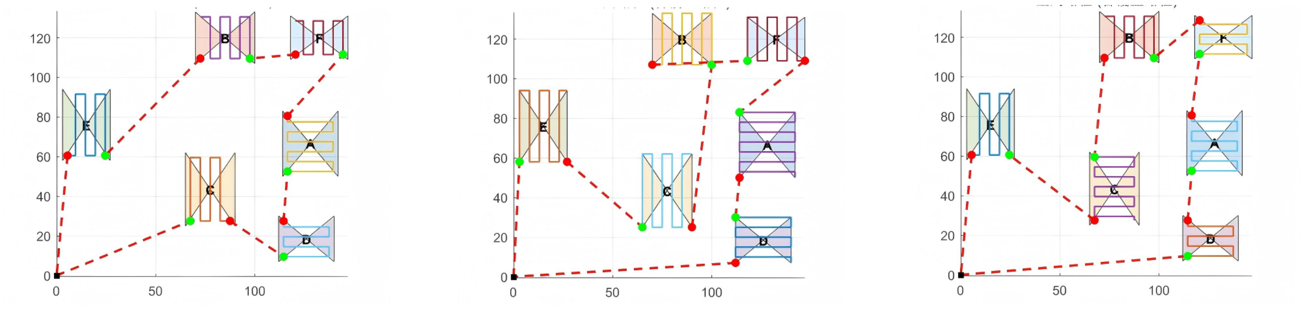

Figure 8. STAOA, PSO, GA trajectory plots.

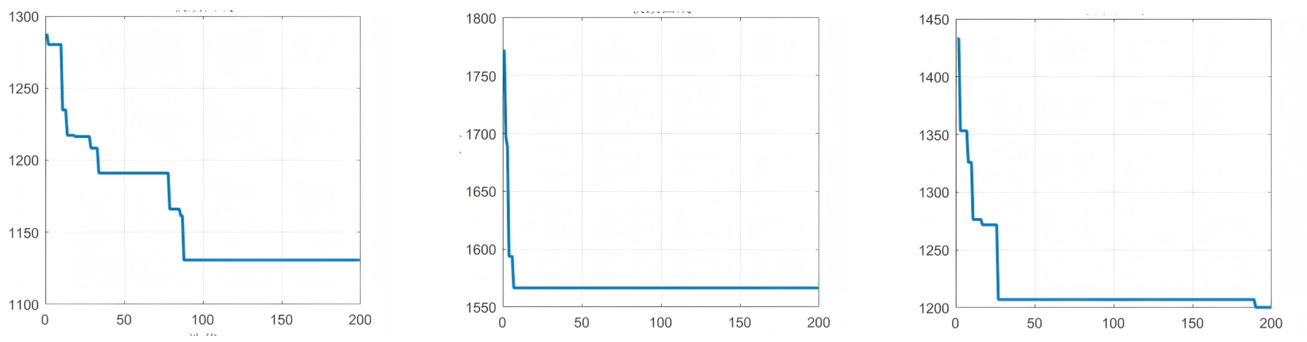

Figure 9. STAOA, PSO, GA convergence plots.

As shown in Figure 9, the STAOA converged to the global optimum by the 88th generation, demonstrating rapid convergence and strong global search capability. Although the PSO algorithm reached its best solution as early as the 7th generation, the quality of the solution was relatively low. The GA did not converge until the 190th generation, exhibiting the slowest convergence, and its final solution was inferior to that of the STAOA. These results indicate that the STAOA outperforms both the PSO and GA in terms of convergence speed, solution quality, and stability, providing a reliable approach for UAV spraying path optimization.

The experimental results presented in Table 9 show that the STAOA converged to the global optimum by the 90th generation, and its optimal solution quality was significantly better than that of the other two algorithms, demonstrating high convergence efficiency and strong global search capability. Although the PSO converged by the 14th generation, its optimal solution quality was relatively poor, failing to identify the best spraying path.The GA converged by the 90th generation, but its final optimal solution was slightly inferior to that of the STAOA. Overall, the STAOA exhibits clear advantages in terms of convergence speed, solution quality, and algorithm stability, making it particularly suitable for large-scale plot path optimization problems.

In this section, the proposed STAOA is applied to UAV path planning for crop protection. Comparisons with benchmark algorithms indicate that the STAOA achieves faster convergence and shorter spraying paths, supports multi-solution design, and satisfies operational constraints.

5. Discussion

The experimental results demonstrate that the STAOA achieves superior average performance and lower variability on most benchmark and CEC2019 test functions, indicating that it attains a better balance between global exploration and local exploitation. The improved search mechanism enhances exploration in the early stages and gradually strengthens local refinement in the later stages, thereby accelerating convergence and improving solution accuracy.

The STAOA performs stably on both unimodal and multimodal functions, and in particular, it effectively avoids premature convergence on multimodal problems, reflecting its capability of maintaining population diversity. However, its advantage is less pronounced on some highly complex functions, suggesting that its performance in extremely complicated problems is still influenced by parameter settings and search strategies.

In the UAV crop-protection path planning problem, the STAOA is able to generate shorter and smoother spraying paths while satisfying operational constraints, and it supports multi-solution design, which improves operational efficiency and flexibility. Its main limitations lie in the relatively large number of parameters, sensitivity to problem scale and dimensionality, and insufficient adaptability to dynamic environments.

6. Conclusions

This study proposed an improved algorithm, STAOA, and validated its performance using standard benchmark functions, the CEC2019 test suite, and a UAV crop-protection path planning application.

The results show that the STAOA achieves higher optimization accuracy, faster convergence speed, and better stability on most test problems, outperforming the compared algorithms overall. In the UAV path planning application, the STAOA can generate shorter and more efficient paths under operational constraints and supports multiple alternative solutions, demonstrating strong practical applicability.

In summary, the STAOA is an efficient and stable intelligent optimization algorithm with significant practical potential. Future work will focus on developing parameter self-adaptation mechanisms, improving performance in dynamic environments, and extending the algorithm to multi-UAV cooperative scenarios.

Author contributions

Renyun Liu: Conceptualization, resources, writing–editing draft, Helei Kang: Conceptualization, methodology, supervision, funding acquisition; Rui Bao: Validation; Siyi Gong: Data curation; Yifei Yao: Formal analysis; Yang Wu: Conceptualization, methodology, validation, data curation, formal analysis, writing–original draft, writing–editing draft. All authors have read and agreed to the published version of the manuscript.

Use of Generative-AI tools declaration

The authors hereby state that Artificial Intelligence (AI) tools were used during the writing process of this manuscript to assist with certain tasks.

AI tools used: Chat GPT. After composition in English, this manuscript was subjected to AI-assisted checking to prevent grammatical errors. In the course of writing, artificial intelligence was employed to correct grammatical structures and enhance the readability of protracted sentences.

Acknowledgments

This work was supported in part by the Jilin Scientific and Technological Development Program, China (Grant No. 20250201080GX).

Conflict of interest

The authors declare no conflict of interest.

References

1. J. Tang, G. Liu, Q. T. Pan, A review on representative swarm intelligence algorithms for solving optimization problems: applications and trends, *IEEE-CAA J. Automatic.*, **8** (2021), 1627–1643. http://dx.doi.org/10.1109/JAS.2021.1004129
2. M. Jain, V. Saihjpal, N. Singh, S. B. Singh, An overview of variants and advancements of PSO algorithm, *Appl. Sci.*, **12** (2022), 8392. http://dx.doi.org/10.3390/app12178392
3. L. Y. Wang, Q. J. Cao, Z. X. Zhang, S. Mirjalili, W. G. Zhao, Artificial rabbits optimization: A new bio-inspired meta-heuristic algorithm for solving engineering optimization problems, *Eng. Appl. Artif. Intel.*, **114** (2022), 105082. http://dx.doi.org/10.1016/j.engappai.2022.105082
4. A. H. Gandomi, X. S. Yang, S. Talatahari, A. H. Alavi, Metaheuristic algorithms in modeling and optimization, In: *Metaheuristic applications in structures and infrastructures*, Amsterdam: Elsevier, 2013, 1–24. http://dx.doi.org/10.1016/C2011-0-08778-1
5. R. Y. Liu, N. Zhou, Y. F. Yao, F. H. Yu, An aphid inspired metaheuristic optimization algorithm and its application to engineering, *Sci. Rep.*, **12** (2022), 18064. http://dx.doi.org/10.1038/s41598-022-22170-8
6. S. S. Rezk, K. S. Selim, Metaheuristic-based ensemble learning: an extensive review of methods and applications, *Neural Comput. Applic.*, **36** (2024), 17931–17959. http://dx.doi.org/10.1007/s00521-024-10203-4
7. R. Rani, S. Jain, H. Garg, A review of nature-inspired algorithms on single-objective optimization problems from 2019 to 2023, *Artif. Intell. Rev.*, **57** (2024), 126. http://dx.doi.org/10.1007/s10462-024-10747-w
8. P. K. Mandal, A review of classical methods and Nature-Inspired Algorithms (NIAs) for optimization problems, *Results in Control and Optimization*, **13** (2023), 100315. http://dx.doi.org/10.1016/j.rico.2023.100315

9. W. Chu, X. G. Gao, S. Sorooshian, A new evolutionary search strategy for global optimization of high-dimensional problems, *Inform. Sciences*, **181** (2011), 4909–4927. http://dx.doi.org/10.1016/j.ins.2011.06.024

10. L. Chen, H. Q. Lu, H. W. Li, G. J. Wang, L. Chen, Dimension-by-dimension enhanced cuckoo search algorithm for global optimization, *Soft Comput.*, **23** (2019), 11297–11312. http://dx.doi.org/10.1007/s00500-019-03844-4

11. Z. Y. Meng, Y. X. Zhong, G. J. Mao, Y. Liang, PSO-sono: A novel PSO variant for single-objective numerical optimization, *Inform. Sciences*, **586** (2022), 176–191. http://dx.doi.org/10.1016/j.ins.2021.11.076

12. H. L. Kang, R. Y. Liu, Y. F. Yao, F. H. Yu, Improved Harris hawks optimization for non-convex function optimization and design optimization problems, *Math. Comput. Simulat.*, **204** (2023), 619–639. http://dx.doi.org/10.1016/j.matcom.2022.09.010

13. A. Mokabberi, M. Golsorkhtabaramiri, R. A. Varzi, Proposing an advanced trending-based grey wolf optimizer for single-objective optimization problems, *2024 20th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP)*, Babol, Iran, 2024, 1–7. http://dx.doi.org/10.1109/AISP61396.2024.10475287

14. O. R. Adegboye, A. K. Feda, Improved exponential distribution optimizer: enhancing global numerical optimization problem solving and optimizing machine learning parameters, *Cluster Comput.*, **28** (2025), 128. http://dx.doi.org/10.1007/s10586-024-04753-4

15. F. Wang, X. J. Wang, S. L. Sun, A reinforcement learning level-based particle swarm optimization algorithm for large-scale optimization, *Inform. Sciences*, **602** (2022), 298–312. http://dx.doi.org/10.1016/j.ins.2022.04.053

16. X. M. Zhu, J. S. Zhang, C. C. Jia, Y. Liu, M. S. Fu, A hybrid black-winged kite algorithm with PSO and differential mutation for superior global optimization and engineering applications, *Biomimetics,* **10** (2025), 236. http://dx.doi.org/10.3390/biomimetics10040236

17. Y. Wang, J. Zhao, X. W. Guo, A multi-strategy enhanced chernobyl disaster optimizer for global optimization and engineering design problems, *2025 37th Chinese Control and Decision Conference (CCDC)*, Xiamen, China, 2025, 178–183. http://dx.doi.org/10.1109/CCDC65474.2025.11090316

18. C. L. Huang, M. J. Wang, H. A. Asghar, Z. L. Wang, H. L. Chen, Q-learning enhanced differential evolution for feature selection in high-dimensional medical data analysis, *J. King Saud Univ. Comput. Inf. Sci.*, **37** (2025), 280. http://dx.doi.org/10.1007/s44443-025-00303-z

19. H. T. Sadeeq, A. M. Abdulazeez, Metaheuristics: A review of algorithms, *International Journal of Online and Biomedical Engineering*, **19** (2023), 142–164. http://dx.doi.org/10.3991/ijoe.v19i09.39683

20. M. S. Shaikh, S. Raj, G. Z. Zheng, S. L. Xie, C. Wang, X. Q. Dong, et al., Applications, classifications, and challenges: a comprehensive evaluation of recently developed metaheuristics for search and analysis, *Artif. Intell. Rev.*, **58** (2025), 390. http://dx.doi.org/10.1007/s10462-025-11377-6

21. A. E. Piotrowska, A. P. Piotrowski, Improving scale parameters in successful-history-based adaptive differential evolution algorithms, *Appl. Soft Comput.*, **187** (2026), 114288. http://dx.doi.org/10.1016/j.asoc.2025.114288

22. H. Pu, Q. X. Zeng, T. R. Song, P. Schonfeld, G. H. Wang, Wei Li, et al., A hybrid proximal policy optimization and particle swarm algorithm for highway alignment optimization, *Adv. Eng. Inform.*, **69** (2026), 103959. http://dx.doi.org/10.1016/j.aei.2025.103959

23. P. Y. Wei, C. Y. Fan, X. W. Yang, X. Chen, J. H. Gan, X. Deng, et al., HOES: an efficient multi-evolutionary expert system for deep learning model optimization in time series prediction, *Sci. Rep.*, **16** (2026), 527. http://dx.doi.org/10.1038/s41598-025-30014-4

24. L. Abualigah, M. A. Elaziz, A. M. Khasawneh, M. Alshinwan, R. A. Ibrahim, M. A. A. Al-Qaness, et al., Meta-heuristic optimization algorithms for solving real-world mechanical engineering design problems: a comprehensive survey, applications, comparative analysis, and results, *Neural Comput. Applic.*, **34** (2022), 4081–4110. http://dx.doi.org/10.1007/s00521-021-06747-4

25. C. Wang, S. Y. Zhang, T. H. Ma, Y. T. Xiao, M. Z. Q. Chen, L. Wang, Swarm intelligence: A survey of model classification and applications, *Chinese J. Aeronaut.*, **38** (2025), 102982. http://dx.doi.org/10.1016/j.cja.2024.03.019

26. M. I. Ghazaan, A. S. Oshnari, A. S. Oshnari, A novel adaptive optimization scheme for advancing metaheuristics and global optimization, *Swarm Evol. Comput.*, **91** (2024), 101779. http://dx.doi.org/10.1016/j.swevo.2024.101779

27. H. Xu, D. X. Yu, Z. Wang, K. H. Cheong, C. L. P. Chen, Nonsingular predefined time adaptive dynamic surface control for quantized nonlinear systems, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, **54** (2024), 5567–5579. http://dx.doi.org/10.1109/TSMC.2024.3407150

28. C. He, J. C. Xing, J. L. Li, Q. L. Yang, R. H. Wang, A new wavelet thresholding function based on hyperbolic tangent function, *Math. Probl. Eng.*, **2015** (2015), 528656. http://dx.doi.org/10.1155/2015/528656

29. A. Vanderschueren, C. De Vleeschouwer, Are straight-through gradients and soft-thresholding all you need for sparse training, *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Waikoloa, HI, USA, 2023, 3797–3806. http://dx.doi.org/10.1109/WACV56688.2023.00380

30. P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, et al., Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, Technical Report, Nanyang Technological University, Singapore, 2005, 1–50.

31. J. Liang, B. Y. Qu, D. W. Gong, C. T. Yue, Problem definitions and evaluation criteria for the CEC 2019 competition on evolutionary computation for large-scale optimization, *2019 IEEE Congress on Evolutionary Computation (CEC)*, 2019, 1–8. http://dx.doi.org/10.13140/RG.2.2.33423.64164

32. R. Sowmya, M. Premkumar, P. Jangir, Newton-Raphson-based optimizer: A new population-based metaheuristic algorithm for continuous optimization problems, *Eng. Appl. Artif. Intel.*, **128** (2024), 107532. http://dx.doi.org/10.1016/j.engappai.2023.107532

33. M. Gafar, R. A. El-Sehiemy, H. M. Hasanien, A. Abaza, Optimal parameter estimation of three solar cell models using modified spotted hyena optimization, *J. Ambient Intell. Human Comput.*, **15** (2024), 361–372. http://dx.doi.org/10.1007/s12652-022-03896-9

34. M. Abdel-Basset, R. Mohamed, M. Abouhawwash, Crested Porcupine Optimizer: A new nature-inspired metaheuristic, *Knowl.-Based Syst.*, **284** (2024), 111257. http://dx.doi.org/10.1016/j.knosys.2023.111257
35. J. K. Xue, B. Shen, Dung beetle optimizer: A new meta-heuristic algorithm for global optimization, *J. Supercomput.*, **79** (2023), 7305–7336. http://dx.doi.org/10.1007/s11227-022-04959-6
36. S. J. Zhao, T. R. Zhang, S. L. Ma, M. C. Wang, Sea-horse optimizer: a novel nature-inspired meta-heuristic for global optimization problems, *Appl. Intell.*, **53** (2023), 11833–11860. http://dx.doi.org/10.1007/s10489-022-03994-3
37. G.-J. Lai, T. Li, B.-J. Shi, RRT-Based optimizer: A novel metaheuristic algorithm based on rapidly-exploring random trees algorithm, *IEEE Access*, **13** (2025), 42744–42776. http://dx.doi.org/10.1109/ACCESS.2025.3547537
38. A. Lambora, K. Gupta, K. Chopra, Genetic algorithm–A literature review, *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, Faridabad, India, 2019, 380–384. http://dx.doi.org/10.1109/COMITCon.2019.8862255
39. T. M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh, S. Mirjalili, Particle swarm optimization: A comprehensive survey, *IEEE Access*, **10** (2022), 10031–10061. http://dx.doi.org/10.1109/ACCESS.2022.3142859
40. S. Aggarwal, N. Kumar, Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges, *Comput. Commun.*, **149** (2020), 270–299. http://dx.doi.org/10.1016/j.comcom.2019.10.014
41. F. F. Du, Y. X. Ju, Z. Li, M. M. Wang, N. N. Xue, Optimization research on task assignment of plant protection UAVs based on ant colony algorithm, *Technology and Economy in Areas of Communications*, **22** (2020), 17–20. http://dx.doi.org/10.19348/j.cnki.issn1008-5696.2020.05.004