AIMS Mathematics, 11(5): 15163–15198.
DOI:10.3934/math.2026624
Received: 15 March 2026
Revised: 01 May 2026
Accepted: 12 May 2026
Published: 29 May 2026

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

Research article

Modified goat optimization algorithm with a population-dispersion-based adaptive exploration coefficient

Liwen Sun*

School of Computer Science and Technology, Anhui University of Technology, Ma'anshan 243032, China

* **Correspondence:** Email: lwsun@ahut.edu.cn.

Abstract: The goat optimization algorithm (GOA) is a novel nature-inspired optimization technique that has demonstrated promising performance across diverse numerical optimization problems. However, its reliance on a fixed exploration coefficient during iterations often leads to premature convergence to local optima and diminished solution accuracy. To address this limitation, we introduced the GOA-PD, a lightweight population-dispersion-driven adaptive strategy that adjusts the exploration coefficient online by integrating normalized dispersion feedback, iteration-dependent decay, and exponential moving average smoothing. Empirical evaluations on the 20-dimensional CEC2021 benchmark suite, together with statistical tests, ablation analysis, and parameter sensitivity analysis, indicate that the proposed method improves the baseline GOA in terms of average ranking, convergence behavior, and stability while maintaining competitive overall performance. In addition, a case studied on welded beam design further suggests that the GOA-PD can obtain feasible and competitive solutions to constrained engineering optimization problems with limited additional computational cost.

Keywords: goat optimization algorithm; population dispersion; adaptive exploration coefficient; local optima; swarm intelligence optimization
Mathematics Subject Classification: 68T20, 90C26

1. Introduction

Complex optimization problems arising in engineering design, machine learning, scheduling, and intelligent control are often characterized by nonlinearity, multimodality, nonconvexity, and strong variable interactions. Under such conditions, classical gradient-based methods may become ineffective because they depend heavily on derivative information and are prone to becoming trapped in local optima. In contrast, nature-inspired, swarm intelligence, and evolutionary algorithms [1–4]

provide derivative-free search mechanisms based on population interaction and stochastic updating. Representative methods such as particle swarm optimization (PSO) [5–7], the genetic algorithm (GA) [8, 9], the grey wolf optimizer (GWO) [10, 11], and ant colony optimization for a continuous domain (ACOR) [12, 13] have been widely adopted for complex optimization tasks. Nevertheless, despite their different inspirations and updating rules, these algorithms still face a common challenge: how to maintain an effective balance between global exploration and local exploitation throughout the search process.

Recent studies indicate that advances in metaheuristic optimization increasingly rely on better regulation of search behavior rather than the sole proposal of new biological or physical metaphors. Some studies have developed entirely new search paradigms, such as the Schrödinger optimizer [14] and secant optimization algorithm [15], to improve robustness, convergence, and search efficiency. Although effective, these methods involve a full redesign of the optimization framework and therefore do not directly address how to enhance an existing optimizer whose overall structure is already reasonable but whose parameter-control strategy remains inflexible. Other studies have improved existing optimizers through hybridization or auxiliary search mechanisms, such as the large-scale salp-based gray wolf optimization method [16] for feature selection and global optimization and the local escape operator based on salp-swarm quadratic interpolation [17] for large-scale optimization and feature selection. These studies have shown that meaningful performance gains can often be achieved by improving the regulation of exploration and exploitation. However, such improvements are frequently obtained by introducing multiple interacting modules, which may increase the algorithmic complexity and reduce the transparency of the core improvement mechanism.

Within this broader context, diversity control has become an important topic in swarm intelligence. Some studies have shown that parameter adaptation [18] based on the population state can help maintain the balance between exploration and exploitation more effectively than purely fixed parameter schedules. In general, diversity-related control strategies can be built from quantities such as pairwise distances, neighborhood structures, or fitness-distribution statistics. The common advantage of these approaches is that they use state feedback rather than the iteration count alone. However, many diversity metrics are computationally costly, sensitive to problem scaling, or tightly tied to a specific algorithmic framework. For an optimizer such as the goat optimization algorithm (GOA), whose main structure already contains exploration, exploitation, jumping, and rebirth operators, introducing an excessively complex diversity controller may blur the functional roles of the original operators and weaken the simplicity of the baseline design.

Against this background, this study focused on the GOA, a recently proposed swarm-based optimizer inspired by random foraging, directed movement, jumping behavior, and parasite-avoidance-based rebirth. The GOA provides a structured search framework, but its original parameter design still suffers from a critical limitation. In the GOA, the exploration step is controlled by a fixed coefficient, which renders the algorithm insufficiently adaptive throughout the search process. Therefore, the main issue is that the fixed exploration mechanism cannot effectively maintain a proper balance between global exploration and convergence during the optimization process. To address this limitation, this study proposes an improved variant, termed the GOA-PD. Instead of introducing a complicated auxiliary search module, the GOA-PD develops a lightweight diversity-control mechanism for the GOA. Specifically, it uses a normalized population-dispersion indicator, computed from the mean standard deviation of the population across dimensions after range normalization, to regulate the

exploration coefficient online. Its role is not to replace richer diversity metrics in general but to provide a simple, low-cost, and easily embeddable state-feedback variable that matches the original GOA framework. In addition, the GOA-PD combines this dispersion-feedback term with an iteration-dependent decay term and an exponential moving average smoothing strategy, allowing the exploration coefficient to respond to both the current search state and the optimization stage while reducing abrupt oscillations caused by instantaneous population changes. Importantly, the GOA-PD adjusts only the exploration coefficient while preserving the original exploitation, jumping, greedy selection, and rebirth operators of the GOA, thereby maintaining the structural simplicity and computational efficiency of the original framework while improving its ability to balance exploration and convergence.

Based on the above considerations, the contributions of this study can be summarized as follows: First, a normalized population-dispersion measure is introduced for the GOA as a lightweight diversity-control signal that characterizes the current global population state in a simple and low-cost manner. Second, a dispersion-driven adaptive exploration mechanism is developed by integrating population feedback, iteration decay, and exponential moving average (EMA) smoothing, so that the search process can better coordinate exploration and exploitation. Third, the proposed GOA-PD is evaluated on the 20-dimensional CEC2021 benchmark suite against several representative metaheuristic algorithms, together with convergence analysis, stability analysis, statistical significance tests, an ablation study, and parameter sensitivity analysis, to verify the effectiveness of the adaptive strategy and the overall competitiveness of the improved method. In addition, a welded beam design problem is introduced as a constrained engineering optimization case study to further examine the feasibility-handling behavior and practical applicability of the GOA-PD. This design further demonstrates that effective performance improvement can be achieved through lightweight state-feedback control without altering the original algorithmic structure, which may be beneficial for improving other swarm-based optimizers.

2. Biological background and mathematical model of the GOA

2.1. Biological background of the GOA

As one of the most adaptable herbivores, goats can adapt to rocky slopes and rugged environments, and are good at jumping and climbing in complex terrains. The following section introduces the unique framework provided by the biological characteristics of goats for the design of GOA optimization algorithms.

2.1.1. Random foraging and selective grazing

The foraging habits of goats are random. Before choosing grazing areas, goats randomly explore the surrounding environment. They tend to prefer highly nutritious plants and avoid harmful areas. This behavior is projected into the exploration of an optimization algorithm. Before determining the optimal grazing location, the population explores multiple candidate regions within the search space and gradually moves toward promising areas to approximate the optimal grazing region.

Studies on goats have shown that they can assess food resources and environmental risks. In the context of swarm intelligence optimization, this characteristic can be abstracted into a search mechanism that combines random exploration with movement toward promising regions. During the

optimization process, this mechanism helps the population balance between global exploration and local exploitation.

2.1.2. Jumping mechanism

Goats often live in the rugged and steep areas, and their superior jumping ability enables them to cross dangerous terrains, reach places that are difficult for other animals to access, and avoid threatening sites. Unlike many gregarious animals that move gradually, goats often combine short-range movements with occasional longer-distance jumps.

This jumping behavior is closely related to the need to escape local optima in the optimization algorithms. By incorporating the random jumping characteristics of goats, the GOA introduced a stochastic but controlled movement strategy that enables the solution to explore new regions while reducing the risk of stagnation.

2.1.3. Parasite avoidance mechanism

Goats instinctively avoid areas where parasites are highly concentrated or food sources are toxic or contaminated. This parasite-avoidance behavior serves as an instinctive protective mechanism that helps improve survival and maintain health and safety. In the GOA, this avoidance mechanism can be translated into a filtering strategy for relatively poor solutions, that is, discarding poor solutions and randomly regenerating new candidate solutions to improve search efficiency.

With this mechanism, the GOA periodically removes poorly performing individuals and injects new candidate solutions into the population, thereby enhancing the global search ability and reducing the risk of premature convergence to local optima.

2.2. Mathematical model of the GOA

Based on the above interpretation of the biological behavior of goats, this section constructs a mathematical model of the GOA. The algorithm adopts a phased search strategy that combines global exploration, local exploitation, jumping mechanisms, and parasite-avoidance mechanisms.

2.2.1. Population initialization

First, population $\mathbf{X}$ is initialized, the goat population size is set to $\mathbf{N}$, and the problem dimension is $\mathbf{d}$. Each individual X_i in the population is represented as a d-dimensional vector:

$$X = \{X_1, X_2, \cdots, X_N\}. \tag{2.1}$$

$$X_i = (x_{i1}, x_{i2}, \cdots, x_{id}). \tag{2.2}$$

The initial population of goats is randomly generated in a given search space, and its generation formula is:

$$X_i = LB + (UB - LB) \cdot \text{rand}(d). \tag{2.3}$$

Here, LB and UB denote the lower and upper bounds of the search space, respectively, and rand(d) generates a d-dimensional random vector with elements uniformly distributed in the interval $[0, 1]$. This initialization strategy enables the goat population to be randomly distributed over the feasible search space, thereby providing a basis for the subsequent exploration stage.

2.2.2. Exploration stage

In the exploration stage, individual goats move randomly in different directions based on the inspiration of foraging behavior. If an updated position exceeds the predefined search range, it is clipped to remain within the feasible boundary. The position update formula for an individual goat is as follows:

$$Y_i^t = \Pi_\Omega \left(X_i^t + \alpha \, \xi_i^t \odot (UB - LB) \right), \quad i = 1, \ldots, N, \qquad \xi_i^t \sim \mathcal{N}(0, I_d). \tag{2.4}$$

Here, Y_i^t denotes the intermediate position of the i-th goat after the exploration stage at iteration t, and X_i^t denotes its original position before exploration. Coefficient α is the exploration step-size parameter, which controls the exploration intensity. Random vector ξ_i^t is sampled from the standard multivariate normal distribution. The term ($\mathbf{UB} - \mathbf{LB}$) scales the perturbation according to the search range of each dimension. The operator $\Pi_\Omega(\cdot)$ projects the updated solution onto the feasible search domain $\Omega = [\mathbf{LB}, \mathbf{UB}]$, which is equivalent to boundary clipping in the implementation.

This formula enables goats to move randomly within a search space, thereby enhancing global exploration.

2.2.3. Exploitation stage

After completing a round of random exploration, each individual goat moves toward the current global best solution. This process simulates the tendency of goats to move toward a more promising grazing area. The position update formula in the exploitation stage is defined as

$$\mathbf{Z}_i^t = \Pi_\Omega(\mathbf{Y}_i^t + \beta \, (\mathbf{X}_{\text{best}}^t - \mathbf{Y}_i^t)), \tag{2.5}$$

where β is the exploitation coefficient, which indicates the extent to which an individual goat moves to the current best solution, and $\mathbf{X}_{\text{best}}^t$ represents the current global best individual position.

This formula guides individual goats toward promising regions of the search space and strengthens the local exploitation around the current best solution.

2.2.4. Jumping process

Inspired by the unique jumping ability of goats in rugged terrain, this study introduces this characteristic into the algorithm to help individuals escape from local optima. The mathematical formulation is as follows:

$$b_i^t \sim \text{Bernoulli}(p_{\text{jump}}), \qquad r_i^t \sim U(\{1, \ldots, N\} \setminus \{i\}), \tag{2.6}$$

$$\mathbf{W}_i^t = \Pi_\Omega\Big(\mathbf{Z}_i^t + b_i^t \, p_{\text{jump}} \left(\mathbf{X}_{r_i^t}^t - \mathbf{Z}_i^t \right) \Big). \tag{2.7}$$

Here, b_i^t is a Bernoulli random variable indicating whether the jump operation is activated for the i-th individual, where $b_i^t = 1$ means that jumping is performed and $b_i^t = 0$ means that the current position remains unchanged at this stage. The parameter p_{jump} denotes the jump control coefficient. In the current implementation, it is used both as the activation probability of the jump and as the proportional step size of jump displacement. The index r_i^t denotes a randomly selected target individual

satisfying $r_i^t \neq i$, and $X_{r_i^t}^t$ is the corresponding target position in the current population. Z_i^t represents the intermediate position after the exploitation stage and W_i^t denotes the position after the jumping stage.

This formula is intended to help individuals jump out of local optima. Similarly, jumping toward different target individuals improves randomness and expands the search region, helping reduce the risk of premature convergence.

2.2.5. Greedy selection

Before the parasite-avoidance rebirth step, a greedy parent-offspring selection is performed. This step compares the candidate solution obtained after the jumping stage with its parent solution at the beginning of the current iteration. If the new candidate has a better fitness value, it is retained; otherwise, the parent is preserved. In this way, the algorithm avoids unnecessary deterioration while still preserving the exploratory effect introduced by the preceding update stages.

$$\tilde{X}_i^{t+1} = \begin{cases} W_i^t, & f(W_i^t) < f(X_i^t), \\ X_i^t, & \text{otherwise.} \end{cases} \tag{2.8}$$

2.2.6. Parasite avoidance mechanism

Inspired by the parasite-avoidance behavior of goats, the GOA periodically eliminates poorly performing individuals and regenerates them randomly in a feasible search space. This mechanism helps maintain population diversity and reduces the risk of stagnation around the local optima.

Specifically, let

$$k = \lfloor \rho N \rfloor, \tag{2.9}$$

where ρ is the reset ratio and N is the population size. Let S^t denote the index set of the worst k individuals in the intermediate population $\{\tilde{X}_i^{t+1}\}_{i=1}^N$ ranked according to fitness. For each individual indexed by $i \in S^t$, its position is reinitialized by uniform random sampling over the feasible search domain $\Omega = [LB, UB]$, while the remaining individuals are kept unchanged. The rebirth operation is defined as

$$X_i^{t+1} = \begin{cases} LB + u_i^t \odot (UB - LB), & i \in S^t, \\ \tilde{X}_i^{t+1}, & i \notin S^t, \end{cases} \qquad u_i^t \sim U([0,1]^d). \tag{2.10}$$

Here, u_i^t is a d-dimensional uniform random vector, and $\odot$ denotes element-wise multiplication. This rebirth operation removes poor solutions and injects new candidate solutions into the population, thereby enhancing diversity and improving the robustness of the search process.

2.3. Algorithm implementation of the GOA

The pseudocode of the basic GOA is shown in Algorithm 1.

Algorithm 1: GOA

Input: Objective function $f(x)$; number of goats N; maximum iterations $T_{\max}$; search space bounds LB, UB; exploration coefficient α; exploitation coefficient β; jump probability p_{jump}; reset ratio ρ

Output: Best solution X_{best}

```
Initialize population X_i randomly in [LB, UB];
Evaluate fitness f(X_i) for all goats;
Identify X_best (goat with lowest f(x));
for t = 1 to T_max do
    Update positions using the exploration equation;
    Move goats toward X_best;
    Apply jump strategy for some goats;
    Perform greedy parent-offspring selection;
    Identify the worst-performing individuals and regenerate them uniformly within [LB, UB];
    Evaluate the fitness of the regenerated individuals and update X_best;
return X_best.
```

3. The proposed GOA-PD

In the basic GOA, the step size is fixed and remains unchanged throughout the entire optimization process, which makes it difficult to balance early global exploration and later fine local exploitation, thus leading to low search efficiency. When the step size is too large, the algorithm tends to oscillate around the optimal solution or even overshoot it, resulting in insufficient convergence accuracy. When the step size is too small, global exploration becomes slow and the algorithm is prone to premature convergence and being trapped in local optima.

To address this, this study proposes a method that dynamically adjusts the exploration coefficient in real time according to population dispersion, which enhances the global search and convergence performance while maintaining diversity, thereby solving the target optimization problem to a certain extent. The improved algorithm is referred to as the GOA-PD.

3.1. Measure of population dispersion

Similar to other heuristic algorithms, the GOA-PD needs to measure population diversity. To reduce the bias caused by different scales across dimensions, this study uses the standard deviation of each dimension together with search-range normalization to characterize the dispersion of individuals, thereby indirectly reflecting population dispersion. The specific steps are as follows:

(1) Calculate the standard deviation of the individual distribution by dimension.

(2) Normalize the search range of the corresponding dimensions.

(3) Average normalized standard deviations over all dimensions to obtain the overall dispersion indicator.

$$D = \frac{1}{d}\sum_{j=1}^{d}\frac{\text{std}(x_{1:N,j})}{\text{range}_j}. \tag{3.1}$$

$$\text{range}_j = UB_j - LB_j. \tag{3.2}$$

Here, D denotes the average normalized standard deviation over all dimensions and is used to measure the degree of population dispersion. A larger D indicates that the population is more dispersed, whereas a smaller D indicates that the population is more concentrated. $\mathrm{std}(x_{1:N,j})$ denotes the standard deviation of all individuals in the j-th dimension, and range_j denotes the search range of that dimension. Therefore, the standard deviation on each dimension is normalized before being averaged across the dimensions.

Generally speaking, the quantity defined in Eq (3.1) is closer to a normalized measure of population dispersion than to dispersion in the physical sense. It does not estimate the number of individuals per unit volume, but instead reflects how spread out the population is in the search space through the average normalized standard deviation across dimensions. According to the above metric, the exploration coefficient will be adaptively adjusted according to the value of **D** in the following subsection, to improve convergence performance while maintaining population diversity.

3.2. Dynamic coefficient adjustment strategy

Based on the overall dispersion indicator obtained above, this study maps D to the adaptive update rule of α, and adopts a linearly inverse mapping strategy so that α can be adjusted adaptively with changes in D, thereby improving search accuracy and convergence efficiency. The formula is as follows:

$$\alpha_t^{(D)} = \alpha_{\min} + (\alpha_{\max} - \alpha_{\min})(1 - \hat{D}), \tag{3.3}$$

$$\hat{\alpha}_t = \alpha_t^{(D)} \left(1 - \frac{t}{T_{\max}}\right), \tag{3.4}$$

$$\hat{D} = \mathrm{clip}(D, 0, 1). \tag{3.5}$$

Here, $\alpha_t^{(D)}$ in Eq (3.3) denotes the dispersion-based exploration coefficient at iteration t, while $\hat{\alpha}_t$ in Eq (3.4) denotes the candidate adaptive exploration coefficient after introducing the time-decay term. In Eq (3.3), $\alpha_{\max}$ and $\alpha_{\min}$ denote the upper and lower bounds of the dispersion-based coefficient $\alpha_t^{(D)}$. $\hat{D}$ is the clipped form of the dispersion indicator D, and t and $T_{\max}$ denote the current and maximum iteration numbers, respectively. The operator $\mathrm{clip}(\cdot)$ truncates a real value to the interval $[0, 1]$.

According to Eq (3.3), when D is large, the population is relatively dispersed, $\hat{D}$ is large, and $(1 - \hat{D})$ becomes small; therefore, $\alpha_t^{(D)}$ decreases accordingly. This helps avoid unnecessary additional wandering and promotes information aggregation. When D is small, the population is relatively concentrated, $(1 - \hat{D})$ increases, and $\alpha_t^{(D)}$ increases. As a result, the exploration step is moderately enlarged, which helps enhance the global search ability and reduces the risk of being trapped in the local optima.

The linear inverse mapping in Eq (3.3) was adopted not as a theoretically optimal control law, but as a simple monotonic and bounded state-feedback rule. Its main advantage is that it preserves interpretability, introduces no extra shape parameters, and can be embedded into the original GOA framework with very low additional complexity.

On the basis of Eq (3.4), the time-decay factor $\left(1 - \frac{t}{T_{\max}}\right)$ is introduced. In the early stage of the algorithm, this factor is relatively large, so the adaptive coefficient remains more responsive to population dispersion and can maintain stronger exploration ability. As t increases, the factor

gradually decreases, which suppresses excessive expansion of the exploration step in the later stage and improves convergence stability. In this way, the adaptive coefficient remains sensitive to the population state while gradually becoming more conservative over time. Therefore, after the time-decay term is introduced, the candidate coefficient $\hat{\alpha}_t$ becomes progressively smaller with the iteration count, so $\alpha_{\min}$ should be interpreted as the lower bound of $\alpha_t^{(D)}$ in Eq (3.3). This design combines state-dependent feedback from population dispersion with stage-dependent regulation from iteration decay, while keeping the adaptive rule simple and low-complexity.

3.3. EMA smoothing

In the practical implementation, the candidate adaptive coefficient $\hat{\alpha}_t$ is further smoothed by an EMA strategy to reduce abrupt oscillations caused by instantaneous population-state changes. The resulting α_t is taken as the final exploration coefficient used in the GOA-PD.

$$\alpha_t = \begin{cases} \hat{\alpha}_t, & t = 1, \\ \eta\alpha_{t-1} + (1-\eta)\hat{\alpha}_t, & t \geq 2, \end{cases} \tag{3.6}$$

where $\hat{\alpha}_t$ denotes the candidate adaptive coefficient computed by Eq (3.4), and $\eta \in (0, 1)$ is the EMA smoothing factor.

Accordingly, in the GOA-PD, the fixed exploration coefficient α in Eq (2.4) is replaced by the smoothed adaptive coefficient α_t, while the exploitation, jumping, greedy selection, and rebirth stages remain structurally unchanged. In this way, the GOA-PD adjusts only the exploration radius in Eq (3.7), whereas the subsequent exploitation and jump operators retain their original roles through the fixed coefficients β and p_{jump}. Therefore, the adaptive update of α_t changes only the intensity of the exploration stage, without altering the functional balance of the later operators. The exploration step of the GOA-PD can be written as:

$$Y_i^t = \Pi_\Omega(X_i^t + \alpha_t \xi_i^t \odot (UB - LB)), \quad i = 1, \ldots, N, \qquad \xi_i^t \sim \mathcal{N}(0, I_d). \tag{3.7}$$

3.4. Overall workflow of the GOA-PD

To provide a clearer overview of how the proposed mechanism is embedded into the GOA framework, the overall workflow of the GOA-PD is illustrated in Figure 1. As shown in the figure, the PD module first computes the population dispersion, then adaptively updates the exploration coefficient through dispersion feedback, iteration-dependent decay, and EMA smoothing. The obtained coefficient is subsequently used in the exploration stage, while the exploitation, jumping, greedy selection, and parasite-avoidance rebirth stages remain structurally unchanged.

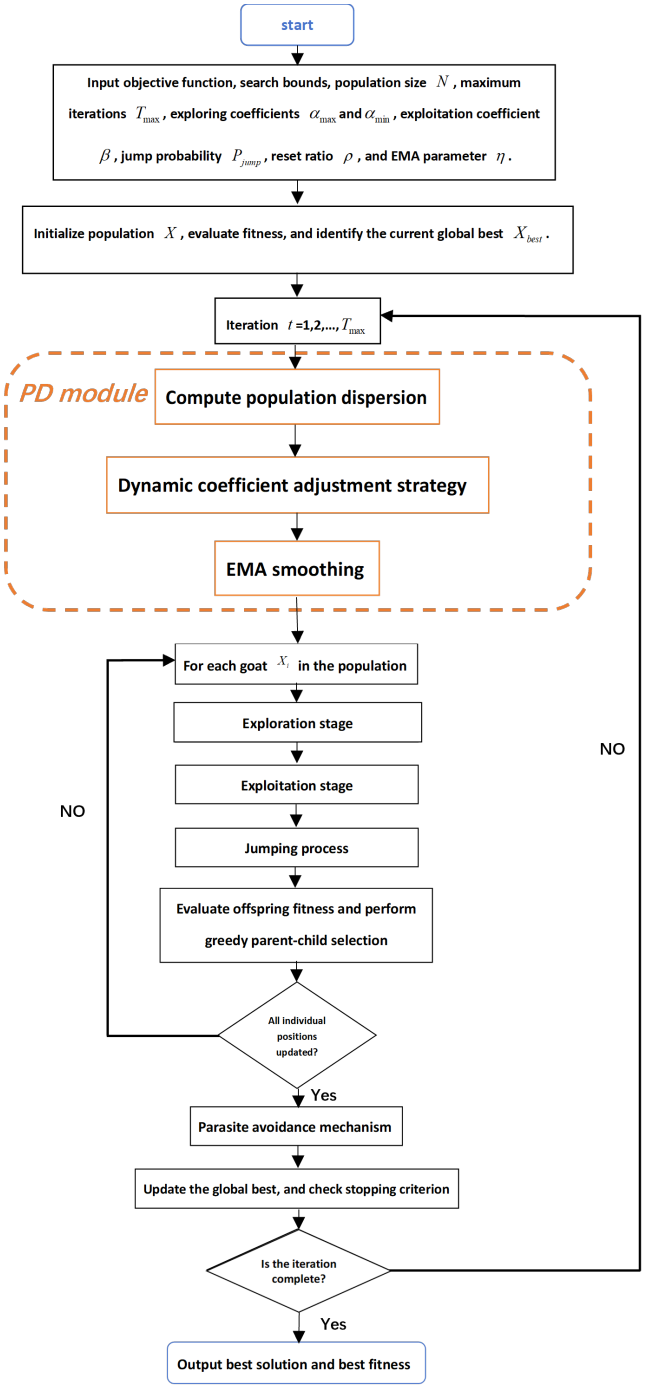


Figure 1. Overall flowchart of the GOA-PD.

3.5. Algorithm implementation of the GOA-PD

The pseudocode of the GOA-PD is shown in Algorithm 2.

Algorithm 2: GOA-PD

Input: Objective function $f(x)$; number of goats N; maximum iterations $T_{\max}$; search bounds LB, UB; exploration coefficients $\alpha_{\min}, \alpha_{\max}$; exploitation coefficient β; jump probability p_{jump}; EMA factor η; reset ratio ρ

Output: Best solution X_{best}

Initialize population X_i randomly in $[LB, UB]$;
Evaluate fitness $f(X_i)$ for all goats;
Identify X_{best} (goat with the lowest $f(x)$);
for $t = 1$ *to* $T_{\max}$ **do**
    Compute population dispersion D, obtain $\hat{D}$, compute $\hat{\alpha}_t$, and update α_t using EMA smoothing;
    Update positions using the exploration equation with α_t;
    Move goats toward X_{best};
    Apply jump strategy to some goats;
    Perform greedy parent-offspring selection;
    Identify the worst-performing individuals and regenerate them uniformly within $[LB, UB]$;
    Evaluate the fitness of the regenerated individuals and update X_{best};
return X_{best}.

3.6. Complexity analysis

The computational cost of the GOA-PD in each iteration mainly consists of four parts: population-dispersion computation, position updating, fitness evaluation, and worst-solution rebirth. Let N denote the population size, d the problem dimension, $T_{\max}$ the maximum number of iterations, C_f the cost of one objective-function evaluation, and ρ the reset ratio.

The population-dispersion statistic requires $O(Nd)$ time per iteration. The exploration, exploitation, and jump operations update all individuals in the d-dimensional space, which also requires $O(Nd)$. Evaluating the updated population requires N objective-function evaluations, leading to a cost of $O(NC_f)$. In addition, the parasite-avoidance mechanism requires $O(N \log N)$ time to identify the worst individuals and $O(\rho NC_f)$ time to reevaluate the regenerated ones.

Therefore, the per-iteration cost of the GOA-PD is

$$T_{\text{iter}} = O\left(Nd + (1+\rho)NC_f + N\log N\right). \tag{3.8}$$

Accordingly, over $T_{\max}$ iterations, the total time complexity is

$$T_{\text{total}} = O\left(T_{\max}\left[Nd + (1+\rho)NC_f + N\log N\right]\right). \tag{3.9}$$

When the objective-function evaluation dominates the internal vector operations, the above expression can be further simplified as

$$T_{\text{total}} \approx O\Big(T_{\max}(1+\rho)NC_f\Big). \tag{3.10}$$

The space complexity of the GOA-PD is $O(Nd)$. Compared with the basic GOA, the GOA-PD introduces an additional $O(Nd)$ computation for population-dispersion statistics, while the asymptotic

order of time complexity remains unchanged.

4. Experimental setup

4.1. CEC2021 benchmark suite

To evaluate the optimization performance of the GOA-PD, the CEC2021 benchmark suite was adopted in this study, as summarized in Table 1. Compared with conventional low-complexity benchmark functions, the CEC2021 test suite provides more challenging search landscapes through shift, rotation, hybridization, and composition mechanisms, making it more suitable for assessing the robustness, search accuracy, and generalization ability of metaheuristic algorithms.

Table 1. Summary of the CEC2021 benchmark suite used in this study.

Function	Category	Function name	Search range	Optimum
F1	Transformed basic	Shifted and Rotated Bent Cigar	$[-100, 100]^D$	100
F2	Transformed basic	Shifted and Rotated Schwefel	$[-100, 100]^D$	1100
F3	Transformed basic	Shifted and Rotated Lunacek bi-Rastrigin	$[-100, 100]^D$	700
F4	Transformed basic	Expanded Rosenbrock plus Griewangk	$[-100, 100]^D$	1900
F5	Hybrid	Hybrid Function 1	$[-100, 100]^D$	1700
F6	Hybrid	Hybrid Function 2	$[-100, 100]^D$	1600
F7	Hybrid	Hybrid Function 3	$[-100, 100]^D$	2100
F8	Composition	Composition Function 1	$[-100, 100]^D$	2200
F9	Composition	Composition Function 2	$[-100, 100]^D$	2400
F10	Composition	Composition Function 3	$[-100, 100]^D$	2500

In this work, the 20-dimensional version of the CEC2021 benchmark suite was used. A total of ten benchmark functions, denoted by F1–F10, were considered. Specifically, F1–F4 are transformed basic functions, F5–F7 are hybrid functions, and F8–F10 are composition functions. These functions cover various optimization difficulties, including ill-conditioning, non-separability, multimodality, and complicated local-global interactions, thereby providing a comprehensive test environment for evaluating the proposed algorithm.

4.2. Engineering constrained optimization case: welded beam design problem

To further examine the practical relevance and engineering applicability of the proposed GOA-PD, a classical engineering constrained optimization benchmark, namely, the welded beam design problem, was introduced as an additional case study. The welded beam design problem is one of the most widely used benchmark problems in constrained engineering optimization and has been adopted in many studies [19–21] to evaluate the ability of optimization algorithms to handle nonlinear objectives, nonlinear constraints, and strict feasibility requirements. This case study is not intended to represent a full industrial deployment scenario, but rather to provide a physically meaningful constrained engineering benchmark for evaluating the feasibility-handling ability and practical optimization behavior of the proposed GOA-PD.

In this study, the standard welded beam design problem originally reported in the constrained optimization literature by Coello [22] was adopted. The objective of this problem is to minimize

the fabrication cost of a welded beam while satisfying several mechanical and geometric constraints, including shear stress, bending stress, buckling load, end deflection, and design-size restrictions. Compared with unconstrained numerical benchmark functions, this problem provides a physically meaningful engineering design scenario and is therefore suitable for evaluating whether the proposed adaptive exploration mechanism can be transferred from synthetic numerical benchmarks to constrained engineering optimization.

The design vector of the problem of the welded beam is defined as:

$$x = [x_1, x_2, x_3, x_4] = [h, l, t, b], \tag{4.1}$$

where h is the thickness of the weld, l is the length of the weld, t is the height of the beam, and b is the thickness of the beam. The schematic meaning of these variables follows the standard welded beam design formulation commonly used in constrained engineering optimization studies.

The objective is to minimize the total fabrication cost of the welded beam, which consists of the welding cost and the material cost of the beam. The objective function is formulated as

$$\min f(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14 + x_2). \tag{4.2}$$

In this formulation, the first term represents the welding-related cost, whereas the second term represents the material-related cost of the beam. The constant 14 denotes the fixed beam length used in the standard welded beam design model.

The welded beam design problem is constrained by shear stress, bending stress, geometric restriction, fabrication cost, minimum weld thickness, end deflection, and buckling load. The constraints are expressed as follows:

$$\begin{aligned}
&g_1(x) = \tau(x) - \tau_{\max} \le 0,\\
&g_2(x) = \sigma(x) - \sigma_{\max} \le 0,\\
&g_3(x) = x_1 - x_4 \le 0,\\
&g_4(x) = 0.10471x_1^2 + 0.04811x_3x_4(14 + x_2) - 5 \le 0,\\
&g_5(x) = 0.125 - x_1 \le 0,\\
&g_6(x) = \delta(x) - \delta_{\max} \le 0,\\
&g_7(x) = P - P_c(x) \le 0.
\end{aligned} \tag{4.3}$$

Here, $\tau(x)$ is the shear stress, $\sigma(x)$ is the bending stress, $\delta(x)$ is the end deflection, and $P_c(x)$ is the critical buckling load. The corresponding allowable limits and physical constants are set as

$$\begin{aligned}
\tau_{\max} &= 13,600,\\
\sigma_{\max} &= 30,000,\\
\delta_{\max} &= 0.25,\\
P &= 6000,\\
L &= 14,\\
E &= 30 \times 10^6,\\
G &= 12 \times 10^6.
\end{aligned} \tag{4.4}$$

The shear stress is calculated by

$$\tau(x) = \sqrt{(\tau')^2 + \frac{2\tau'\tau'' x_2}{2R} + (\tau'')^2}, \tag{4.5}$$

where

$$\begin{aligned} \tau' &= \frac{P}{\sqrt{2}x_1 x_2}, \\ \tau'' &= \frac{MR}{J}, \\ M &= P\left(L + \frac{x_2}{2}\right), \\ R &= \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}, \\ J &= 2\sqrt{2}x_1 x_2 \left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]. \end{aligned} \tag{4.6}$$

The bending stress, end deflection, and critical buckling load are defined as

$$\begin{aligned} \sigma(x) &= \frac{6PL}{x_4 x_3^2}, \\ \delta(x) &= \frac{4PL^3}{E x_3^3 x_4}, \\ P_c(x) &= \frac{4.013E\sqrt{x_3^2 x_4^6/36}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right). \end{aligned} \tag{4.7}$$

The four design variables are bounded as follows:

$$\begin{aligned} &0.1 \le x_1 \le 2.0, \\ &0.1 \le x_2 \le 10.0, \\ &0.1 \le x_3 \le 10.0, \\ &0.1 \le x_4 \le 2.0. \end{aligned} \tag{4.8}$$

For fair comparison, the same exterior penalty-based constraint-handling strategy was applied to all compared algorithms in the welded beam design problem. A candidate solution was regarded as feasible only when the maximum positive constraint violation was not larger than the feasibility tolerance. Specifically, the penalized fitness value used for optimization was defined as

$$F(x) = f(x) + \lambda \sum_{k=1}^{7} \max(0, g_k(x))^2, \tag{4.9}$$

where $F(x)$ denotes the penalized fitness value, $f(x)$ is the original fabrication cost, $g_k(x)$ is the k-th inequality constraint, and λ is the penalty coefficient. In the implementation, λ was set to 10^8, and the feasibility tolerance was set to 10^{-6}. Therefore, infeasible candidate solutions were penalized according to the squared magnitude of their constraint violations, whereas feasible solutions were evaluated directly by their original objective costs.

The corresponding numerical results and feasibility analysis are presented in Section 5.7.

4.3. Compared algorithms

To provide a broader evaluation of the proposed method, the GOA-PD was compared with six representative metaheuristic algorithms, namely, the original GOA, ACOR, PSO, GWO, differential evolution (DE), and GA. Among them, the GOA was included as the direct baseline to verify the contribution of the proposed population-dispersion-based adaptive exploration strategy, while the other algorithms were used as representative comparison methods from different optimization frameworks.

The same set of compared algorithms was used for both the CEC2021 benchmark suite and the welded beam design case study. This setting allows the proposed method to be examined under a consistent comparison framework across both synthetic numerical benchmarks and an engineering-oriented constrained optimization case.

4.4. Parameter settings and implementation details

All algorithms were evaluated under consistent comparison settings within each experiment. For the CEC2021 benchmark suite, all algorithms were tested on the 20-dimensional functions using the same stopping criterion and the same number of independent runs. For the welded beam design case study, the same algorithm parameter settings, stopping criterion, and number of independent runs were adopted, while the problem dimension and search ranges followed the standard welded beam formulation described in Section 4.2. The detailed parameter settings of all compared algorithms are listed in Table 2.

Table 2. Parameter settings of all compared algorithms.

Algorithm	Parameter settings
GOA	Population size = 50; exploration coefficient $\alpha = 0.02$; exploitation coefficient $\beta = 0.5$; jump probability $p_{\text{jump}} = 0.15$; reset ratio $\rho = 0.10$.
GOA-PD	Population size = 50; exploration coefficients $\alpha_{\max} = 0.03$, $\alpha_{\min} = 0.005$; exploitation coefficient $\beta = 0.5$; jump probability $p_{\text{jump}} = 0.15$; EMA factor $\eta = 0.65$; reset ratio $\rho = 0.10$.
ACOR	Archive size = 50; sample count = 50; $q = 0.25$; $\xi = 0.9$.
PSO	Number of particles = 50; inertia weight = 0.8; $c_1 = 1.8$; $c_2 = 1.8$; $v_{\max}$ ratio = 0.2.
GWO	Number of wolves = 50.
DE	Population size = 50; mutation factor $F = 0.8$; crossover rate $CR = 0.9$.
GA	Population size = 50; crossover rate = 0.8; mutation rate = 0.1; elitism = 2; tournament size = 2.

For the CEC2021 benchmark suite, all algorithms were run for 1000 iterations on the 20-dimensional functions, with 30 independent runs for each algorithm-function pair. For the welded beam design case study, all algorithms were also run for 1000 iterations with 30 independent runs, and the four-dimensional design-variable bounds were those defined in Section 4.2.

4.5. Evaluation criteria and experimental analysis

Performance evaluation was conducted from both benchmark and engineering perspectives. For the CEC2021 benchmark suite, final optimality-gap statistics, average rank, runtime, convergence and stability plots, nonparametric statistical tests, ablation analysis, and parameter sensitivity analysis were used. For the welded beam design case study, best, mean, standard deviation, worst objective value, feasible rate, average runtime, and convergence behavior over 30 independent runs were reported to assess the engineering applicability of the proposed method.

5. Results and discussion

Section 5 reports the numerical results of the GOA-PD on the CEC2021 20-dimensional benchmark suite and further presents a welded beam design case study. The results are discussed from the perspectives of overall performance, convergence behavior, statistical significance, ablation, parameter sensitivity, and engineering applicability.

5.1. Overall results on the CEC2021 benchmark suite

The overall experimental results of the compared algorithms on the 20-dimensional CEC2021 benchmark suite are summarized in Table 3, where the mean, standard deviation, and rank of the final optimality gap over 30 independent runs are reported.

As shown in Table 3, the GOA-PD generally exhibits competitive performance on the CEC2021 benchmark suite and achieves relatively favorable mean gaps on a considerable number of benchmark functions. In comparison with the original GOA, the proposed method shows a more consistent tendency toward smaller final gaps and, in many cases, lower variability, suggesting that the improved algorithm indeed provides a certain performance gain in terms of search accuracy and stability across a range of problem landscapes.

On the other hand, the advantage of the GOA-PD is not equally pronounced on all functions, especially on some composition functions where the differences among algorithms become smaller. This is reasonable because such functions involve stronger landscape heterogeneity, rotation, hybridization, and interactions among multiple subcomponents. Under these conditions, a single global dispersion indicator may not fully characterize the local search difficulty of different regions, which may reduce the consistency of the improvement on particularly difficult landscapes.

For a more comprehensive overall comparison, the average rank and the average runtime of all compared algorithms are reported in Table 4. In terms of average rank, the GOA-PD achieves the most favorable overall result among all methods, followed by DE and ACOR, whereas the GOA, PSO, and the GWO are ranked lower overall. This ranking pattern indicates that the GOA-PD performs relatively well across different benchmark functions in a more balanced manner, rather than relying on isolated advantages on only a few problems. In particular, the clear improvement over the original GOA suggests that the proposed adaptive mechanism improves the baseline framework at the overall level.

Table 3. Computational results.

Function	Optimality gap (mean ± std) and rank						
	GOA-PD	GOA	ACOR	DE	GA	GWO	PSO
F1	**2028.19** ± **1360.50**	1.45E7 ± 2.18E6	6693.76 ± 3834.87	1.18E7 ± 4.70E6	1.59E7 ± 4.81E6	1.53E9 ± 6.71E8	2.58E8 ± 5.86E8
Rank	1	4	2	3	5	7	6
F2	2016.13 ± 444.72	2628.49 ± 468.30	3543.46 ± 348.53	3051.19 ± 479.23	**968.41** ± **242.57**	2697.54 ± 400.95	1215.46 ± 349.85
Rank	3	4	7	6	1	5	2
F3	**87.641** ± **24.065**	674.81 ± 84.750	133.77 ± 11.371	494.98 ± 159.50	703.62 ± 166.46	2.12E4 ± 8331.38	4483.99 ± 9699.76
Rank	1	4	2	3	5	7	6
F4	**3.1647** ± **1.5709**	11.035 ± 1.1250	10.108 ± 0.7619	12.683 ± 1.1903	5.4962 ± 1.2870	36.607 ± 79.264	108.55 ± 536.24
Rank	1	4	3	5	2	6	7
F5	5.87E4 ± 3.15E4	8.71E4 ± 5.35E4	9.39E5 ± 6.20E5	**5.59E4** ± **1.93E4**	1.13E5 ± 7.17E4	4.86E5 ± 5.41E5	1.92E5 ± 2.51E5
Rank	2	3	7	1	4	6	5
F6	5939.25 ± 3210.25	1.06E4 ± 4670.87	102.73 ± 205.81	**44.240** ± **40.924**	1236.88 ± 1022.64	1.09E4 ± 983.94	8946.34 ± 1.31E4
Rank	4	6	2	1	3	7	5
F7	5.96E4 ± 4.07E4	2.47E5 ± 1.29E5	4.98E5 ± 5.82E5	1.05E5 ± 5.35E4	**4.18E4** ± **5.03E4**	3.39E6 ± 1.91E6	4.10E5 ± 5.06E5
Rank	2	4	6	3	1	7	5
F8	171.99 ± 34.694	231.13 ± 69.754	**105.43** ± **11.213**	116.10 ± 3.9119	116.13 ± 1.3413	137.67 ± 5.3932	156.22 ± 17.304
Rank	6	7	1	2	3	4	5
F9	214.97 ± 37.445	502.46 ± 36.824	**207.05** ± **38.627**	376.88 ± 50.370	491.38 ± 60.731	1862.54 ± 633.74	2599.20 ± 1573.79
Rank	2	5	1	3	4	6	7
F10	440.34 ± 29.341	446.97 ± 32.589	417.59 ± 8.1935	**417.00** ± **0.3625**	456.13 ± 34.287	514.02 ± 38.785	459.14 ± 65.306
Rank	3	4	2	1	5	7	6

Here, E denotes scientific notation, e.g., 1.45E7 = 1.45×10^7. The best result in each row is highlighted in bold. The rank of each algorithm is reported below the corresponding result row, and a smaller rank indicates better performance.

Table 4. Average rank and average runtime of the algorithms.

Metric	GOA-PD	GOA	ACOR	DE	GA	GWO	PSO
Avg. Rank	2.5	4.5	3.3	2.8	3.3	6.2	5.4
Avg. Runtime (s)	8.949	8.788	19.953	9.703	10.195	8.974	6.204

Avg. Runtime (s) is the average computational time in seconds for each algorithm over all benchmark functions and independent runs.

In terms of computational cost, the GOA-PD remains within a moderate runtime range among the compared algorithms. Although it introduces additional dispersion-statistics and adaptive-control computations, the resulting runtime overhead is marginal. As reported in Table 4, the average runtime of the GOA-PD is 8.949 s, compared with 8.788 s for the original GOA, corresponding to only a slight increase in computational cost. In contrast, the average rank improves from 4.5 to 2.5, indicating that

the proposed adaptive mechanism provides a favorable performance–efficiency trade-off.

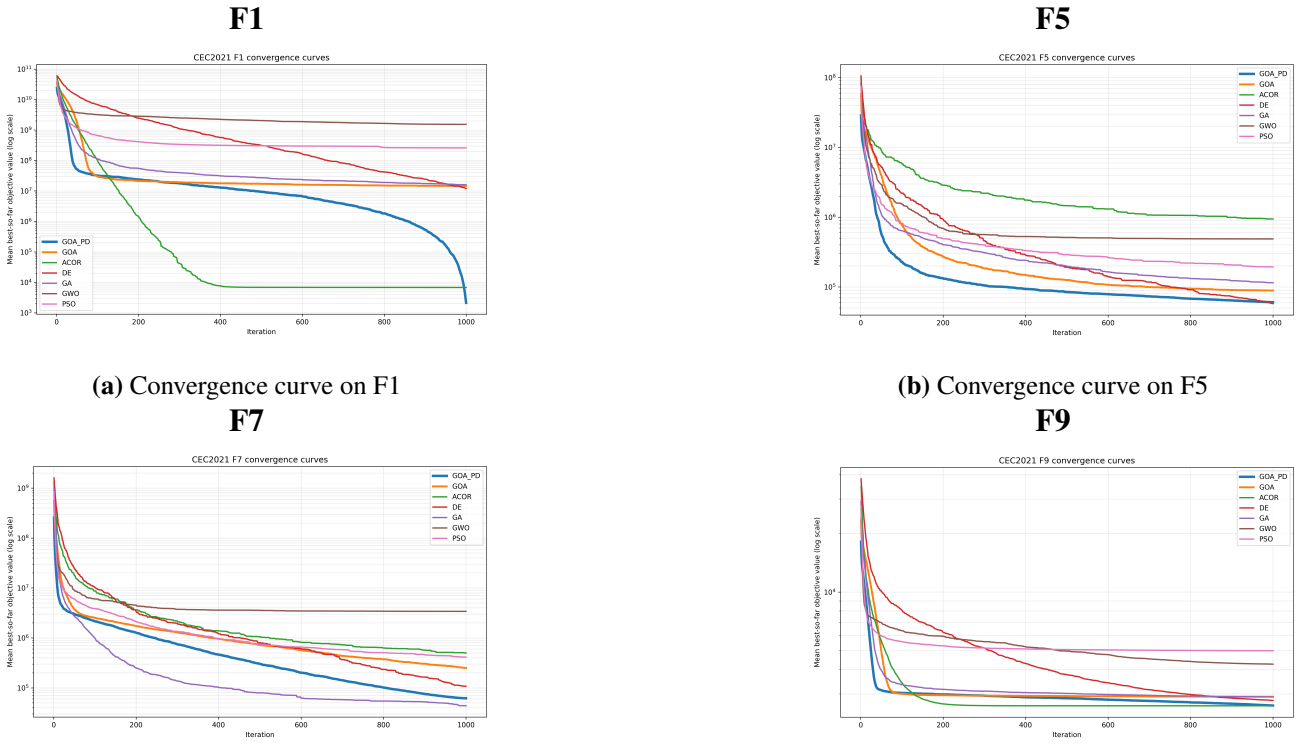
Overall, these results suggest that the GOA-PD improves the baseline GOA with only a limited additional computational burden, while maintaining competitive efficiency relative to the other compared metaheuristics.

5.2. Convergence and stability analysis

In this subsection, only four representative CEC2021 benchmark functions are presented. This is because including the plots of all benchmark functions in the main text would be unnecessarily repetitive and would occupy excessive space. The selected functions reflect relatively different convergence patterns and therefore provide a representative view of the search behavior of the compared algorithms.

5.2.1. Convergence curves

The convergence curves of four representative CEC2021 benchmark functions are shown in Figure 2. For each function, the mean best-so-far optimality gap over 30 independent runs is plotted against the iteration number, so as to illustrate the dynamic search behavior of the compared algorithms during the optimization process.



(a) Convergence curve on F1

(b) Convergence curve on F5

(c) Convergence curve on F7

(d) Convergence curve on F9

Figure 2. Convergence curves for four representative CEC2021 functions.

As shown in Figure 2(a), the GOA-PD exhibits a favorable convergence trend on F1. Although several comparison algorithms reduce the gap rapidly in the early stage, the GOA-PD continues to

improve steadily in the later stage and eventually reaches the lowest final gap among the compared methods. This result suggests that the GOA-PD maintains a relatively effective balance between global exploration and late-stage refinement on this function.

For F5, as shown in Figure 2(b), the GOA-PD also demonstrates competitive convergence behavior. Its objective gap decreases rapidly in the early stage and then continues to improve in a relatively stable manner. Although the final performance difference between the GOA-PD and some competing algorithms is not very large, the GOA-PD remains among the better-performing methods throughout most of the search process, indicating a favorable convergence profile on this hybrid function.

For F7, shown in Figure 2(c), the relative performance differences among algorithms are more evident. In this case, the GOA-PD still shows a clear downward convergence trend and achieves a competitive final result, but it does not attain the lowest final gap among all methods. In particular, the GA exhibits stronger convergence performance on this function. This result indicates that the advantage of the GOA-PD is not uniform across all problems, but it nevertheless remains competitive under more complex search landscapes.

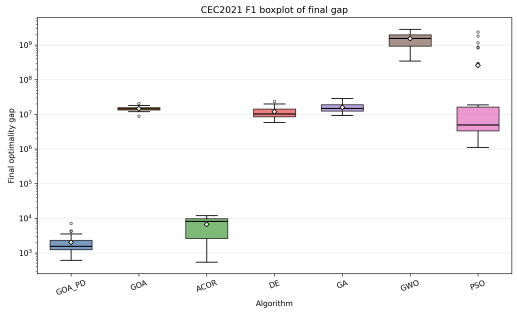
For F9, as shown in Figure 2(d), the GOA-PD reduces the objective gap quickly in the early stage and then enters a relatively stable refinement phase. Its final performance is competitive, although the differences among several algorithms become smaller in the later stage, and some competing methods achieve slightly lower final gaps. This observation again suggests that the convergence behavior of the GOA-PD is problem-dependent, rather than absolutely dominant on every function.

Specifically, the remaining six convergence curves in the Supplementary Information further support the same overall observation. On F3 and F4, the GOA-PD converges rapidly and maintains a favorable late-stage refinement trend, yielding highly competitive final results. On F2 and F10, the GOA-PD remains among the leading methods during most of the search process, although it does not achieve the best final gap. On F6 and F8, the GOA-PD still shows a stable downward trend and improves over the original GOA, but its relative advantage over some competing algorithms becomes weaker. These results further indicate that the convergence behavior of the GOA-PD is generally competitive, while its degree of improvement remains dependent on the landscape characteristics of different benchmark functions.

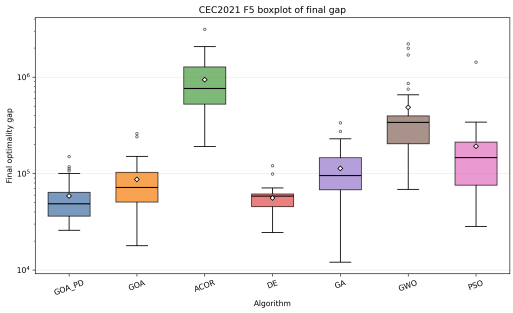
Overall, the convergence curves indicate that the GOA-PD generally exhibits a stable and competitive search trend on the selected representative benchmark functions. Compared with the original GOA, the GOA-PD tends to show more favorable convergence behavior in several cases, especially in terms of late-stage refinement or overall search stability. At the same time, the relative advantage of the GOA-PD varies across functions, which is consistent with the heterogeneous landscape characteristics of the CEC2021 benchmark suite.

5.2.2. Boxplots of final gaps

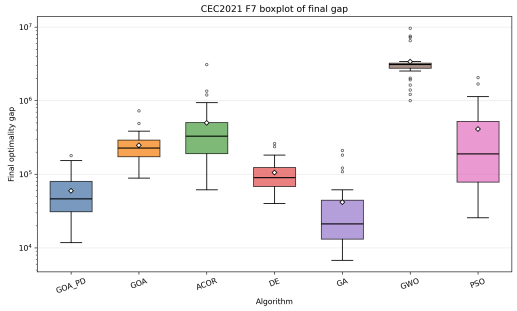
The boxplots of the final optimality gaps for the same four representative CEC2021 benchmark functions are shown in Figure 3. These boxplots summarize the distribution of the final results over 30 independent runs and provide a direct view of the convergence stability of the compared algorithms in terms of the median, interquartile range, and potential outliers.



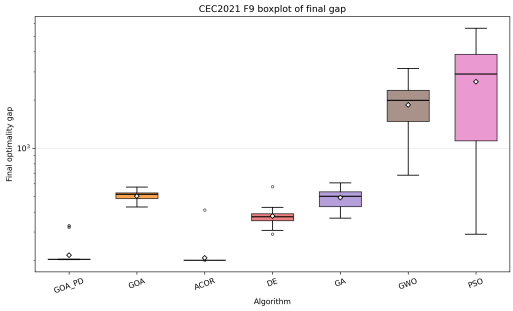
(a) Boxplot of final optimality gaps on F1



(b) Boxplot of final optimality gaps on F5



(c) Boxplot of final optimality gaps on F7



(d) Boxplot of final optimality gaps on F9

Figure 3. Boxplots of final optimality gaps for four representative CEC2021 functions.

As shown in Figure 3(a), the GOA-PD exhibits the lowest median gap and a relatively concentrated distribution on F1, indicating favorable final accuracy and stability.

For F5, the GOA-PD also shows a relatively low median and a compact distribution. Although the differences among several algorithms are smaller than those on F1, the GOA-PD remains among the better-performing methods.

For F7, the performance differences become more pronounced. The GOA-PD still achieves a competitive final distribution, but it does not attain the lowest median gap; in this case, the GA shows a lower and more concentrated distribution.

For F9, the GOA-PD presents a relatively low median and a narrow interquartile range, indicating stable final performance. However, its advantage over the best-performing competing method is limited, suggesting competitive but not dominant behavior on this function.

Specifically, the remaining boxplots in the Supplementary Information show similar patterns. On F3 and F4, the GOA-PD exhibits relatively low median gaps together with compact interquartile ranges, indicating favorable final accuracy and stability over repeated runs. On F2 and F10, the GOA-PD also shows a comparatively concentrated distribution and performs more favorably than the original GOA and several competing methods, although it is not the best-performing algorithm on these functions. By contrast, on F6 and F8, the advantage of the GOA-PD becomes less pronounced, and some competing algorithms achieve lower medians or narrower distributions. This suggests that, on these more difficult landscapes, the improvement brought by the adaptive exploration mechanism is more problem-dependent and may be influenced by the interaction between the global dispersion signal and the local search structure of the function. Nevertheless, the GOA-PD still maintains a relatively stable

final distribution and remains competitive overall.

Overall, the boxplot results are broadly consistent with the convergence-curve analysis. Compared with the original GOA, the GOA-PD generally achieves lower median gaps and more concentrated final distributions on the selected representative functions. These results suggest that the GOA-PD provides favorable convergence stability across repeated runs, while the degree of improvement may vary with different benchmark functions.

5.3. Statistical significance analysis

To further verify whether the observed performance differences obtained on the CEC2021 benchmark suite are statistically meaningful rather than caused by random fluctuation, statistical significance analysis was performed using the Wilcoxon signed-rank test, Friedman test, and Holm-adjusted post-hoc comparison, following recent practices in metaheuristic benchmarking and algorithm comparison [23–25]. Unless otherwise stated, all statistical conclusions in this subsection were drawn at the 0.05 significance level.

5.3.1. Pairwise Wilcoxon signed-rank test

To further examine whether the observed performance differences are statistically meaningful, pairwise Wilcoxon signed-rank tests were conducted between the GOA-PD and each competing algorithm on the CEC2021 benchmark suite.

As shown in Table 5, the GOA-PD exhibits a clear pairwise advantage over the original GOA, achieving a 10/0/0 result across the 10 benchmark functions. A similarly favorable outcome can also be observed against the GWO, for which the GOA-PD obtains a 9/0/1 summary. Against ACOR and the GA, the GOA-PD performs better on 6 out of 10 functions, while against DE and PSO, it still shows a majority of favorable comparisons, with results of 6/2/2 and 6/3/1, respectively. Detailed per-function Wilcoxon test results are provided in the Supplementary Information.

Table 5. Pairwise Wilcoxon test summary.

Baseline	+	=	−
ACOR	6	0	4
DE	6	2	2
GA	6	0	4
GOA	10	0	0
GWO	9	0	1
PSO	6	3	1

Note: Pairwise Wilcoxon signed-rank test results between the GOA-PD and each competing algorithm on the CEC2021 benchmark suite. "+" indicates that the GOA-PD performs better, "=" indicates no significant difference, and "-" indicates worse performance.

Overall, the Wilcoxon results indicate that the GOA-PD maintains a comparative advantage over several representative algorithms on the adopted benchmark suite. In particular, the comparison with the baseline GOA provides strong statistical support for the effectiveness of the proposed adaptive strategy. Although the extent of improvement varies across competing methods, the pairwise analysis generally suggests that the GOA-PD maintains favorable overall competitiveness on the adopted benchmark suite.

5.3.2. Friedman test and post-hoc comparison

To assess the overall statistical difference among all compared algorithms, the Friedman test was further performed, and the result is reported in Table 6. As can be seen, the Friedman statistic is 25.1143 with a p-value of 0.000325, indicating that statistically significant differences exist among the compared algorithms on the CEC2021 benchmark suite.

Table 6. Friedman test results.

Metric	Blocks	Algorithms	Friedman statistic	p-value	Significant
Gap	10	7	25.1143	0.000325	Yes

The average ranks obtained by the Friedman procedure and the Holm-adjusted post-hoc results are summarized in Table 7. The GOA-PD achieves the best overall rank of 2.5, followed by DE with 2.8, while the GA and ACOR both obtain an average rank of 3.3. The remaining algorithms are ranked lower, namely, the GOA (4.5), PSO (5.4), and the GWO (6.2). This overall ranking is broadly consistent with the numerical results, convergence curves, and boxplot analysis, and further supports the overall competitiveness of the GOA-PD on the CEC2021 benchmark suite.

Table 7. Average ranks and Holm post-hoc results.

Algorithm	Average rank	Holm-adjusted p-value	Significant after Holm
GOA-PD	2.5	–	–
DE	2.8	1.0000	No
GA	3.3	1.0000	No
ACOR	3.3	1.0000	No
GOA	4.5	0.0117	Yes
PSO	5.4	0.0781	No
GWO	6.2	0.0293	Yes

In addition, the Holm-adjusted post-hoc comparisons show that the GOA-PD remains significantly better than the GOA and GWO after multiple-comparison correction, whereas the differences with DE, the GA, ACOR, and PSO are not statistically significant at the adopted significance level. This result does not contradict the significant Friedman test, because the two analyses address different questions: the Friedman test evaluates whether overall differences exist among all algorithms, whereas the Holm post-hoc test further examines whether the GOA-PD significantly outperforms each individual competitor after family-wise error correction. Since only 10 benchmark functions are used as statistical blocks, the post-hoc test has limited power. Moreover, the advantages of the GOA-PD over DE, the GA, ACOR, and PSO are favorable but not overwhelming. As shown in Table 5, the corresponding pairwise Wilcoxon summaries are 6/2/2 against DE, 6/0/4 against the GA, 6/0/4 against ACOR, and 6/3/1 against PSO, indicating that these competitors still achieve strong results on several functions. Therefore, although the GOA-PD attains the best overall average rank, the rank margins over these strong baselines are not always large enough to remain significant after Holm correction. By contrast, the improvements over the GOA and GWO are more consistent, which explains why only these two pairwise differences remain statistically significant. In other words, the results suggest that the GOA-PD provides balanced overall performance on the adopted benchmark suite, while its relative advantage

over some strong competitors is better interpreted as a robust overall tendency rather than a uniformly dominant advantage on every benchmark function.

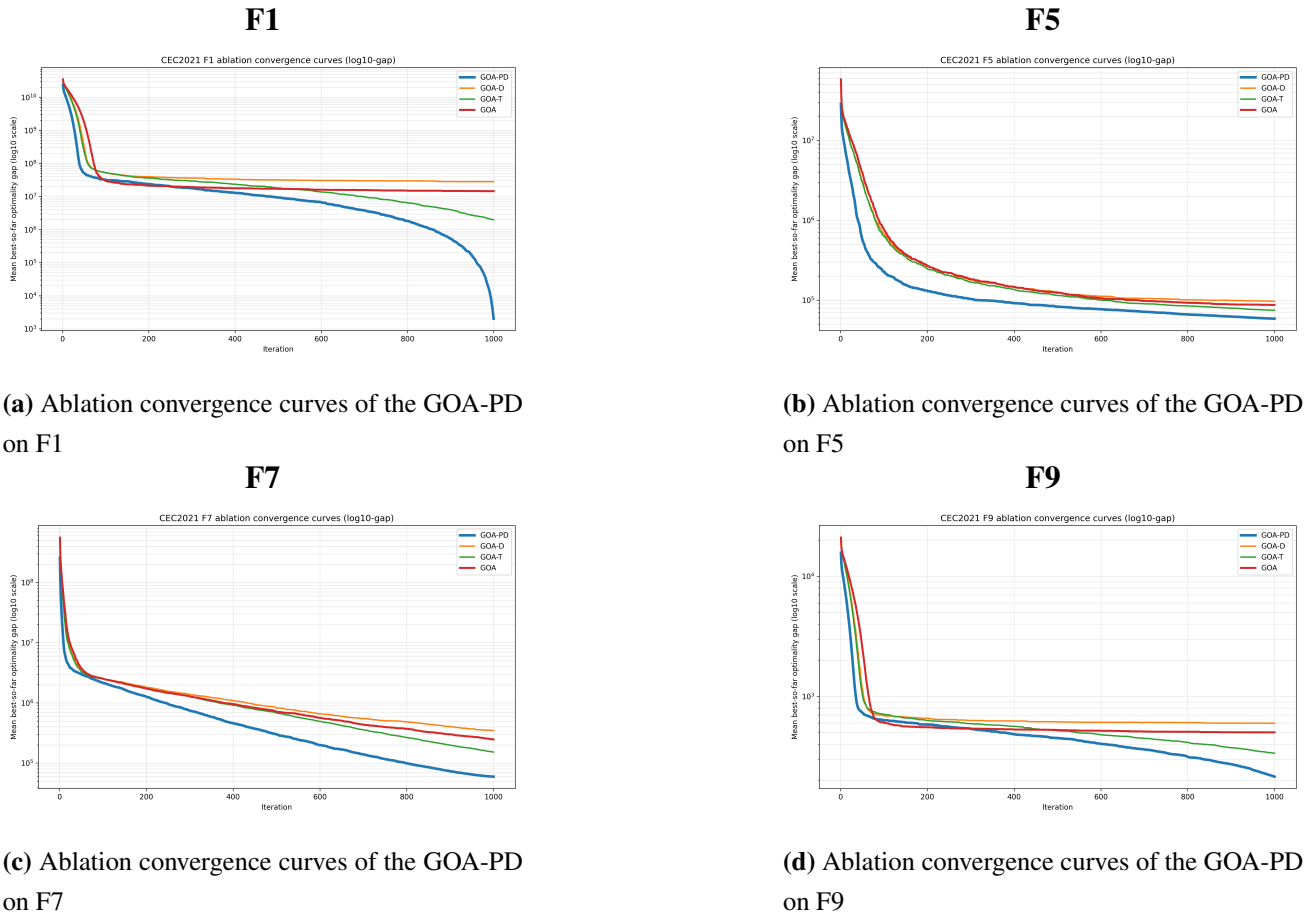
Taken together, the statistical significance analysis supports the effectiveness of the GOA-PD from both the overall comparison and the pairwise comparison. The proposed method maintains favorable overall competitiveness among the compared representative metaheuristic algorithms and also improves the original GOA in a statistically meaningful manner.

5.4. Discussion on applicability in high-dimensional complex search spaces

Although the CEC2021 20-dimensional benchmark suite already contains ill-conditioning, non-separability, multimodality, hybridization, and composition, the current validation is still limited to a single fixed dimensional setting. In the GOA-PD, the adaptive regulation is driven by the population-dispersion indicator D, which is computed as the mean normalized standard deviation across dimensions and then mapped to the exploration coefficient α_t. In higher-dimensional search spaces, however, different dimensions or subspaces may exhibit heterogeneous convergence states, variable interactions, and search difficulties. Under such conditions, this global average signal may smooth out local differences and may not accurately reflect where stronger exploration or more cautious refinement is needed. This may partly explain why the improvement of the GOA-PD is not equally pronounced on all complex benchmark functions. Therefore, the GOA-PD should be regarded as an effective adaptive extension of the GOA under the present benchmark setting, while its applicability to larger-scale high-dimensional optimization still requires further investigation.

5.5. Ablation analysis of time-decay and dispersion feedback

To clarify the respective roles of the proposed adaptive components, we conducted an ablation study on the CEC2021 benchmark set by comparing the original GOA with three variants. All variants were evaluated under the same experimental settings, with 30 independent runs on each test function. The final statistical results are reported in Table 8, and the corresponding mean best-so-far optimality gap curves are shown in Figure 4.



(a) Ablation convergence curves of the GOA-PD on F1

(b) Ablation convergence curves of the GOA-PD on F5

(c) Ablation convergence curves of the GOA-PD on F7

(d) Ablation convergence curves of the GOA-PD on F9

Figure 4. Representative ablation convergence curves of the GOA-PD on the CEC2021 benchmark functions.

The ablation results show that the two components play different but complementary roles. The GOA-T outperforms the original GOA on all ten benchmark functions, indicating that the time-decay term is the main source of broad and consistent improvement. Its role is to provide a coarse-to-fine search schedule along the iteration axis, so that the search process gradually shifts from broader exploration to finer local refinement instead of relying on a fixed exploration strength. By contrast, the GOA-D does not yield equally stable improvement when used alone. Although it improves the baseline on several functions, its performance is not consistently better than that of the GOA, suggesting that dispersion feedback should not be viewed as an independent replacement for time scheduling. Rather, it acts as a state-dependent correction term: when the population becomes overly concentrated, it enlarges the exploration radius to reduce the risk of premature convergence; when the population remains highly dispersed, it suppresses unnecessary wandering and promotes more effective information aggregation.

When these two terms are combined, the GOA-PD achieves the best overall performance, ranking first on 9 out of the 10 test functions. These ablation results further support the necessity of combining dispersion feedback and time-decay regulation, indicating that the proposed hybrid schedule is not an arbitrary combination but a complementary design. More specifically, the time-decay term supplies a global search schedule, whereas the dispersion-feedback term provides a state-aware local correction. Their cooperation allows the adaptive coefficient to vary in a structured yet responsive manner, leading

to a better balance between exploration and exploitation across different search landscapes.

The ablation convergence curves further support the results in Table 8. On F1, F3, F4, and F9, the GOA-PD maintains a clearer late-stage improvement trend and finally achieves the lowest gap among the four variants. On F2, F5, F6, and F7, the GOA-PD also exhibits faster or more stable descent than the GOA, GOA-D, and GOA-T, while the GOA-T usually remains the second-best variant. On F10, the GOA-PD and GOA-T show very similar convergence behavior, but the GOA-PD still attains a slightly better final result. F8 is the main exception, where the GOA-T achieves a marginally lower final gap than the GOA-PD. Overall, these curves indicate that the time-decay term provides the main source of consistent improvement, while the additional dispersion feedback further enhances the search performance on most benchmark functions.

Table 8. Ablation results of the GOA-PD on the CEC2021 benchmark functions.

Function	Optimality gap (mean ± std) and rank			
	GOA-PD	GOA-T	GOA-D	GOA
F1	**2028.19** ±1360.50	1.99E6 ±3.39E5	2.82E7 ±4.77E6	1.45E7 ±2.18E6
Rank	1	2	4	3
F2	**2016.13** ±444.72	2063.80 ±466.23	2411.04 ±513.77	2628.49 ±468.30
Rank	1	2	3	4
F3	**87.641** ±24.065	237.44 ±21.122	1049.75 ±156.40	674.81 ±84.750
Rank	1	2	4	3
F4	**3.165** ±1.571	8.467 ±1.048	9.886 ±0.859	11.035 ±1.125
Rank	1	2	3	4
F5	**5.87E4** ±3.15E4	7.48E4 ±4.09E4	9.65E4 ±5.17E4	8.71E4 ±5.35E4
Rank	1	2	4	3
F6	**5939.25** ±3210.25	8306.32 ±3719.89	1.08E4 ±4269.67	1.06E4 ±4670.87
Rank	1	2	4	3
F7	**5.96E4** ±4.07E4	1.52E5 ±9.58E4	3.47E5 ±1.85E5	2.47E5 ±1.29E5
Rank	1	2	4	3
F8	171.99 ±34.694	**170.52** ±50.265	187.69 ±50.462	231.13 ±69.754
Rank	2	1	3	4
F9	**214.97** ±37.445	336.06 ±49.245	597.70 ±81.677	502.46 ±36.824
Rank	1	2	4	3
F10	**440.34** ±29.341	443.31 ±27.131	455.59 ±35.299	446.97 ±32.589
Rank	1	2	4	3
Avg. rank	**1.1**	1.9	3.7	3.3
Wins	**9**	1	0	0

Here, E denotes scientific notation, e.g., 1.99E6 = 1.99×10^6. The best result in each function row is highlighted in bold. The rank of each algorithm is reported in the corresponding rank row, and a smaller rank indicates better performance.

It should also be noted that the GOA-PD is not the best variant on every function. On F8, the GOA-T achieves a slightly smaller final mean gap than the GOA-PD. However, this should not be attributed to the composition-function category itself, since F9 and F10 do not show the same pattern. A more plausible explanation is that the landscape of F8 interacts more favorably with the smooth monotonic

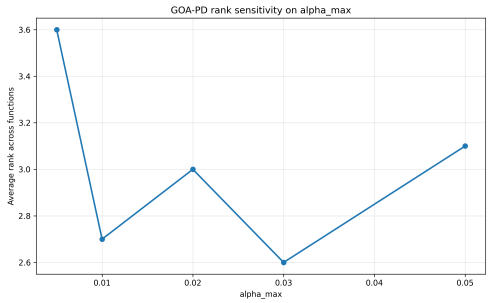
decay schedule of the GOA-T. In the GOA-T, the exploration coefficient decreases only with iteration, yielding a stable one-way shrinkage of the search radius. In the GOA-PD, the additional dispersion-based correction makes the coefficient more responsive to the instantaneous population state. Although this correction is beneficial on most functions, on F8, it does not further improve the final result. Therefore, the F8 result is better regarded as a landscape-dependent exception.

From the operator perspective, this also helps explain why adapting only α_t does not invalidate the subsequent exploitation and jump stages, even though β and p_{jump} remain fixed. The adaptive α_t changes only the exploration radius and the intermediate candidate entering the later stages, but not the functional roles of β and p_{jump}. Thus, on some landscapes such as F8, the GOA-T may cooperate slightly better with the fixed later-stage operators, whereas on most other functions, the additional dispersion correction in the GOA-PD brings greater benefit. Hence, the F8 result should be viewed as a local exception rather than evidence against the overall effectiveness of the complete adaptive mechanism.

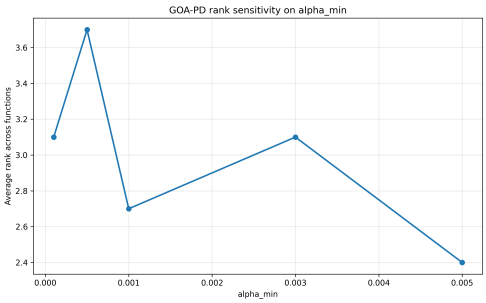
5.6. Parameter sensitivity analysis

To justify the parameter settings used in the main experiments and assess the robustness of the GOA-PD, we conducted a parameter sensitivity analysis on the CEC2021 20-dimensional benchmark suite. Using a one-factor-at-a-time strategy, we varied one parameter while keeping the others at their default values. Performance was evaluated by the average rank across the benchmark functions, where a smaller rank indicates better overall robustness. Here, the average rank is calculated among the candidate values of the same parameter on each benchmark function, rather than against other parameters or other algorithms. The corresponding results are shown in Figures 5 and 6.

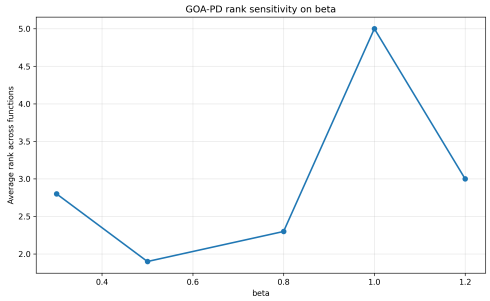
Figure 5 presents the sensitivity results for $\alpha_{\max}$, $\alpha_{\min}$, β, and EMA. It can be observed that the GOA-PD achieves better performance under moderate settings of $\alpha_{\max}$, while overly small or overly large values lead to inferior average ranks. In particular, $\alpha_{\max} = 0.03$ gives the best overall rank, indicating that a moderately large upper exploration bound is beneficial for maintaining sufficient global search ability in the early stage. For $\alpha_{\min}$, the best result is obtained at $\alpha_{\min} = 0.005$. This indicates that the lower bound of the dispersion-based coefficient in Eq (3.3) should be set to a moderate level. Here, $\alpha_{\min}$ controls the baseline scale before time decay, rather than acting as a strict lower bound of the final adaptive coefficient in the later stage. The sensitivity of β is more pronounced. The average rank is minimized at $\beta = 0.5$, whereas $\beta = 1.0$ leads to a clear deterioration in performance, implying that excessively strong attraction toward the current best solution may accelerate loss of diversity and reduce the stability of the search. For EMA, the best performance is achieved at 0.65, while both weaker smoothing and excessively strong smoothing yield poorer ranks. This indicates that a moderate smoothing effect is more suitable for stabilizing the adaptive update of α_t without making the search overly inertial.



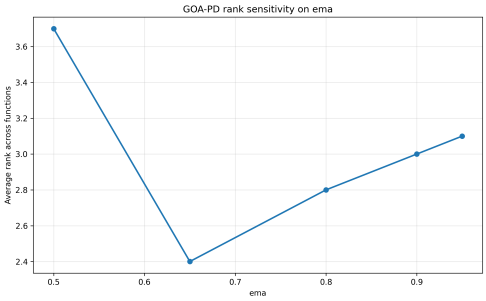
(a) Sensitivity to $\alpha_{\max}$



(b) Sensitivity to $\alpha_{\min}$



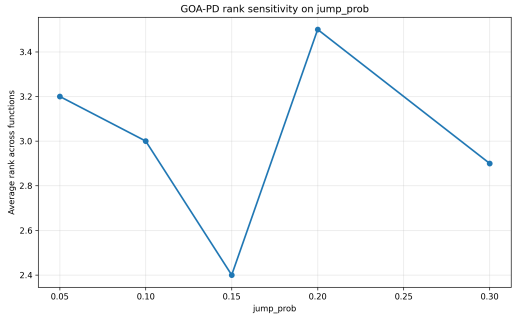
(c) Sensitivity to β



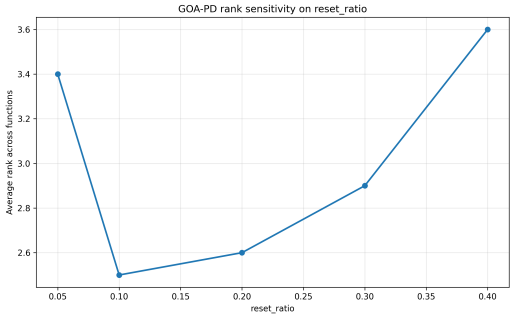
(d) Sensitivity to EMA

Figure 5. Parameter sensitivity analysis of the GOA-PD with respect to $\alpha_{\max}$, $\alpha_{\min}$, β, and EMA in terms of average rank across functions.

Figure 6 shows the sensitivity results for the jump probability and reset ratio. Among these parameters, the jump probability obtains the best average rank at 0.15, while both smaller and larger values are less effective. This suggests that an intermediate jump intensity is more appropriate for balancing escape from local optima and preserving search stability. For the reset ratio, the best rank is obtained at 0.10, whereas both smaller and larger reset ratios lead to worse performance. This result suggests that a moderate replacement intensity is more effective, since too small a reset ratio is insufficient to refresh poor individuals, while too large a ratio may disrupt the ongoing search process.



(a) Sensitivity to jump probability



(b) Sensitivity to reset ratio

Figure 6. Parameter sensitivity analysis of the GOA-PD with respect to the jump probability and reset ratio in terms of the average rank across functions.

Based on the above sensitivity results, the final parameter settings of the GOA-PD were chosen as

$\alpha_{\max} = 0.03$, $\alpha_{\min} = 0.005$, $\beta = 0.5$, EMA factor $\eta = 0.65$, jump probability $p_{\text{jump}} = 0.15$, and reset ratio $\rho = 0.10$. Overall, the parameter sensitivity analysis indicates that the GOA-PD does not rely on overly narrow tuning, but it performs more reliably under moderate and well-balanced parameter settings.

5.7. Engineering case study on welded beam design

To further evaluate the engineering applicability of the GOA-PD, the welded beam design problem described in Section 4.2 was used as a constrained engineering optimization case study. The results from 30 independent runs are summarized in Table 9. Best, Mean, Std, and Worst denote the statistics of the feasible objective costs, while x_1, x_2, x_3, and x_4 denote the best feasible design variables identified by each algorithm. The rank is determined according to the mean objective cost, with a smaller rank indicating better performance.

Table 9. Results on the welded beam design problem over 30 independent runs.

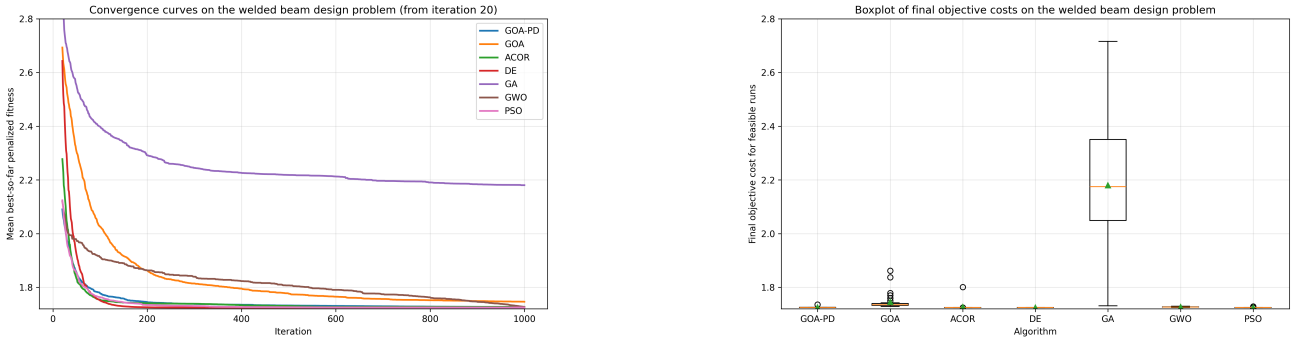
Algorithm	Best	Mean	Std	Worst	x_1	x_2	x_3	x_4	Feasible rate	Runtime (s)	Rank
DE	1.724852	1.724852	5.17E-13	1.724852	0.205730	3.470489	9.036624	0.205730	100%	6.409	1
PSO	1.724852	1.725016	7.65E-04	1.729060	0.205730	3.470490	9.036624	0.205730	100%	2.103	2
GOA-PD	1.724984	1.725558	1.94E-03	1.735786	0.205694	3.471596	9.036722	0.205732	100%	4.825	3
GWO	1.725075	1.726967	1.04E-03	1.729406	0.205643	3.472353	9.036740	0.205741	100%	5.013	4
ACOR	1.724852	1.727397	1.39E-02	1.801207	0.205730	3.470489	9.036624	0.205730	100%	8.676	5
GOA	1.728877	1.746351	3.05E-02	1.862035	0.204429	3.506987	9.036669	0.205876	100%	4.302	6
GA	1.731048	2.180420	2.28E-01	2.716408	0.206264	3.470723	9.013594	0.206957	100%	5.404	7

As shown in Table 9, all compared algorithms obtained feasible solutions in all 30 independent runs, indicating that the adopted penalty-based constraint-handling strategy was effective for guiding the search toward feasible design regions. Among the compared methods, the GOA-PD achieved a mean objective cost of 1.725558, ranking among the top-performing algorithms. Its mean result was very close to those of DE and PSO, with differences of only 7.06×10^{-4} and 5.42×10^{-4}, respectively. In terms of computational cost, the GOA-PD required an average runtime of 4.825 s, which was lower than those of DE, the GWO, ACOR, and the GA. This is consistent with the lightweight design of the proposed adaptive mechanism, which introduces only low-cost dispersion statistics without altering the overall algorithmic structure. These results indicate that the GOA-PD provides a favorable balance between solution quality and computational efficiency on this constrained engineering design problem.

Compared with the original GOA, the GOA-PD showed a clear improvement in both optimization accuracy and robustness. The mean objective cost was reduced from 1.746351 to 1.725558, and the worst objective cost was reduced from 1.862035 to 1.735786. Moreover, the standard deviation decreased from 0.030516 to 0.001936, indicating that the proposed population-dispersion-based adaptive exploration mechanism substantially improved the stability of the original GOA. These results suggest that the GOA-PD is not only competitive among representative metaheuristic algorithms, but also more reliable than the original GOA when applied to constrained engineering design optimization.

Figure 7 further illustrates the convergence and stability results on the welded beam design problem. As shown in Figure 7(a), the GOA-PD shows a stable convergence process and reaches a better final fitness level than the original GOA. Figure 7(b) further indicates that the GOA-PD produces a relatively compact final-result distribution, whereas the original GOA shows larger fluctuations and several higher-cost outliers. These observations are consistent with the numerical results in Table 9 and

further support the robustness improvement brought by the proposed adaptive exploration mechanism.



(a) Convergence curve (b) Boxplot of final objective costs

Figure 7. Convergence and stability results on the welded beam design problem.

The best feasible welded beam design identified by the GOA-PD was $x_1 = 0.205694$, $x_2 = 3.471596$, $x_3 = 9.036722$, and $x_4 = 0.205732$, with no constraint violation. This solution achieved a competitive fabrication cost, indicating that the GOA-PD can effectively search within a constrained engineering design space and obtain a physically feasible design.

From a practical optimization perspective, the welded beam design problem provides a physically meaningful constrained benchmark for evaluating the behavior of the GOA-PD under nonlinear engineering constraints. Unlike unconstrained numerical benchmarks, this problem requires the optimizer to reduce fabrication cost while simultaneously satisfying stress, deflection, buckling-load, and geometric constraints. Therefore, it allows the practical relevance of the GOA-PD to be examined from the perspectives of feasibility preservation, solution robustness, and computational efficiency. Although DE and PSO achieved slightly lower mean objective costs, the performance differences among the top-performing methods remain small. More importantly, the GOA-PD consistently improves the baseline GOA in terms of mean performance, worst-case performance, and solution stability. Therefore, this case study supports that the proposed population-dispersion-based adaptive exploration mechanism can be effectively transferred from numerical benchmarks to constrained engineering optimization scenarios, providing a reliable improvement over the baseline algorithm.

More importantly, this case study further clarifies the methodological novelty of the proposed modification. The improvement of the GOA-PD does not come from adding a problem-specific repair rule, a hybrid local search module, or an additional constraint-handling operator. Instead, the same population-dispersion-based adaptive exploration mechanism used in the unconstrained CEC2021 experiments is directly transferred to the constrained welded beam problem. The observed improvement over the original GOA in mean objective value, worst-case result, and standard deviation therefore suggests that the proposed adaptive coefficient is not merely a benchmark-specific adjustment, but a lightweight and transferable mechanism for improving the robustness of the baseline GOA under both numerical and constrained engineering optimization settings.

6. Conclusions

This study proposes the GOA-PD, an improved variant of the goat optimization algorithm with a population-dispersion-driven adaptive exploration mechanism. By introducing a normalized dispersion indicator, time-decay regulation, and EMA smoothing, the GOA-PD adjusts the exploration coefficient online while preserving the original exploitation, jumping, greedy selection, and rebirth stages of the GOA. In this way, the proposed method enhances the adaptability of the exploration process without substantially increasing the algorithmic complexity.

The experimental results on the 20-dimensional CEC2021 benchmark suite showed that the GOA-PD achieved competitive overall performance against six representative metaheuristic algorithms. The convergence curves, boxplots, and statistical tests further indicated that the proposed method provides a favorable balance between search accuracy, stability, and computational cost. In addition, the ablation study confirmed the contribution of the proposed adaptive components, and the parameter sensitivity analysis supported the rationality of the adopted parameter settings. The welded beam design case study further demonstrated that the GOA-PD can obtain feasible and competitive solutions for a constrained engineering optimization problem, suggesting its potential applicability beyond unconstrained numerical benchmarks.

Nevertheless, this study still has several limitations. First, the current validation was restricted to the 20-dimensional CEC2021 benchmark suite, so the scalability of the GOA-PD to larger-scale search spaces has not yet been fully established. Second, although a classical constrained engineering optimization benchmark was included, the present study did not involve large-scale real-world industrial data or deployment-oriented optimization tasks. Third, the current population-dispersion indicator is a global state signal and may not adequately reflect local or subspace-level search differences in highly complex landscapes. Future work could therefore extend the evaluation to higher-dimensional benchmark settings, such as 50D and 100D problems, further test the GOA-PD on practical optimization tasks including engineering design, scheduling, and path-planning scenarios, and investigate more informative state descriptors, such as multi-scale or region-aware diversity measures. Future work could also explore alternative mapping mechanisms between the population-state indicator and the adaptive exploration coefficient, such as nonlinear, piecewise, or data-driven forms, to examine whether the current linear inverse mapping can be further improved for different landscape characteristics. It would also be worthwhile to explore more coordinated adaptive control strategies for additional operators rather than adjusting only the exploration coefficient. In this way, the coupling between population-state feedback and the search process can be further refined, and the generalizability of the method can be further improved.

Use of AI tools declaration

The author declares he has not used Artificial Intelligence (AI) tools in the creation of this article.

Conflict of interest

The author declares there are no conflicts of interest.

References

1. 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. https://doi.org/10.1007/s10462-024-10747-w
2. M. H. Xu, L. Cao, D. W. Lu, Z. Y. Hu, Y. G. Yue, Application of swarm intelligence optimization algorithms in image processing: A comprehensive review of analysis, synthesis, and optimization, *Biomimetics*, **8** (2023), 235. https://doi.org/10.3390/biomimetics8020235
3. J. Tang, H. B. Duan, S. Y. Lao, Swarm intelligence algorithms for multiple unmanned aerial vehicles collaboration: A comprehensive review, *Artif. Intell. Rev.*, **56** (2023), 4295–4327. https://doi.org/10.1007/s10462-022-10281-7
4. H. Gao, Q. K. Zhang, Alpha evolution: An efficient evolutionary algorithm with evolution path adaptation and matrix generation, *Eng. Appl. Artif. Intel.*, **137** (2024), 109202. https://doi.org/10.1016/j.engappai.2024.109202
5. 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. https://doi.org/10.1109/ACCESS.2022.3142859
6. J. C. Yao, X. N. Luo, F. Li, J. Li, J. D. Dou, H. T. Luo, Research on hybrid strategy particle swarm optimization algorithm and its applications, *Sci. Rep.*, **14** (2024), 24928. https://doi.org/10.1038/s41598-024-76010-y
7. J. W. Qiao, G. Y. Wang, Z. Yang, X. C. Luo, J. Chen, K. Li, et al., A hybrid particle swarm optimization algorithm for solving engineering problem, *Sci. Rep.*, **14** (2024), 8357. https://doi.org/10.1038/s41598-024-59034-2
8. B. Alhijawi, A. Awajan, Genetic algorithms: theory, genetic operators, solutions, and applications, *Evol. Intel.*, **17** (2024), 1245–1256. https://doi.org/10.1007/s12065-023-00822-6
9. Z. Y. Taha, A. A. Abdullah, T. A. Rashid, Optimizing feature selection with genetic algorithms: A review of methods and applications, *Knowl. Inf. Syst.*, **67** (2025), 9739–9778. https://doi.org/10.1007/s10115-025-02515-1
10. J. Q. Yang, F. Yan, J. Zhang, C. G. Peng, Hybrid chaos game and grey wolf optimization algorithms for UAV path planning, *Appl. Math. Model.*, **142** (2025), 115979. https://doi.org/10.1016/j.apm.2025.115979
11. Y. F. Wang, Y. M. Yin, H. Zhao, J. X. Liu, C. Y. Xu, W. Y. Dong, Grey wolf optimizer with self-repulsion strategy for feature selection, *Sci. Rep.*, **15** (2025), 12807. https://doi.org/10.1038/s41598-025-97224-8
12. M. A. Awadallah, S. N. Makhadmeh, M. A. Al-Betar, L. M. Dalbah, A. Al-Redhaei, S. Kouka, et al., Multi-objective ant colony optimization: Review, *Arch. Computat. Methods Eng.*, **32** (2025), 995–1037. https://doi.org/10.1007/s11831-024-10178-4
13. R. Priyadarshi, R. R. Kumar, Evolution of swarm intelligence: A systematic review of particle swarm and ant colony optimization approaches in modern research, *Arch. Computat. Methods Eng.*, **32** (2025), 3609–3650. https://doi.org/10.1007/s11831-025-10247-2

14. N. K. Hussein, M. Qaraad, A. M. El Najjar, M. A. Farag, M. A. Elhosseini, S. Mirjalili, et al., Schrödinger optimizer: A quantum duality-driven metaheuristic for stochastic optimization and engineering challenges, *Knowl.-Based Syst.*, **328** (2025), 114273. https://doi.org/10.1016/j.knosys.2025.114273

15. M. Q. Ibrahim, M. Qaraad, N. K. Hussein, M. Farag, D. Guinovart, Secant optimization algorithm for efficient global optimization, *Sci. Rep.*, **16** (2026), 6659. https://doi.org/10.1038/s41598-026-36691-z

16. M. Qaraad, S. Amjad, N. K. Hussein, M. A. Elhosseini, Large scale salp-based grey wolf optimization for feature selection and global optimization, *Neural Comput. Applic.*, **34** (2022), 8989–9014. https://doi.org/10.1007/s00521-022-06921-2

17. M. Qaraad, S. Amjad, N. K. Hussein, M. A. Elhosseini, An innovative quadratic interpolation salp swarm-based local escape operator for large-scale global optimization problems and feature selection, *Neural Comput. Applic.*, **34** (2022), 17663–17721. https://doi.org/10.1007/s00521-022-07391-2

18. X. B. Zhou, H. J. Ma, J. G. Gu, H. L. Chen, W. Deng, Parameter adaptation-based ant colony optimization with dynamic hybrid mechanism, *Eng. Appl. Artif. Intel.*, **114** (2022), 105139. https://doi.org/10.1016/j.engappai.2022.105139

19. E. H. Houssein, M. H. A. Gafar, N. Fawzy, A. Y. Sayed, Recent metaheuristic algorithms for solving some civil engineering optimization problems, *Sci. Rep.*, **15** (2025), 7929. https://doi.org/10.1038/s41598-025-90000-8

20. M. S. Shaikh, H. Y. Lin, S. L. Xie, X. Q. Dong, Y. F. Lin, C. K. Shiva, et al., An intelligent hybrid grey wolf-particle swarm optimizer for optimization in complex engineering design problem, *Sci. Rep.*, **15** (2025), 18313. https://doi.org/10.1038/s41598-025-02154-0

21. P. Rajasekar, M. Jayalakshmi, Adaptive memory-based opposition and midpoint mutation in black winged kite algorithm for global optimization and engineering applications, *Sci. Rep.*, **16** (2026), 2401. https://doi.org/10.1038/s41598-025-31729-0

22. C. A. C. Coello, Use of a self-adaptive penalty approach for engineering optimization problems, *Comput. Ind.*, **41** (2000), 113–127. https://doi.org/10.1016/S0166-3615(99)00046-9

23. A. P. Piotrowski, J. J. Napiorkowski, A. E. Piotrowska, Metaheuristics should be tested on large benchmark set with various numbers of function evaluations, *Swarm Evol. Comput.*, **92** (2025), 101807. https://doi.org/10.1016/j.swevo.2024.101807

24. J. Brest, M. S. Maučec, Comparative study of modern differential evolution algorithms: Perspectives on mechanisms and performance, *Mathematics*, **13** (2025), 1556. https://doi.org/10.3390/math13101556

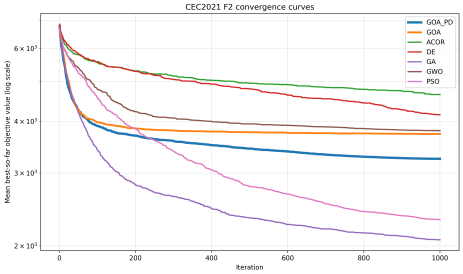
25. H. Emami, M. Fardi, B. Azarnavid, An enhanced seasons optimization algorithm for numerical optimization and engineering design, *Sci. Rep.*, **15** (2025), 25675. https://doi.org/10.1038/s41598-025-11626-2

A. Supplementary

Supplementary material provides the remaining experimental results that are not shown in the main text.

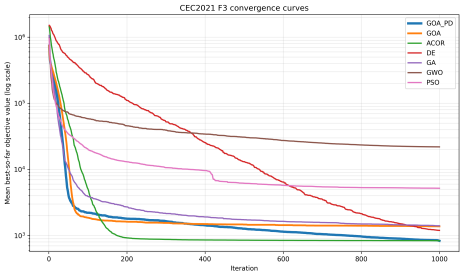
Figures 8 and 9 present the remaining convergence curves.

F2



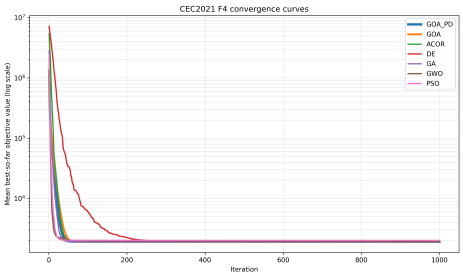
(a) Convergence curve on F2

F3



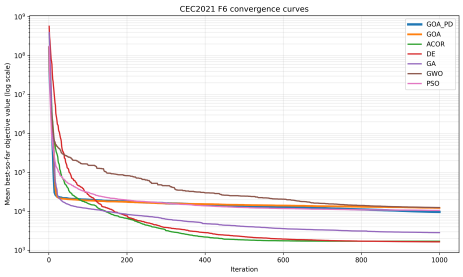
(b) Convergence curve on F3

F4



(c) Convergence curve on F4

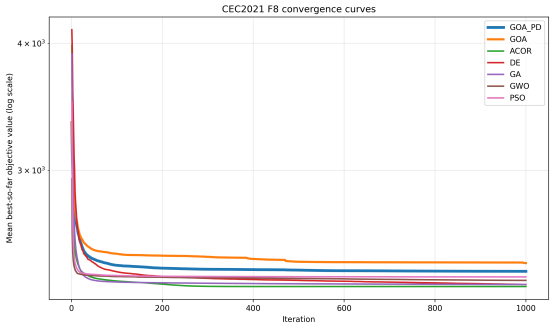
F6



(d) Convergence curve on F6

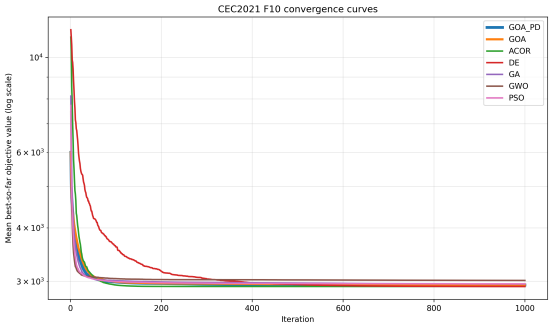
Figure 8. Convergence curves for CEC2021 functions F2, F3, F4, and F6.

F8



(a) Convergence curve on F8

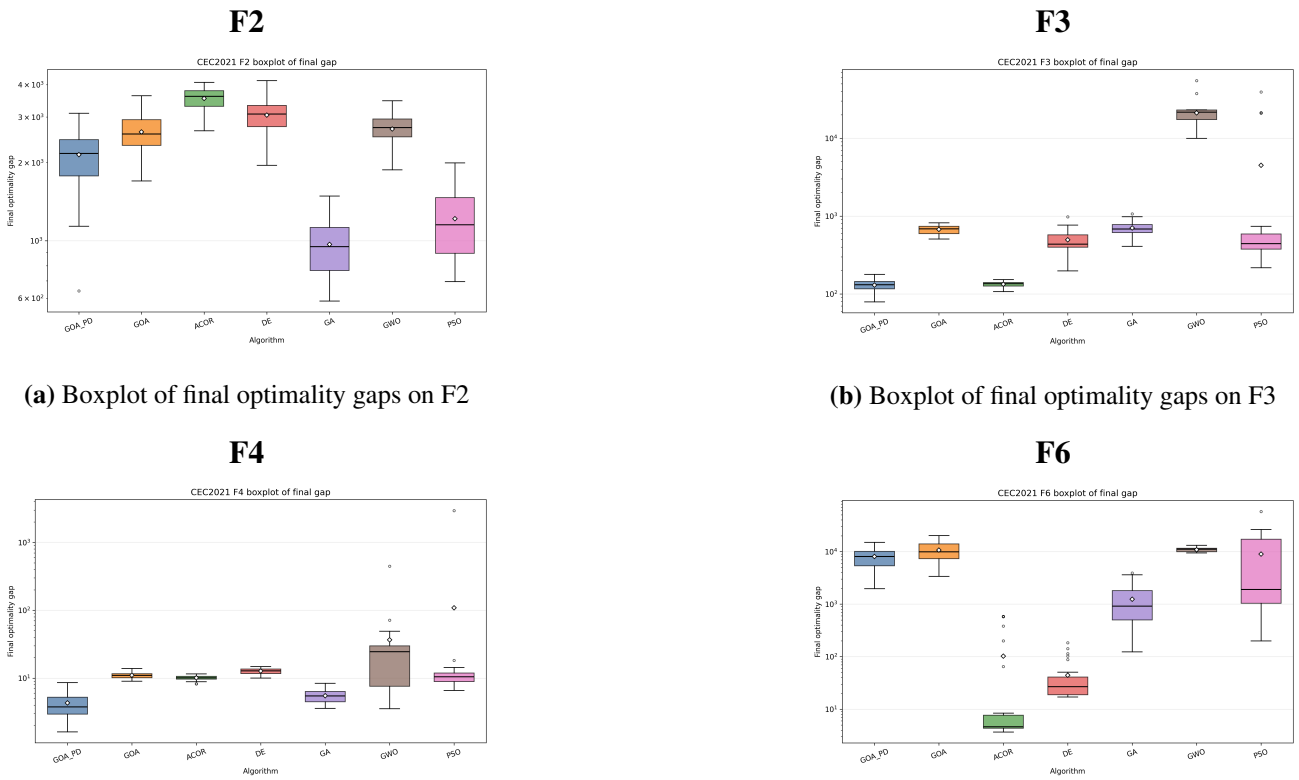
F10



(b) Convergence curve on F10

Figure 9. Convergence curves for CEC2021 functions F8 and F10.

Figures 10 and 11 show the remaining boxplots of final optimality gaps.



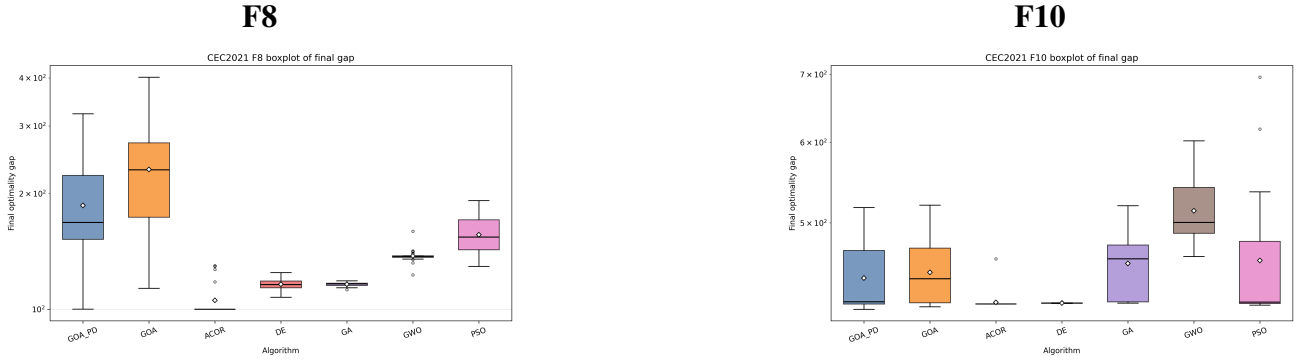
(a) Boxplot of final optimality gaps on F2

(b) Boxplot of final optimality gaps on F3

(c) Boxplot of final optimality gaps on F4

(d) Boxplot of final optimality gaps on F6

Figure 10. Boxplots of final optimality gaps for CEC2021 functions F2, F3, F4, and F6.



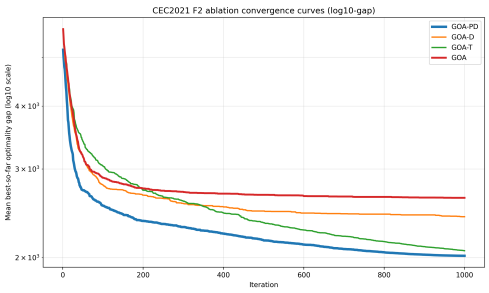
(a) Boxplot of final optimality gaps on F8

(b) Boxplot of final optimality gaps on F10

Figure 11. Boxplots of final optimality gaps for CEC2021 functions F8 and F10.

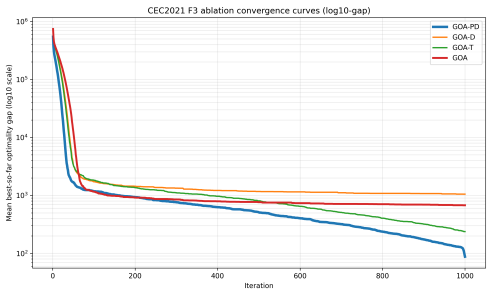
Figures 12 and 13 report the remaining ablation convergence curves.

F2



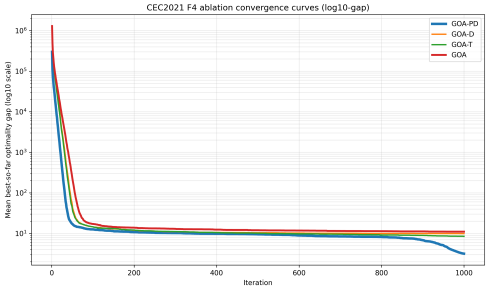
(a) Ablation convergence curves of the GOA-PD on F2

F3



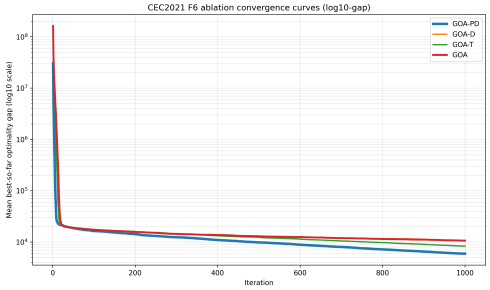
(b) Ablation convergence curves of the GOA-PD on F3

F4



(c) Ablation convergence curves of the GOA-PD on F4

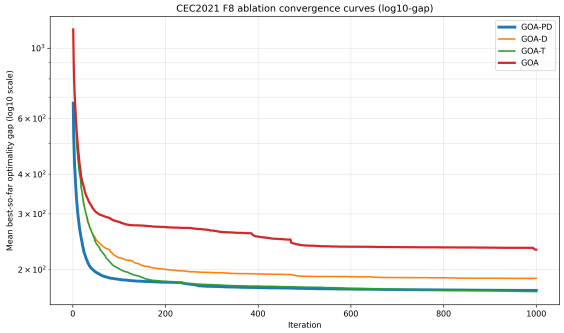
F6



(d) Ablation convergence curves of the GOA-PD on F6

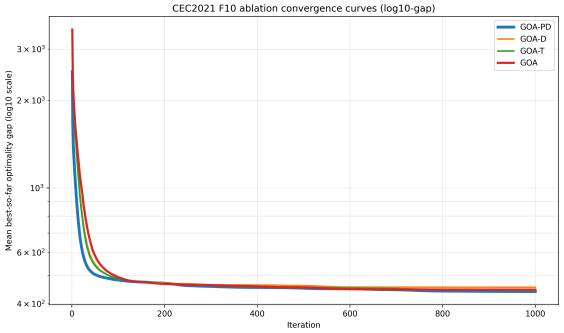
Figure 12. Ablation convergence curves of the GOA-PD for CEC2021 functions F2, F3, F4, and F6.

F8



(a) Ablation convergence curves of the GOA-PD on F8

F10



(b) Ablation convergence curves of the GOA-PD on F10

Figure 13. Ablation convergence curves of the GOA-PD for CEC2021 functions F8 and F10.

Table 10 presents the detailed Wilcoxon signed-rank test results.

Table 10. Wilcoxon signed-rank test results of the GOA-PD against each baseline.

Baseline	Function	GOA-PD Mean Gap	Baseline Mean Gap	Statistic	p-value	Decision
ACOR	F1	2028.191	6693.757	33	5.970×10^{-6}	+
	F2	2016.134	3543.457	1	3.730×10^{-9}	+
	F3	87.641	133.771	3	9.310×10^{-9}	+
	F4	3.165	10.108	0	1.860×10^{-9}	+
	F5	5.872×10^{4}	9.388×10^{5}	0	1.860×10^{-9}	+
	F6	5939.246	102.728	0	1.860×10^{-9}	-
	F7	5.956×10^{4}	4.975×10^{5}	0	1.860×10^{-9}	+
	F8	171.991	105.429	0	1.860×10^{-9}	-
	F9	214.969	207.052	28	2.760×10^{-6}	-
	F10	440.341	417.590	56	1.106×10^{-4}	-
DE	F1	2028.191	1.182×10^{7}	0	1.860×10^{-9}	+
	F2	2016.134	3051.193	0	1.860×10^{-9}	+
	F3	87.641	494.977	0	1.860×10^{-9}	+
	F4	3.165	12.683	0	1.860×10^{-9}	+
	F5	5.872×10^{4}	5.587×10^{4}	229	0.951526	=
	F6	5939.246	44.240	0	1.860×10^{-9}	-
	F7	5.956×10^{4}	1.052×10^{5}	86	0.001864	+
	F8	171.991	116.098	0	1.860×10^{-9}	-
	F9	214.969	376.883	0	1.860×10^{-9}	+
	F10	440.341	417.003	152	0.100397	=
GA	F1	2028.191	1.595×10^{7}	0	1.860×10^{-9}	+
	F2	2016.134	968.410	2	5.590×10^{-9}	-
	F3	87.641	703.617	0	1.860×10^{-9}	+
	F4	3.165	5.496	43	2.370×10^{-5}	+
	F5	5.872×10^{4}	1.130×10^{5}	75	7.296×10^{-4}	+
	F6	5939.246	1236.877	5	1.860×10^{-8}	-
	F7	5.956×10^{4}	4.179×10^{4}	114	0.013663	-
	F8	171.991	116.132	0	1.860×10^{-9}	-
	F9	214.969	491.383	0	1.860×10^{-9}	+
	F10	440.341	456.130	117	0.016431	+
GOA	F1	2028.191	1.453×10^{7}	0	1.860×10^{-9}	+
	F2	2016.134	2628.489	43	2.370×10^{-5}	+
	F3	87.641	674.810	0	1.860×10^{-9}	+
	F4	3.165	11.035	0	1.860×10^{-9}	+
	F5	5.872×10^{4}	8.710×10^{4}	106	0.008143	+
	F6	5939.246	1.064×10^{4}	38	1.220×10^{-5}	+
	F7	5.956×10^{4}	2.474×10^{5}	0	1.860×10^{-9}	+
	F8	171.991	231.134	79	0.001038	+
	F9	214.969	502.456	0	1.860×10^{-9}	+
	F10	440.341	446.972	132	0.038418	+
GWO	F1	2028.191	1.531×10^{9}	0	1.860×10^{-9}	+
	F2	2016.134	2697.535	27	2.350×10^{-6}	+
	F3	87.641	2.120×10^{4}	0	1.860×10^{-9}	+
	F4	3.165	36.607	0	1.860×10^{-9}	+
	F5	5.872×10^{4}	4.855×10^{5}	0	1.860×10^{-9}	+
	F6	5939.246	1.088×10^{4}	12	1.300×10^{-7}	+
	F7	5.956×10^{4}	3.392×10^{6}	0	1.860×10^{-9}	+
	F8	171.991	137.673	16	3.150×10^{-7}	-
	F9	214.969	1862.540	0	1.860×10^{-9}	+
	F10	440.341	514.016	3	9.310×10^{-9}	+
PSO	F1	2028.191	2.584×10^{8}	0	1.860×10^{-9}	+
	F2	2016.134	1215.457	14	2.050×10^{-7}	-
	F3	87.641	4483.985	0	1.860×10^{-9}	+
	F4	3.165	108.550	0	1.860×10^{-9}	+
	F5	5.872×10^{4}	1.920×10^{5}	38	1.220×10^{-5}	+
	F6	5939.246	8946.342	229	0.951526	=
	F7	5.956×10^{4}	4.098×10^{5}	46	3.450×10^{-5}	+
	F8	171.991	156.222	156	0.119077	=
	F9	214.969	2599.200	1	3.730×10^{-9}	+
	F10	440.341	459.142	180	0.289366	=