Networks and Heterogeneous Media

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

NHM, 20(5): 1466–1508
DOI: 10.3934/nhm.2025063
Received: 20 October 2025
Revised: 01 December 2025
Accepted: 15 December 2025
Published: 23 December 2025

Research article

An enhanced crossover strategy for the artificial lemming algorithm for engineering design optimization

Yan Zhong, Li-Bin Liu, Xiongfa Mai*and Haiyan Luo

Center for Applied Mathematics of Guangxi, Nanning Normal University, Nanning 530100, China

* **Correspondence:** Email: maixf@nnnu.edu.cn; Tel: +867716214028.

Abstract: To address the limitations of the artificial lemming algorithm (ALA) in convergence accuracy and premature convergence, this paper proposes an enhanced variant, the cross-strategy-integrated artificial lemming algorithm (CALA). Specifically, the CALA integrates a linear inertia weight strategy to balance exploration and exploitation, a historical best-guided strategy to enhance local search, and a crossover strategy to maintain population diversity. The proposed algorithm was evaluated on the IEEE CEC2017 and CEC2022 benchmark suites, achieving minimum Friedman mean ranks of 1.37 and 1.5, respectively, and outperforming several state-of-the-art algorithms in terms of accuracy and robustness. Furthermore, the CALA was successfully applied to constrained engineering design problems and a photovoltaic model parameter estimation task, demonstrating its effectiveness and practical applicability.

Keywords: artificial lemming algorithm; linear inertia weight strategy; historical best-guided strategy; crossover strategy; engineering design problems

1. Introduction

Metaheuristic algorithms (MAs) possess several advantages, including simple structures, few control parameters, independence from gradient information, and ease of implementation. These features enable MAs to effectively tackle complex optimization problems characterized by multimodality and high dimensionality, while avoiding entrapment in local optima. Compared with traditional methods, MAs demonstrate superior robustness and efficiency in solving extremum problems and real-world applications [1, 2]. Due to their capability to deliver reliable optimal or near-optimal solutions for various complex NP-hard problems within a reasonable computational time, MAs have gained widespread popularity in engineering optimization across diverse fields [3–5]. For instance, Chen et al. [6] proposed a basketball team optimization algorithm (BTOA), inspired by the dynamics and strategic features of basketball gameplay. Through simulation and real-world case studies, the BTOA has shown excellent performance in practical problems such as UAV path planning, which involve complex constraints and large decision spaces.

In response to increasingly challenging optimization problems, a growing number of novel MAs have emerged and are playing a pivotal role in the field of computational intelligence. However, in practice, the average performance of these algorithms in global optimization tasks is often comparable, and they generally suffer from common drawbacks such as low convergence accuracy, slow convergence speed, and a tendency to become trapped in local optima [7].

According to the "no free lunch" (NFL) theorem proposed by Wolpert and Macready [8], the development of new algorithms is crucial. Nevertheless, enhancing existing optimization algorithms to address complex problems has also become a prominent research focus [9]. As revealed by the literature, researchers typically adopt two main approaches to achieve this goal [10]: (1) Add one or more search strategies to the original algorithm. For example, to address the issues of falling into a local optimum in the crested porcupine optimizer (CPO), Liu et al. [11] proposed a multi-strategy enhanced crested porcupine optimizer (CAPCPO), which integrates a composite Cauchy mutation strategy, an adaptive dynamic adjustment strategy, and a population mutation strategy. Evaluated on the IEEE CEC2017 and CEC2022 benchmark suites, the CAPCPO demonstrated superior optimization performance compared to the original CPO. (2) Hybridize different MAs with the potential embedding of one or more search operators. For instance, Liu et al. [12] proposed a hybrid algorithm named the L-QBOA, which combines the butterfly optimization algorithm (BOA) with quantum-behavior particle swarm optimization (QPSO) algorithms, and establishes an inter-algorithm communication framework via the *gbest* evolution strategy. Experimental results on benchmark test functions and the Muskingum model confirmed that the L-QBOA possesses excellent robustness and optimization capability. This study adopts the first approach by refining the artificial lemming algorithm (ALA) to enhance its search capability and address complex optimization tasks more effectively.

The artificial lemming algorithm (ALA) [13] is a recently proposed population-based optimizer inspired by the collective migratory behavior of lemmings. Compared with many classical algorithms, the ALA features a simple structure, flexible migration dynamics, and strong global exploration capability. These properties make it a promising candidate for solving both constrained engineering problems and modern benchmark functions. However, despite its effectiveness, the ALA still suffers from several limitations: The original ALA exhibits insufficient convergence refinement due to the absence of an adaptive mechanism for dynamically adjusting search intensity, which may result in premature convergence or unstable performance across different test functions. Moreover, its exploitation ability remains weak, as the algorithm is highly sensitive to deviations in the search direction. Because the search process relies predominantly on random migration, the algorithm's ability to refine solutions in later iterations is substantially constrained. In addition, population diversity preservation is limited; the lack of population mixing operations makes the ALA susceptible to population degradation when facing complex fitness landscapes.

These issues have been reported or implied in several recent analyses of the ALA and similar emerging metaheuristics. In addition, the ALA's performance degradation becomes more apparent on functions with stronger multimodality or complex interaction structures. For more details, see the Section 2. Therefore, there remains substantial room to improve the ALA by designing mechanisms that enhance convergence stability, solution refinement, and population diversity while preserving the lightweight structure of the original algorithm. This forms the key motivation of the present work. To address these challenges, this paper proposes an enhanced variant of the ALA, named the cross-strategy-integrated artificial lemming algorithm (CALA). The proposed CALA integrates three strategies that are specifically adapted to the ALA's migration-based search dynamics:

• To address the insufficient search accuracy of individuals during the exploration phase, a linear inertia weight strategy is introduced to gradually regulate the aggressiveness of movement. This mechanism effectively adjusts the transition from exploration to exploitation, thereby improving the population's convergence accuracy.

• To overcome the reliance on random migration and the inadequate local refinement capability during the exploitation phase, a historical best-guided strategy is incorporated. By embedding the information of the global best individual into the migration step, the population is progressively guided toward high-quality regions. This enhances the stability of the iterative process, strengthens the local exploitation ability, and increases the search efficiency of individuals.

• To maintain population diversity and prevent the algorithm from being trapped in local optima, a bidirectional crossover strategy tailored to the ALA is designed and applied after each iteration. By appropriately recombining individuals at each iteration, this mechanism effectively enhances population diversity, breaks the structural constraints formed by local patterns, accelerates global convergence, and avoids premature stagnation.

These strategies are carefully incorporated into the ALA without disrupting its core behavioral mechanism, thereby improving convergence precision while maintaining simplicity and computational tractability.

The proposed CALA is comprehensively evaluated on the IEEE CEC2017 and CEC2022 test suites, as well as on four classical constrained engineering optimization problems and a photovoltaic model parameter estimation task. The experimental results demonstrate that the CALA achieves competitive performance compared with the original ALA and several well-established metaheuristic algorithms. In particular, the CALA provides significantly improved convergence accuracy and solution stability. The main contributions of this work are summarized as follows:

• To effectively address the slow convergence and premature stagnation of the original ALA, this study introduces three improvement strategies specifically designed to fit within the ALA framework.

• Extensive comparative experiments, together with Wilcoxon and Friedman statistical tests, demonstrate that the CALA exhibits significant advantages in both convergence speed and optimization accuracy.

• An analysis of the CALA's time complexity and its exploration–exploitation balance indicates that the algorithm maintains excellent convergence performance while incurring relatively low computational cost.

• Applications to four engineering design problems and a photovoltaic parameter identification task further show that the CALA consistently attains optimal solutions, highlighting its strong potential for practical deployment.

The remainder of this paper is organized as follows: Section 2 presents a systematic review of the literature related to improved variants of the ALA. Section 3 introduces the core mechanism of the original ALA. Section 4 discusses the proposed linear inertia weight strategy, historical best-guided strategy, and crossover strategy in detail, and formally describes the CALA — a multi-strategy enhanced ALA — along with its time complexity analysis. Section 5 provides experimental settings, performance analysis, and result discussions. Section 6 validates the practicality of the CALA through four constrained engineering design cases. In Section 7, we use the CALA to solve a real-world photovoltaic model parameter identification problem. Finally, Section 8 concludes the paper and highlights future research directions.

2. Related work

Metaheuristic optimization algorithms have been extensively studied due to their ability to handle highly nonlinear, multimodal, and black-box optimization problems. In recent years, numerous nature-inspired and evolutionary algorithms have been proposed, such as the KOA [14], HO [15], SAO [16], and many of their variants, including the CGKOA [17], IHO [18], and MSAO [19]. Despite their different biological inspirations, these algorithms fundamentally rely on the balance between global exploration and local exploitation. Consequently, a substantial body of research has focused on designing targeted enhancement strategies to improve convergence accuracy and robustness when solving complex optimization problems.

The ALA is a recently introduced population-based optimizer inspired by the migration, digging, and foraging behaviors of lemmings. Its simple search rules and flexible behavioral model provide a solid foundation for further methodological developments. However, as the ALA is still relatively new, the number of studies dedicated to its enhancement remains limited. Existing research on the ALA or ALA-based variants can generally be categorized into two groups: (1) applying the original ALA to specific optimization tasks, and (2) preliminary attempts to modify the ALA using heuristic components or hybrid strategies. For example, in [20], Han et al. employed the artificial lemming algorithm for feature selection and further refined the feature set using the ALA. Experimental results on seven high-dimensional datasets demonstrated the effectiveness and efficiency of the method, outperforming existing approaches. In [21], Zhu et al. introduced chaotic initialization, adaptive perturbation, and hybrid mutation to enhance the exploration–exploitation balance of the ALA. Their improved version achieved faster convergence and better performance than the standard ALA and ten competitor algorithms on CEC2017 and CEC2022 benchmarks, while generating efficient and collision-free UAV paths under realistic three-dimensional constraints. More studies on the ALA improvements are summarized in Table 1.

Overall, research focusing specifically on the ALA remains sparse, and systematic studies on mechanism-level enhancements are still lacking. This scarcity indicates a clear need for developing more refined and structurally grounded improvements for the ALA. Therefore, this work designs three targeted strategies to address the limitations of the ALA, aiming to enhance its convergence speed and improve the search efficiency of individuals. To address the insufficient search precision of individuals during the exploration phase, a linear inertia weight strategy is introduced. By adjusting the movement amplitude during migration or foraging, this strategy effectively regulates the transition from exploration to exploitation, thereby improving the convergence accuracy of the population. For the exploitation phase of the ALA, where the algorithm relies mainly on random migration—thus limiting its refinement capability in later iterations—we incorporate a historical best-guided strategy. This mechanism directs individuals to follow the global best with higher accuracy, significantly improving convergence stability and refinement strength. Finally, to maintain population diversity and generate promising candidate solutions, a crossover strategy is incorporated. Owing to the ALA's pairwise and group-based movement structure, crossover operations can effectively enhance population diversity, prevent premature convergence, and accelerate convergence speed.

By systematically integrating the three complementary strategies—linear inertia weight strategy, historical bestguided strategy, crossover strategy—into the ALA, we enhance its convergence accuracy, stability, and robustness while preserving its inherent search characteristics. The resulting improved algorithm, referred to as the CALA, is presented in this study. The performance of the CALA algorithm is comprehensively evaluated in subsequent sections.

Table 1. Review of the ALA algorithm.

Improved algorithm	Reference	Additional improvement strategies/algorithm	Application area
An enhanced artificial lemming algorithm (EALA)	[21]	Chaotic initialization, adaptive perturbation, and hybrid mutation	Applied to UAV path planning in large- and medium-scale environments with realistic obstacle constraints
An improved artificial lemming algorithm	[22]	The cubic chaotic map initialization, double adaptive t-distribution perturbation, and population dynamic optimization	Applied to the 3D route planning model
An improved bionic artificial lemming algorithm	[23]	A collaborative second-order Bernstein polynomial and chaotic mapping function initialization strategy (BPSC), a quadratic interpolation random mutation strategy, and an adaptive evolutionary strategy	Applied to the CEC2021 problem and the cloud task-scheduling problem
The artificial lemming algorithm (ALA)	[24]	/	Predict the hydrogen storage capacity of MOFs
An enhanced artificial lemming algorithm (DMSALAs)	[25]	A dynamic adaptive mechanism, a hybrid Nelder–Mead method, and a localized perturbation strategy	Mobile robot path planning scenarios
An improved ALA (GALA)	[26]	Applies ReliefF-guided population initialization and a hybrid search strategy combining historical optimal positions with sine cosine optimization (SCA)	Feature selection of multiple biomedical datasets

3. Artificial lemming algorithm (ALA)

In this section, we review the mathematical model of the original artificial lemming algorithm (ALA). The algorithm, a novel bio-inspired optimization algorithm proposed by Xiao et al. in 2025, is inspired by four behaviors of lemmings in nature: long-distance migration, digging holes, foraging for food, and evading natural predators. Based on these behaviors, the ALA is divided into three phases: initialization phase, exploration phase, and exploitation phase. Among them, the long-distance migration and digging behaviors are used for exploration of the searched space, and the foraging and predator evasion behaviors are used for exploitation of the searched space.

3.1. Initialization phase

As with other MAs, the iteration procedure of the ALA commences with a randomly generated population. Assuming that the dimensional size of the optimization problem is *Dim*, the lower and upper boundaries of the search domain are Lb and Ub, respectively, and the number of search agents is N. Then the initial position of the entire population can be modeled as a matrix with N rows and Dim columns.

$$X = Lb + rand \times (Ub - Lb), \tag{3.1}$$

where $rand$ indicates a random number in $[0, 1]$. Usually, the position vector of the i_{th} search agent is expressed as: $X_i = (X_{i,1}, X_{i,2}, ..., X_{i,j}, ..., X_{i,Dim}), i = 1, 2, ..., N; j = 1, 2, ..., Dim$.

3.2. Exploration phase

3.2.1. Long-distance migration

The first behavior of lemmings is to randomly undertake long-distance migration, where lemmings explore the search space based on their current position and the position of random individuals in the population to find habitats with abundant food resources, updating the lemming positions according to Eq (3.2).

$$X_i(t+1) = X_{best}(t) + F \times \overrightarrow{BM} \times (\overrightarrow{R} \times (X_{best}(t) - X_i(t)) + (1 - \overrightarrow{R}) \times (X_i(t) - X_a(t))), \tag{3.2}$$

where $X_i(t+1)$ denotes the position of the i_{th} search agent at iteration $t+1$, and $X_{\text{best}}(t)$ represents the current optimal solution. F is served as a flag, whose calculation is defined in Eq (3.4). $\overrightarrow{BM}$ is a random vector generated by Brownian motion, as shown in Eq (3.3). $\overrightarrow{R}$ is a $1 \times Dim$ vector whose elements are uniformly distributed random numbers within the interval $[-1, 1]$, and it is generated according to Eq (3.5). $X_i(t)$ indicates the current position of the i_{th} search agent, while $X_a(t)$ represents a randomly selected individual from the population, with a being an integer index in the range $[1, N]$.

$$f_{\overrightarrow{BM}}(x; 0, 1) = \frac{1}{\sqrt{2\pi}} \times exp(-\frac{x^2}{2}), \tag{3.3}$$

$$F = \begin{cases} 1, & if\lfloor 2 \times rand + 1 \rfloor = 1 \\ -1, & if\lfloor 2 \times rand + 1 \rfloor = 2, \end{cases} \tag{3.4}$$

$$\overrightarrow{R} = 2 \times rand(1, Dim) - 1, \tag{3.5}$$

where $\lfloor \cdot \rfloor$ means the floor function, for rounding down.

3.2.2. Digging holes

The second behavior of lemmings is digging burrows, which helps lemmings to quickly escape from predators and find food more efficiently. The behavior is simulated in Eq (3.6).

$$X_i(t+1) = X_i(t) + F \times L \times (X_{best}(t) - X_b(t)), \tag{3.6}$$

$$L = rand \times (1 + sin(\frac{t}{2})), \tag{3.7}$$

$X_b(t)$ designates a search individual randomly selected.

3.3. Exploitation phase

3.3.1. Foraging for food

In the third behavior, random foraging behavior, the lemming will wander randomly within their foraging area in order to intake as much food as possible. The spiral wrapping mechanism is simulated in Eq (3.8).

$$X_i(t+1) = X_{best}(t) + F \times spiral \times rand \times X_i(t), \tag{3.8}$$

$$spiral = radius \times (sin(2 \times \pi \times rand) + cos(2 \times \pi \times rand)), \tag{3.9}$$

$$radius = \sqrt{\sum_{j=1}^{Dim} (X_{best,j}(t) - X_{i,j}(t))^2}. \tag{3.10}$$

3.3.2. Evading natural predators

In the final phase, the modeling primarily emphasizes the evasion and protective behaviors exhibited by lemmings upon encountering threats. The corresponding mathematical expression is shown in Eq (3.11).

$$X_i(t+1) = X_{best}(t) + F \times G \times Levy(Dim) \times (X_{best}(t) - X_i(t)), \tag{3.11}$$

$$G = 2 \times (1 - \frac{t}{T}), \tag{3.12}$$

where G is the escape coefficient of lemmings, t indicates the current number of iterations, and T indicates the maximum number of iterations. $Levy(\cdot)$ is the Levy flight function.

4. Proposed CALA algorithm

4.1. Linear inertia weight strategy

The inertia weight strategy is a mechanism used to control the exploration and exploitation behavior of particles during the search process. In [27–29], a linearly inertia weight strategy was introduced and demonstrated to be effective in enhancing the fine-tuning capability of the PSO. Inspired by the principles underlying most particle swarm optimization algorithms, this paper addresses the limitation of low search accuracy among individuals in the exploration phase of the ALA framework by effectively incorporating a linear inertia weight strategy into the ALA. This strategy progressively adjusts the aggressiveness of movement, thereby enabling a smooth transition from exploration to exploitation. In this method, the value of the inertia weight w is dynamically adjusted based on the iteration number. Specifically, w decreases linearly from an initial value of 1 to a final value of 0.5 according to Eq (4.1), as shown in Figure 1. The sensitivity of its parameters is detailed in Section 5.2.1. The modified search equations are provided in Eqs (4.2) and (4.3):

$$w = (1 - t/T) \times 0.5 + 0.5, \tag{4.1}$$

$$X_i(t+1) = X_{best}(t) + w \times F \times \overrightarrow{BM} \times (\overrightarrow{R} \times (X_{best}(t) - X_i(t)) + (1 - \overrightarrow{R}) \times (X_i(t) - X_a(t))), \tag{4.2}$$

$$X_i(t+1) = X_i(t) + w \times F \times L \times (X_{best}(t) - X_b(t)). \tag{4.3}$$

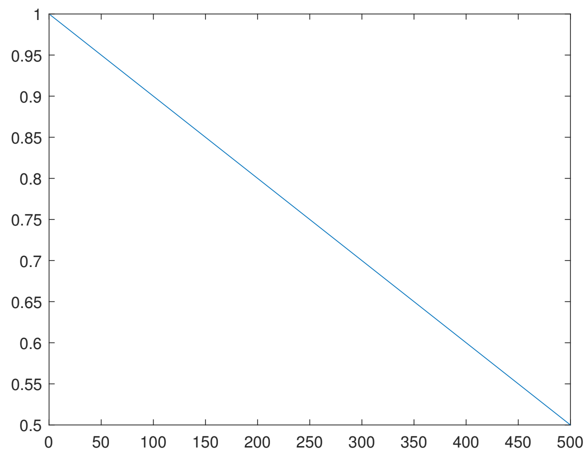

Figure 1. Trends of the dynamic parameter w.

In the above, t denotes the current iteration number, and T represents the maximum number of iterations. When w is relatively large, particles tend to perform global exploration, allowing the algorithm to explore new regions and accelerate convergence. Conversely, when w is small, particles are more inclined toward local exploitation, refining known regions and promoting stability in the later stages of the search, which contributes to improved convergence accuracy.

4.2. Historical best-guided strategy

In the ALA, the position update of individual particles during the exploitation phase relies on stochastic migration and exhibits a limited local refinement capability, which may consequently lead to a reduction in the algorithm's convergence speed. To address this issue, the guidance of the best individual in the current population plays a crucial role [30]. With the assistance of the historical best position, the remaining individuals are attracted toward it and tend to converge in the neighborhood of the optimal solution [31, 32]. To enhance the exploration efficiency of individuals, the exploitation phase no longer relies on random search. Instead, particle updates are performed based on the historical-best solution. The modified search equations are provided in Eqs (4.4) and (4.5):

$$X_i(t+1) = X_{best}(t) + F \times spiral \times rand \times X_i(t) + k \times (best\_history(t) - X_i(t)), \tag{4.4}$$

$$X_i(t+1) = X_{best}(t) + F \times G \times Levy(Dim) \times (X_{best}(t) - X_i(t)) + k \times (best\_history(t) - X_i(t)). \tag{4.5}$$

Here, k is the control parameter of the historical best-guided strategy, set to 0.2 (specific parameter sensitivity experiments are detailed in Section 5.2.1), $best\_history$ represents the history-best position, and $X(t + 1)$ is the new position obtained from $X(t)$.

4.3. Crossover strategy

During the iterative process of the ALA, the positions of the agents gradually converge toward the global best direction within the current population. However, if the global leader falls into a local optimum, the algorithm may suffer from premature convergence. As the number of iterations increases, the diversity within the population tends to decrease, thereby reducing the overall accuracy of the algorithm. In response to these limitations, we deliberately incorporate a crossover strategy [33] to moderately reorganize individuals after each iteration. This approach effectively enhances population diversity, breaks structural constraints formed by local patterns, accelerates global convergence, and helps avoid premature convergence to local optima.

The proposed crossover strategy consists of two components: horizontal crossover and vertical crossover. The mathematical models for horizontal crossover are presented in Eqs (4.6) and (4.7):

$$X^{hc}_{i1,j} = r_1 \times X_{i1,j} + (1 - r_1) \times X_{i2,j} + c_1 \times (X_{i1,j} - X_{i2,j}), \tag{4.6}$$

$$X^{hc}_{i2,j} = r_2 \times X_{i2,j} + (1 - r_2) \times X_{i1,j} + c_2 \times (X_{i2,j} - X_{i1,j}). \tag{4.7}$$

Among them, $X_{i1,j}$ and $X_{i2,j}$ represent the j_{th} dimension of X_{i1} and X_{i2}, respectively, $j = 1, 2, 3...dim$, $X^{hc}_{i1,j}$ and $X^{hc}_{i2,j}$ represent the j_{th} dimension where X_{i1} and X_{i2} cross horizontally to produce offspring on the j_{th} dimension, r_1 and r_2 represent uniformly distributed random numbers within the range of $(0, 1)$, and c_1 and c_2 are uniformly distributed random numbers within the range of $(-1, 1)$. The generated offspring compete with their parents to ultimately retain the optimal individual.

The model for vertical crossover is given in Eq (4.8), where the two dimensions j_1 and j_2 are randomly selected, and the j_1 dimension of their descendant X^{vc}_i is obtained by the following Eq (4.8), while the other dimensions remain the same as those of the parent.

$$X^{vc}_{i,j} = r \times X_{i,j1} + (1 - r) \times X_{i,j2}, \tag{4.8}$$

$$r = rand.$$

The crossover strategy enables individuals to learn from each other, enhancing population diversity and mitigating premature convergence. Moreover, it helps to overcome stagnation in specific dimensions, thereby enabling the algorithm to escape from local optima.

The flowchart of the suggested CALA is presented in Figure 2. The detailed procedural steps of the CALA are outlined below in Algorithm 1.

Algorithm 1 Pseudocode of the CALA

```
Input: Population size $N$, dimension size $Dim$, current iteration $t$, maximum iterations $T$, supremum
    $Ub$, infimum $Lb$;
Output: Global optimal solution $X_{best}$ and its fitness value $fitness_{best}$;
 1: set $N = 100, T = 500$;
 2: Randomly initialize the position of all lemming individuals $X_i (i = 1, 2, ..., N)$;
 3: Evaluate the fitness values of all lemming individuals;
 4: Record the current optimal solution $X_{best}$;
 5: $t = 1$;
 6: while $t < T$ do
 7:     Calculate the value of $E$ using Eq (5.1) and the value of $w$ using Eq (4.1);
 8:     for $i = 1 : N$ do
 9:         if $E > 1$ then
10:             Exploration:
11:             %Accelerating algorithm convergence by integrating a linear weight decay strategy
    during the exploration phase.%
12:             if $rand < 0.3$ then
13:                 Calculate new status of the $i_{th}$ lemming using Eq (4.2);
14:             else
15:                 Calculate new status of the $i_{th}$ lemming using Eq (4.3);
16:             end if
17:         else
18:             Exploitation:
19:             %During the exploitation phase, introduce historically optimal individuals to guide the
    population, thereby enhancing individual search efficiency.%
20:             if $rand < 0.5$ then
21:                 Calculate new status of the $i_{th}$ lemming using Eq (4.4);
22:             else
23:                 Calculate new status of the $i_{th}$ lemming using Eq (4.5);
24:             end if
25:         end if
26:     end for
27:     Recalculate fitness values of all lemming individuals;
28:     Update the optimal solution obtained so far $X_{best}$;
29:     %Introducing crossover strategies helps populations escape local optima.%
30:     Calculate new status of the $i_{th}$ lemming individuals using Eqs (4.6)–(4.8);
31:     Calculate the new population generated by the crossover strategy and select the optimal
    particles to enter the next iteration;
32:     Save best candidate solution so far.
33:     t = t + 1;
34: end while
```

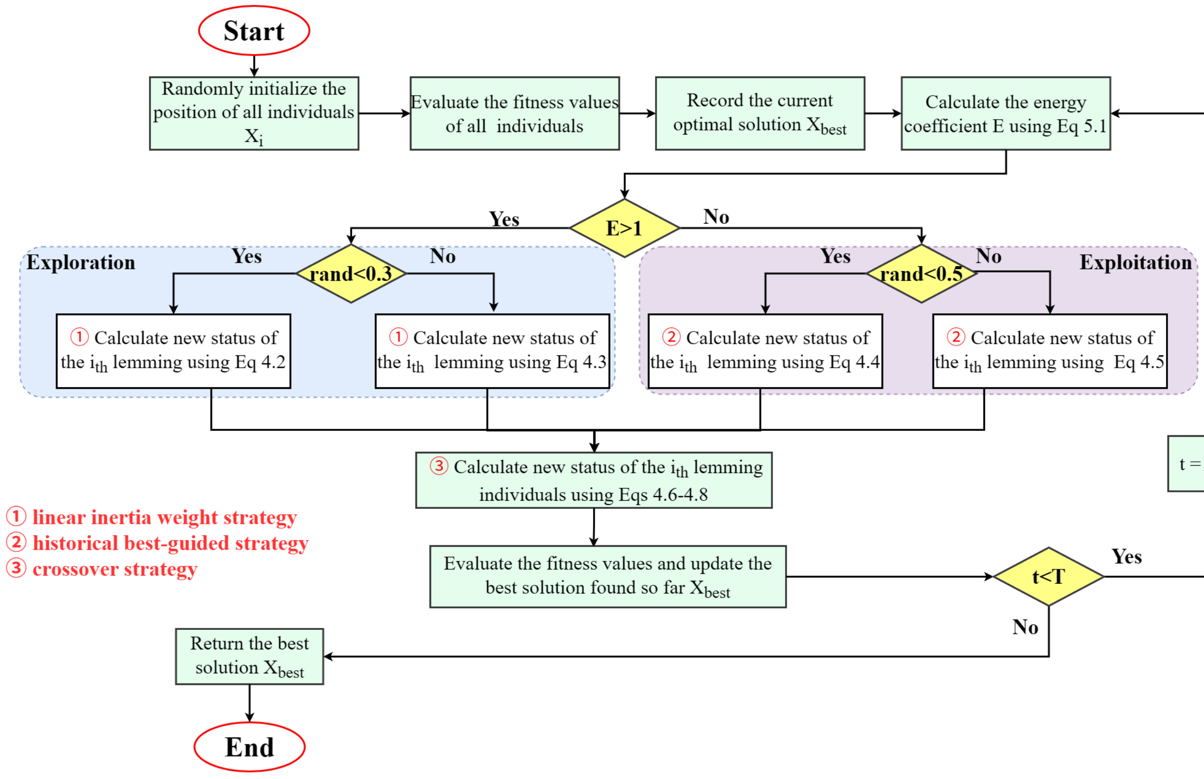
Start
Randomly initialize the position of all individuals X_i
Evaluate the fitness values of all individuals
Record the current optimal solution X_{best}
Calculate the energy coefficient E using Eq 5.1
E>1
Yes
No
Exploration
rand<0.3
Yes
No
① Calculate new status of the i_{th} lemming using Eq 4.2
① Calculate new status of the i_{th} lemming using Eq 4.3
Exploitation
rand<0.5
Yes
No
② Calculate new status of the i_{th} lemming using Eq 4.4
② Calculate new status of the i_{th} lemming using Eq 4.5
③ Calculate new status of the i_{th} lemming individuals using Eqs 4.6-4.8
① linear inertia weight strategy
② historical best-guided strategy
③ crossover strategy
Evaluate the fitness values and update the best solution found so far X_{best}
t<T
Yes
No
t = t + 1
Return the best solution X_{best}
End

Figure 2. Flowchart of the proposed CALA.

4.4. CALA complexity analysis

The computational time varies across algorithms when solving the same problem, making the evaluation of computational complexity essential for enhancing execution efficiency. The computational complexity of the CALA is primarily influenced by four factors: initialization $O(N \times Dim)$; position updates of individuals $O(T \times N \times Dim)$; the implementation of horizontal and vertical crossover strategies, also $O(T \times N \times Dim)$; and fitness evaluations $O(T \times N)$, where T denotes the maximum number of iterations, N is the number of search agents, and Dim represents the problem dimensionality. Thus, the cumulative computational complexity of the CALA can be approximated as $O(N \times Dim) + O(T \times N \times Dim) + O(T \times N \times Dim) + O(T \times N)$.

Additionally, all experiments were performed on a laptop computer with an Intel(R) Core(TM) i5-10210U CPU @1.60GHz, and all programs were coded on MATLAB R2018B and a later version.

4.5. Average CPU time comparison

In the field of intelligent optimization, comparing computational time is essential for evaluating algorithmic efficiency. Table 2 reports the average CPU time of several metaheuristics, including GWO, AVAO, and the proposed CALA, on twelve CEC2022 benchmark functions. Although algorithms such as the COA and HHO exhibit shorter runtimes, they suffer from notable shortcomings: slow or unstable convergence, a tendency to prematurely stagnate, and limited capability to escape local optima in complex landscapes. As a result, their solution accuracy degrades significantly on challenging optimization problems. In contrast, the CALA—while marginally slower than the simplest algorithms—demonstrates clear advantages. It remains substantially faster than advanced optimizers such as LSHADE, whose sophisticated parameter-adaptation and differential-mutation mechanisms impose a much heavier computational burden. Benefiting from enhanced search dynamics and refined exploitation strategies, the CALA is capable of performing more comprehensive exploration and delivering higher-precision and more-reliable solutions. The slight increase in computational time is therefore well justified by the considerable improvement in optimization accuracy, especially in scenarios where precision is critical.

Moreover, as the complexity of CEC2022 functions increases, the CALA's performance advantage becomes more pronounced. It maintains stability and convergence quality even as other algorithms exhibit performance deterioration. These characteristics highlight the CALA's robustness and its suitability for tackling complex, real-world optimization tasks. Therefore, while computational efficiency is an important consideration, the superior solution quality and strong adaptability of the CALA make it a highly competitive algorithm in the intelligent optimization domain.

Table 2. Average CPU time comparison of all algorithms (Unit: seconds).

Function	PSO	SCA	COA	HHO	LCA	LSHADE	ALA	CALA
F1	0.191981	0.200084	0.363613	0.537173	0.160935	1.081820	0.293381	0.443411
F2	0.210017	0.186891	0.359721	0.536337	0.181179	1.586240	0.288953	0.485138
F3	0.303347	0.296095	0.567749	0.766610	0.251148	0.797553	0.372948	0.750006
F4	0.217648	0.246265	0.397310	0.622895	0.198534	1.700162	0.309670	0.573138
F5	0.232040	0.251206	0.432935	0.650438	0.200381	1.902792	0.322188	0.577003
F6	0.195523	0.218977	0.383538	0.581321	0.179296	1.252991	0.295440	0.475906
F7	0.315525	0.365801	0.689914	0.914754	0.312766	1.435736	0.395339	0.973447
F8	0.389885	0.404439	0.827080	1.024635	0.374136	1.558543	0.468387	1.096237
F9	0.295087	0.326805	0.607409	0.751693	0.281775	1.608254	0.372069	0.798477
F10	0.262089	0.297255	0.568536	0.948898	0.268335	1.435040	0.343232	0.783668
F11	0.342542	0.392808	0.772676	0.974207	0.340443	1.523989	0.418357	1.011575
F12	0.356555	0.399711	0.934674	0.934674	0.360005	1.885389	0.438407	1.056505

5. Experimental results and analyses

5.1. Competing algorithms and parameter settings

To comprehensively assess the performance of the proposed CALA, comparative experiments were conducted on the CEC2017 and CEC2022 benchmark suites against seven representative algorithms. These algorithms are categorized into four groups: (1) the original algorithm: artificial lemming algorithm (ALA) [13]; (2) classical and widely-cited algorithms: particle swarm optimization (PSO) [34], sine cosine algorithm (SCA) [35]; (3) recently developed advanced algorithms: crayfish optimization algorithm (COA) [36], harris hawks optimization (HHO) [37], and liver cancer algorithm (LCA) [38]; and (4) the champion algorithm: success-history based adaptive DE with linear population size reduction (LSHADE) [39]. The parameter settings for each algorithm follow the recommendations provided in the corresponding literature, as summarized in Table 3. For fair comparison, the population size and the maximum number of iterations were uniformly set to 100 and 500, respectively. Each algorithm was executed independently 30 times, and the resulting performance data are reported and analyzed in the subsequent subsections.

Table 3. Comparison of the parameter settings of the algorithms.

Algorithms	Reference	year	Name of the parameter	Value of the parameter
PSO	[34]	1995	w, c_1, c_2	[0.9,0.4], 0.2, 0.2
SCA	[35]	2016	α	2
COA	[36]	2023	$temp$	[20,35]
HHO	[37]	2019	E_0	(-1,1)
LCA	[38]	2023	f	1
LSHADE	[39]	2014	M_{CR}, M_F	0.5, 0.5
ALA	[13]	2025	F	-1 or 1
CALA	/	2025	w, k	[1,0.5], 0.2

5.2. Parameter sensitivity analysis

5.2.1. Analysis of the impact of the linear ineria weight bounds (w) and the historical best-guided coefficient (k)

In the CALA, the linear ineria weight bounds (w) and historical best-guided coefficient (k) are introduced. Both parameters are compared with random numbers in the algorithm, and the lemming then selects the appropriate formula to update its position. This mechanism allows the algorithm to introduce a certain degree of randomness during the search process, helping to avoid getting trapped in local optima. To ensure that the CALA can demonstrate optimal search capability, a sensitivity analysis of the parameter selection for w and k is conducted in this section. As shown in Table 4, six derivative algorithms are set with different values for w and k. For example, CALA1 is configured with $w = [1, 0.5]$ and $k = 0.1$, and the other derivative algorithms follow similar parameter settings.

Table 4. Parameter combination settings for w and k.

Parameters	CALA1	CALA2	CALA3	CALA4	CALA5	CALA6
w	[1,0.5]	[1,0.5]	[0.8,0.3]	[0.8,0.3]	[0.6,0.1]	[0.6,0.1]
k	0.1	0.2	0.1	0.2	0.1	0.2

Table 5 presents the experimental results of six CALA variants on 12 CEC2022 benchmark functions, obtained by averaging the results from 30 independent runs on the 10-dimensional CEC2022 test functions. It can be observed that CALA2 exhibits superior optimization performance across multiple test functions, ranking within the top one for F1–6, F9, and F12, demonstrating strong convergence capability. Additionally, CALA2 achieves significantly lower standard deviations on several functions compared to other derivative algorithms, indicating its high stability. For unimodal functions (F1) and multimodal functions (F2–F5), CALA2 consistently ranks among the top performing variants, which suggests that CALA2 possesses strong problem solving ability for both unimodal and multimodal optimization problems, along with excellent global exploration capability. Regarding hybrid functions (F6–F8) and composition functions (F10–F12), CALA2 attains competitive mean values and rankings, highlighting its robustness in high-dimensional search spaces while maintaining superior convergence stability compared to other CALA variants. Therefore, in the CALA, the range of parameter w selection is $[1, 0.5]$, and the value of parameter k is 0.2. Figure 3 illustrates the Friedman mean rankings of all CALA variants across the 12 benchmark functions.

Table 5. Experimental results for 12 CEC2022 functions at different combination values of w and k.

Function	Metric	CALA1	CALA2	CALA3	CALA4	CALA5	CALA6
F1	Ave	**300**	**300**	**300**	**300**	**300**	**300**
	Std	4.930E-13	1.455E-13	2.577E-13	5.013E-14	6.564E-14	**4.641E-14**
	Rank	**1**	**1**	**1**	**1**	**1**	**1**
F2	Ave	403.52522	**403.12675**	404.29670	406.97036	405.66416	404.804863
	Std	4.084E+00	**2.737E+00**	3.614E+00	3.685E+00	3.717E+00	3.871E+00
	Rank	2	**1**	3	6	5	4
F3	Ave	**600**	**600**	**600**	600.00001	600.00000	600.00001
	Std	4.523E-07	4.494E-07	**1.123E-07**	4.718E-05	3.550E-06	4.646E-05
	Rank	**1**	**1**	**1**	4	2	3
F4	Ave	806.89714	**805.91948**	806.80585	807.20337	807.56196	806.66622
	Std	2.643E+00	2.934E+00	2.111E+00	2.197E+00	**1.828E+00**	2.571E+00
	Rank	4	**1**	3	5	6	2
F5	Ave	900.00895	**900**	900.02685	900	900.08124	900.07229
	Std	2.831E-02	**9.282E-14**	4.325E-02	1.366E-13	1.975E-01	1.698E-01
	Rank	2	**1**	3	**1**	5	4
F6	Ave	1801.51670	**1801.51465**	1803.40107	1801.69674	1803.41744	1808.33601
	Std	1.288E+00	**5.085E-01**	2.788E+00	1.301E+00	2.891E+00	1.342E+01
	Rank	2	**1**	4	3	5	6
F7	Ave	2002.89605	2003.78764	**2001.65888**	2004.82645	2008.46825	2002.73443
	Std	6.042E+00	3.476E+00	**2.404E+00**	8.125E+00	1.007E+01	5.149E+00
	Rank	3	4	**1**	5	6	2
F8	Ave	2208.56328	2207.94843	2207.12875	2206.47208	2211.43649	**2203.57019**
	Std	7.580E+00	7.127E+00	8.552E+00	**5.590E+00**	9.242E+00	6.113E+00
	Rank	5	4	3	2	6	**1**
F9	Ave	**2529.28438**	**2529.28438**	**2529.28438**	**2529.28438**	**2529.28438**	**2529.28438**
	Std	**0**	**0**	**0**	**0**	**0**	**0**
	Rank	**1**	**1**	**1**	**1**	**1**	**1**
F10	Ave	2500.28582	2500.30604	2500.28776	2500.27144	2510.86189	**2500.24130**
	Std	4.747E-02	5.809E-02	**2.929E-02**	7.657E-02	3.357E+01	4.997E-02
	Rank	3	5	4	2	6	**1**
F11	Ave	**2630**	2660	2675.04271	2660	2720	2735.04271
	Std	9.487E+01	**1.265E+02**	1.275E+02	1.265E+02	1.549E+02	1.492E+02
	Rank	**1**	2	3	2	4	5
F12	Ave	2860.92799	**2860.89526**	2861.86174	2861.58707	2861.99619	2862.61034
	Std	1.420E+00	**1.279E+00**	1.548E+00	1.284E+00	1.288E+00	1.850E+00
	Rank	2	**1**	4	3	5	6

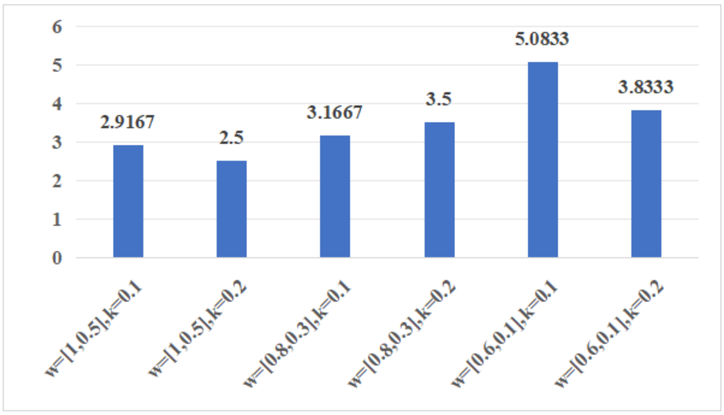

Figure 3. Friedman mean ranking for different w and k combination values on 12 CEC2022 functions.

5.2.2. Analysis of the impact of the switching probability p

In nature, long-distance migration of lemmings is not very frequently observed, so we set the switching probability $p = 0.3$ during the modeling process. If $rand < 0.3$, the long-distance migration behavior is executed; otherwise, the digging strategy is executed. Here, $rand$ is a random number between 0 and 1. This means that the algorithm will mainly perform regular exploration, interspersed with occasional large-scale random perturbations to force a restart of the exploration, which helps to overcome local optima and improve the convergence accuracy of the CALA. In this subsection, a parameter sensitivity analysis is conducted to verify the impact of different switching probability values on the performance of the CALA. Theoretically, the switching probability p is in the range of (0, 1). In the experiment, a value is taken every 0.1, and p is set to $0.1, 0.2, 0.3, ...,$ and 0.9, respectively.

Table 6. Optimization performance ranking results for the CALA with different probability values.

Function	$p = 0.1$	$p = 0.2$	$p = 0.3$	$p = 0.4$	$p = 0.5$	$p = 0.6$	$p = 0.7$	$p = 0.8$	$p = 0.9$
F1	1	1	1	1	1	1	1	1	1
F2	1	2	7	5	4	9	3	8	6
F3	5	4	1	2	1	1	1	1	3
F4	9	7	1	6	3	2	5	4	8
F5	1	1	1	1	2	3	2	4	2
F6	1	2	3	2	4	6	5	8	7
F7	6	3	1	5	2	4	7	8	9
F8	7	2	1	3	4	6	8	5	9
F9	1	1	1	1	1	1	1	1	1
F10	2	5	3	4	6	9	1	7	8
F11	1	3	1	5	2	3	3	4	6
F12	1	2	4	3	5	6	7	9	8
Mean rank	3.00	2.75	2.08	3.17	2.92	4.25	3.67	5.00	5.67
Final Ranking	4	2	1	5	3	7	6	8	9

Table 6 presents the ranking results of the CALA under different switching probability values on each 10-dimensional CEC2022 benchmark function, obtained based on the mean fitness over 10 independent runs. When $p = 0.3$, the CALA achieves the optimal solution on 8 out of the 12 functions and attains the best overall ranking. When $p > 0.3$, the optimization performance of the ALA deteriorates. This is because frequent long-distance migration induced by a larger switching probability tends to reduce search efficiency. Once the algorithm enters a local optimum, it becomes more susceptible to premature convergence, ultimately lowering convergence accuracy. Overall, the experimental results indicate that $p = 0.3$ provides the most appropriate balance between exploration and exploitation for the CALA. To ensure consistent convergence behavior and algorithmic stability, the switching probability p is therefore fixed to 0.3 in all subsequent experiments.

5.3. Analysis of exploration and exploitation in the CALA

Among metaheuristic algorithms, exploration can be defined as a global search that aims to discover better solutions in the global domain. In contrast, exploitation refers to local search that focuses on finding better solutions in a known smaller area. Balancing exploration and exploitation is of crucial importance for the adaptability and robustness of the algorithm. To maintain a robust balance between exploration and exploitation, an energy factor is designed to decrease over the course of iterations. The calculation formula for the energy factor is provided as follows:

$$E(t) = 4 \times arctan[1 - \frac{t}{T}] \times ln(\frac{1}{rand}). \tag{5.1}$$

Therefore, Hussain et al. [40] proposed a method for measuring dimensional diversity. Eqs (5.2) and (5.3), respectively, calculate the percentages of exploration and exploitation. Moreover, $Div(t)$ is a measure of dimensional diversity calculated from Eq (5.4).

$$Exploration(\%) = \frac{Div(t)}{Div_{max}} \times 100, \tag{5.2}$$

$$Exploitation(\%) = \frac{|Div(t) - Div_{max}|}{Div_{max}} \times 100, \tag{5.3}$$

$$Div(t) = \frac{1}{Dim} \sum_{d=1}^{Dim} \frac{1}{N} \sum_{i=1}^{N} |median(x_d(t)) - x_{id}(t)| \,. \tag{5.4}$$

Here, x_{id} represents the position of the i_{th} represents the position of the d_{th} dimension, and Div_{max} denotes the maximum diversity throughout the entire iteration process. $median(x_d(t))$ indicates the median of the d_{th} dimension.

Figure 4 shows the simulation results of the CALA on six 10-dimensional benchmark functions from the CEC2022. According to the literature [41], the best balance between exploration and exploitation is 10% exploration and 90% exploitation. When this balance is reached, the algorithm performs best. As shown in Figure 4, in the early stage of the search process, the proportion of exploration and exploitation of the CALA is large. In the later stage, the proportion of exploration decreased to less than 10%, while the proportion of exploitation increased to more than 90%, which significantly improved the convergence accuracy of the algorithm. This is also the reason why the

CALA can converge faster on all test functions. In conclusion, the CALA shows a good balance between exploration and exploitation.

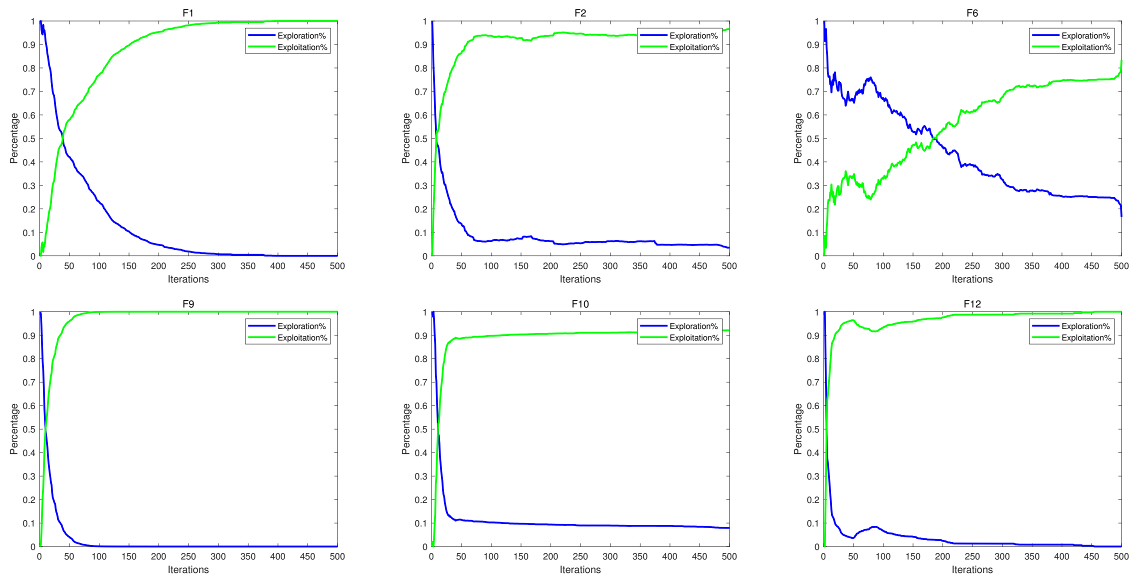

Figure 4. Balance between exploration and exploitation.

5.4. *Effectiveness validation of different components*

Table 7. Different CALA-derived variants with three improvement strategies.

Strategy	CALA-1	CALA-2	CALA-3	CALA-4	CALA-5	CALA-6	CALA
Linear inertia weight strategy	1	0	0	1	1	0	1
Historical best-guided strategy	0	1	0	1	0	1	1
Crossover strategy	0	0	1	0	1	1	1

To enhance the optimization capability of the conventional ALA, three effective strategies are integrated: the linear inertia weight strategy, historical best-guided strategy, and crossover strategy. To confirm the effectiveness of each component, six variants of the ALA are developed, each incorporating different combinations of the proposed strategy. The specific integration status of each strategy in the variants is illustrated in Table 7, where 1 represents that the strategy is embedded and 0 indicates that the strategy is not embedded. To comprehensively assess the effectiveness of the strategies, the original ALA, the CALA, and all six ALA-based variants are subjected to an ablation study on the CEC2022 benchmark test set. The comparative performance results are systematically presented in Table 8.

As shown in Table 8, the full CALA algorithm integrating all strategies achieves the best overall performance, ranking first on 8 out of 12 test functions in terms of mean fitness. It also demonstrates excellent stability, attaining the lowest standard deviation in 5 functions. Among the variants, CALA-6 and CALA-5 perform well in specific cases (e.g., F5 and F12), but their overall robustness is inferior to the CALA. All methods reach the optimum on F9, suggesting this function is less challenging. On unimodal and multimodal functions (F1–F5), the CALA and CALA-6 show strong local search capability. For hybrid and composition functions (F6–F12), the CALA consistently ranks at the top, confirming its ability to handle complex landscapes. In summary, the three proposed strategies work best when combined, with the crossover strategy contributing the most, followed by the historical best guided strategy and linear inertia weight strategy.

Table 8. Experimental results of CALA-derived variants on the 10 dimensional CEC2022 test suite.

Function	Metric	ALA	CALA-1	CALA-2	CALA-3	CALA-4	CALA-5	CALA-6	CALA
F1	Ave	300.0002	300.0002	**300**	**300**	**300**	**300**	**300**	**300**
	Std	0.0003	0.0006	6.27E-05	2.27E-13	1.64E-05	1.60E-13	1.76E-13	**7.57E-14**
	Rank	2	2	**1**	**1**	**1**	**1**	**1**	**1**
F2	Ave	406.0338	405.1570	405.5249	403.9063	404.5364	405.2994	402.9048	**402.1656**
	Std	2.4811	2.8711	2.3410	3.8545	2.9006	2.9243	3.0291	**1.5390**
	Rank	8	7	6	3	4	5	2	**1**
F3	Ave	600.0001	600.0001	600.0033	**600**	600.0003	**600**	**600**	**600**
	Std	5.45E-05	0.0001	0.0056	6.08E-07	0.0003	3.58E-06	1.07E-06	**1.58E-12**
	Rank	3	2	5	**1**	4	**1**	**1**	**1**
F4	Ave	817.5113	815.7211	814.2315	808.0622	814.6853	807.9620	808.9856	**807.2662**
	Std	6.5193	6.6622	4.3521	2.3189	5.4918	2.7385	**1.8850**	2.2876
	Rank	8	7	5	3	6	2	4	**1**
F5	Ave	900.0992	900.0812	900.1542	900.0544	900.1356	900.0090	**900**	**900**
	Std	0.1679	0.1679	0.2307	0.1433	0.2017	0.0283	2.36E-13	**2.04E-13**
	Rank	5	4	7	3	6	2	**1**	**1**
F6	Ave	1883.1189	1866.0621	1864.6110	1801.7911	1853.8736	1801.5829	1802.0434	**1800.9473**
	Std	49.2351	22.6126	21.7596	0.7089	23.5898	1.2308	**0.6810**	1.8998
	Rank	8	7	6	3	5	2	4	**1**
F7	Ave	2021.3281	2019.4736	2021.3220	**2002.2211**	2019.5061	2005.9432	2004.8092	2003.9174
	Std	0.6521	6.5146	6.3422	2.3505	6.6424	6.5314	6.6821	**0.4900**
	Rank	8	5	7	**1**	6	4	3	2
F8	Ave	2214.0796	2213.1307	2220.1153	2209.9012	2218.4534	2210.1118	2213.1982	**2207.8340**
	Std	8.4031	9.6552	**3.1581**	6.5500	8.4233	8.2078	8.6441	8.6442
	Rank	6	4	8	2	7	3	5	**1**
F9	Ave	**2529.2844**	**2529.2844**	**2529.2844**	**2529.2844**	**2529.2844**	**2529.2844**	**2529.2844**	**2529.2844**
	Std	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
F10	Ave	2511.6642	2500.3405	2511.4980	2510.9800	2500.3292	**2500.2893**	2500.3217	2500.3076
	Std	35.9168	0.0783	35.3039	33.8853	0.0555	**0.0439**	0.0528	0.0737
	Rank	8	5	7	6	4	**1**	3	2
F11	Ave	2720.0000	2775.0473	2655.0427	**2630**	2690.0000	**2630**	2660.0000	**2630**
	Std	94.8683	162.0105	130.0988	94.8683	144.9138	154.9193	126.4911	**94.8683**
	Rank	5	6	2	**1**	4	**1**	3	**1**
F12	Ave	2861.6717	2861.4622	2861.1295	2860.9030	2861.4774	**2860.7251**	2861.5046	2861.2292
	Std	1.2713	0.7984	1.1717	1.5097	**0.4069**	1.1273	1.1725	1.3761
	Rank	8	5	3	2	6	**1**	7	4

5.5. CEC2017 experimental results

In this subsection, to comprehensively evaluate the performance of the CALA, we conduct comparative experiments using the CEC2017 benchmark functions [42]. The CEC2017 test suite is categorized into four groups: unimodal functions, multimodal functions, hybrid functions, and composition functions. Unimodal functions contain only one global optimum and no local optima, making them suitable for assessing the exploitation capability of an algorithm. Multimodal functions involve numerous local optima and are primarily used to evaluate the algorithm's ability to locate the global optimum and escape from local optima. Hybrid and composition functions are designed to test an algorithm's capability in handling complex and continuous optimization problems. Table A1 lists the function names, dimension sizes, search ranges, and theoretical optimal values for these functions. The results are presented in Table 9, where the best result for each test function is highlighted in bold. The corresponding convergence curves are illustrated in Figure 5.

To comprehensively evaluate the performance of the proposed CALA, experiments were conducted on the 30-dimensional CEC2017 benchmark functions. This test suite includes unimodal, multimodal, composition, and hybrid functions, which together provide a comprehensive assessment of both the exploitation and exploration capabilities of optimization algorithms. As shown in Table 9, the CALA demonstrates superior performance across nearly all function categories. In the unimodal group (F1–F3), the CALA achieves competitive results, especially on F3, where it significantly outperforms other algorithms in both solution accuracy and stability. For multimodal functions (F4–F10), the CALA exhibits strong global search ability, achieving the best or near-best performance on several functions, such as F5, F8, and F9, with notably smaller standard deviations, reflecting its robustness in avoiding local optima. More notably, in the more challenging composition (F11–F20) and hybrid (F21–F30) functions, the CALA consistently achieves outstanding results. It ranks first in terms of average error on a large number of functions including F12, F13, F14, F18, F20, F21, and F30, and its standard deviations are often the lowest among all compared algorithms, indicating excellent convergence stability. Compared with the original ALA algorithm, the CALA significantly improves both convergence precision and consistency while preserving the core structure of the ALA. These results clearly verify that the three strategies introduced in the CALA are effective and contribute positively to performance enhancement. Overall, the CALA achieves top-tier optimization performance on CEC2017, demonstrating strong adaptability, robustness, and search efficiency across a wide range of optimization landscapes.

Table 9. Experimental results of 8 algorithms on unimodal and multimodal functions of CEC2017 (Dim = 30).

Function	Metric	PSO	SCA	COA	HHO	LCA	LSHADE	ALA	CALA
F1	Ave	1.2520E+09	1.7046E+10	4.6335E+10	2.3633E+07	6.4547E+10	**1.3991E+02**	2.1573E+04	4.3913E+03
	Std	1.5524E+09	2.6449E+09	7.0786E+09	7.2959E+06	6.8332E+09	**5.4908E+01**	1.7813E+04	4.8064E+03
F3	Ave	2.5623E+04	5.7857E+04	7.6220E+04	2.9816E+04	9.3703E+04	6.5961E+04	1.3603E+04	**3.2024E+02**
	Std	6.4965E+03	9.9089E+03	6.6726E+03	6.7767E+03	1.4020E+03	1.0924E+04	3.3423E+03	**2.1822E+01**
F4	Ave	5.8316E+02	2.1144E+03	1.1469E+04	5.4509E+02	1.6352E+04	4.9321E+02	5.0903E+02	**4.8258E+02**
	Std	1.2800E+02	5.7091E+02	2.4522E+03	3.1535E+01	3.5285E+03	1.4385E+03	5.9252E+01	**3.0665E+01**
F5	Ave	5.7657E+02	8.0063E+02	8.8209E+02	7.3095E+02	9.7023E+02	6.5686E+02	5.7952E+02	**5.7653E+02**
	Std	1.4189E+01	2.2518E+01	3.2867E+01	4.0343E+01	3.3422E+01	1.8554E+02	2.0923E+01	**1.6885E+01**
F6	Ave	6.0280E+02	6.5636E+02	6.7965E+02	6.6210E+02	6.9998E+02	**6.0000E+02**	6.0638E+02	6.0012E+02
	Std	1.9509E+00	4.7054E+00	6.8223E+00	5.9572E+00	7.1060E+00	**2.9068E-05**	3.5851E+00	1.3460E-01
F7	Ave	**8.1349E+02**	1.1828E+03	1.3492E+03	1.2502E+03	1.5132E+03	8.9249E+02	8.5284E+02	8.3333E+02
	Std	3.0788E+01	5.5182E+01	5.6880E+01	7.6740E+01	4.5125E+01	1.2663E+02	3.5127E+01	**2.1705E+01**
F8	Ave	8.8454E+02	1.0842E+03	1.1104E+03	9.6905E+02	1.1941E+03	9.5816E+02	8.8424E+02	**8.7769E+02**
	Std	3.1533E+01	2.3340E+01	2.4071E+01	2.2256E+01	2.6889E+01	1.0129E+02	2.0542E+01	**1.9277E+01**
F9	Ave	1.1153E+03	6.9818E+03	9.7479E+03	7.4561E+03	1.3623E+04	**9.0000E+02**	1.4675E+03	1.0469E+03
	Std	3.5016E+02	1.2452E+03	1.1935E+03	8.8905E+02	2.2906E+03	1.0888E+03	3.8099E+02	**1.8375E+02**
F10	Ave	**4.2143E+03**	8.5324E+03	8.4757E+03	5.7237E+03	9.9784E+03	7.8660E+03	5.0495E+03	4.7160E+03
	Std	6.1170E+02	2.9739E+02	3.7811E+02	5.4991E+02	4.2872E+02	**2.5306E+02**	6.7340E+02	6.2097E+02

*The experimental results of 8 algorithms on CEC2017 (Dim = 30) hybrid and composition functions (F11–F30) are presented in the appendix, Table A2.

Figure 5 illustrates the convergence behaviors of eight algorithms on nine representative CEC2017 benchmark functions. In each subplot, the best fitness value is plotted against the number of iterations, and the red curve denotes the performance of the proposed CALA. As shown in Figure 5, the CALA consistently demonstrates superior convergence speed and solution quality across all test functions. On functions such as F4, F8, F23, and F24, the CALA rapidly approaches a high-quality solution in the early iterations and continues to refine the result as the optimization progresses. Notably, in F13, F18, and F30—functions known for their complexity and rugged landscapes—the CALA not only converges faster than the other algorithms but also achieves the lowest final fitness value, indicating its strong ability to balance exploration and exploitation. Compared to the original ALA and other peer methods (e.g., PSO, SCA, HHO, LSHADE), the CALA achieves significantly better convergence curves, with steeper descent trends and smaller final fitness value. This improvement reflects the effectiveness of the enhancements introduced in the CALA, which accelerate convergence while maintaining robust search performance across diverse problem types. Overall, the convergence plots further validate the outstanding optimization capability and stability of the CALA in solving complex and high-dimensional problems.

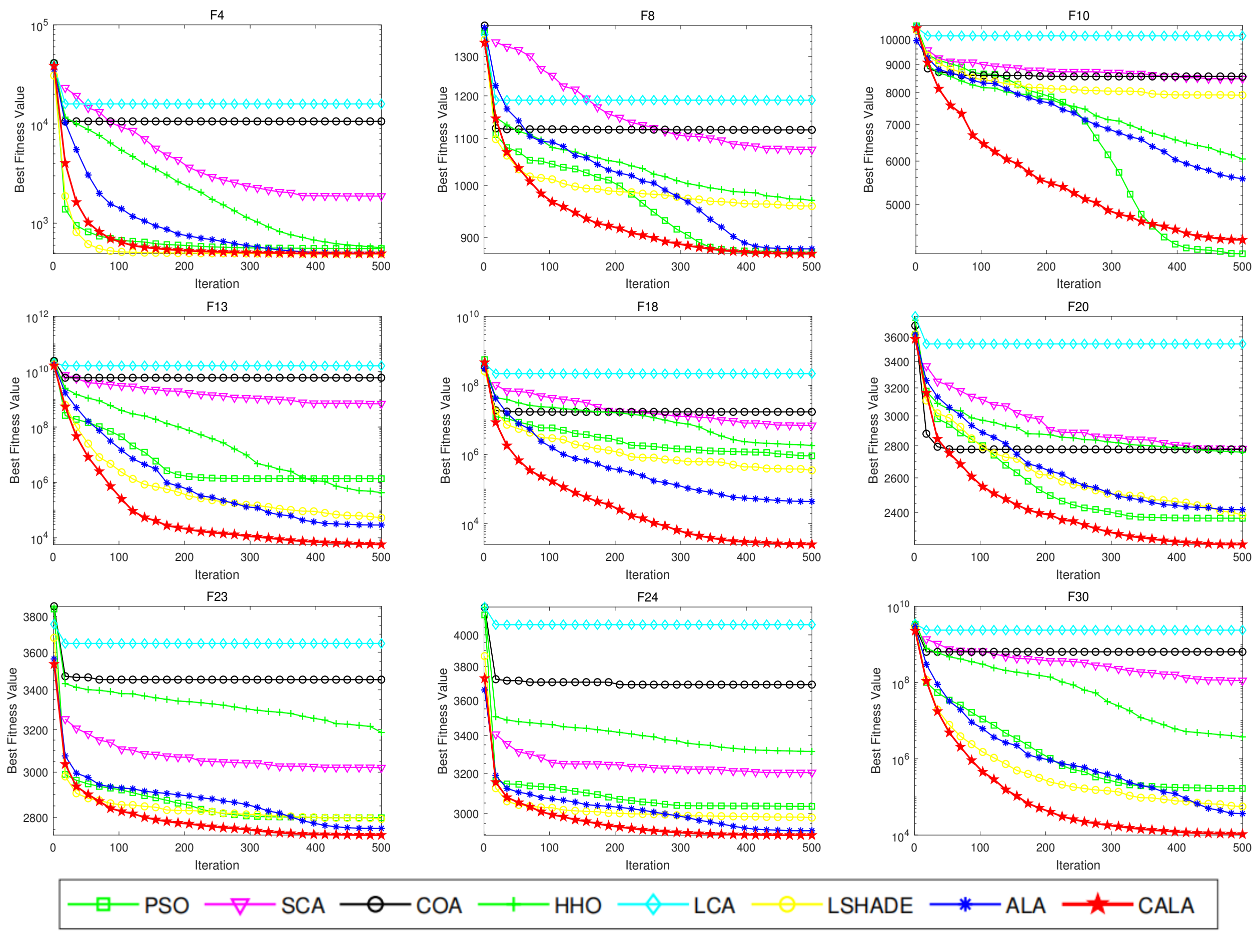

Figure 5. CEC2017 test function convergence curve (Dim = 30).

5.6. CEC2022 experimental results

To evaluate the effectiveness of the CALA algorithm, a comparative analysis was conducted between the CALA and seven well-established benchmark algorithms using the CEC2022 test suite. The CEC2022 benchmark consists of 12 functions, including a combination of unimodal, multimodal, hybrid, and composition functions [43]. Detailed descriptions of the benchmark functions are provided in the Appendix, Table A3. The performance results of the eight algorithms on the CEC2022 test suite are summarized in Table 10, and the corresponding convergence curves are illustrated in Figure 6.

Table 10. Experimental results of 8 algorithms on CEC2022 (Dim = 10).

Function	Metric	PSO	SCA	COA	HHO	LCA	LSHADE	ALA	CALA
F1	Ave	**3.0000E+02**	1.0477E+03	6.2700E+03	3.0230E+02	1.0363E+04	3.0004E+02	3.0000E+02	**3.0000E+02**
	Std	7.1143E-07	4.0021E+02	1.5178E+03	1.1538E+00	6.1938E+02	5.0230E-02	3.7153E-04	**4.1396E-13**
F2	Ave	4.1236E+02	4.5472E+02	8.5992E+02	4.1972E+02	2.0646E+03	4.0741E+02	4.0548E+02	**4.0313E+02**
	Std	1.1770E+01	1.6195E+01	2.3581E+02	2.8609E+01	9.1092E+02	**2.1735E+00**	2.9195E+00	3.4444E+00
F3	Ave	6.0006E+02	6.1753E+02	6.3621E+02	6.3104E+02	6.6594E+02	**6.0000E+02**	6.0000E+02	**6.0000E+02**
	Std	2.4541E-01	4.6642E+00	8.2210E+00	1.1722E+01	1.1601E+01	**1.6193E-09**	5.4929E-04	6.1568E-07
F4	Ave	8.1130E+02	8.3820E+02	8.4084E+02	8.2124E+02	8.7032E+02	8.1782E+02	8.1705E+02	**8.0816E+02**
	Std	4.3851E+00	6.7504E+00	8.7539E+00	7.6399E+00	1.0397E+01	3.4821E+00	5.3594E+00	**2.9232E+00**
F5	Ave	9.0011E+02	1.0162E+03	1.2622E+03	1.3763E+03	1.9011E+03	**9.0000E+02**	9.0018E+02	**9.0000E+02**
	Std	1.8616E-01	8.5990E+01	1.5392E+02	1.4578E+02	2.8452E+02	**0**	3.3500E-01	1.6346E-02
F6	Ave	4.4009E+03	2.0331E+06	1.7611E+06	3.0156E+03	1.0707E+08	1.8116E+03	1.8674E+03	**1.8019E+03**
	Std	2.3954E+03	2.1771E+06	2.8501E+06	1.5464E+03	6.9088E+07	4.7648E+00	2.4320E+01	**1.2468E+00**
F7	Ave	2.0151E+03	2.0537E+03	2.0718E+03	2.0553E+03	2.1400E+03	2.0074E+03	2.0197E+03	**2.0049E+03**
	Std	9.4071E+00	8.2937E+00	1.4561E+01	2.7418E+01	3.0041E+01	**2.2733E+00**	5.7822E+00	6.5287E+00
F8	Ave	2.2204E+03	2.2317E+03	2.2328E+03	2.2332E+03	2.3072E+03	2.2127E+03	2.2167E+03	**2.2096E+03**
	Std	2.2154E+01	**3.0286E+00**	4.8127E+00	1.2296E+01	8.3112E+01	4.1793E+00	7.8788E+00	8.4495E+00
F9	Ave	2.5345E+03	2.5661E+03	2.7033E+03	2.5537E+03	2.8460E+03	**2.5293E+03**	**2.5293E+03**	**2.5293E+03**
	Std	2.6784E+01	2.1719E+01	3.6941E+01	3.1170E+01	1.0185E+02	**0**	**0**	**0**
F10	Ave	2.5390E+03	2.5114E+03	2.5654E+03	2.5394E+03	2.7845E+03	2.5040E+03	2.5003E+03	**2.5003E+03**
	Std	5.5482E+01	3.6436E+01	7.2033E+01	6.5152E+01	4.9570E+02	2.0311E+01	7.9155E-02	**5.4838E-02**
F11	Ave	2.6901E+03	2.9403E+03	3.3810E+03	2.7774E+03	4.6452E+03	2.8200E+03	2.6784E+03	**2.6300E+03**
	Std	1.1558E+02	2.3181E+02	3.0099E+02	1.4550E+02	4.8732E+02	1.3493E+02	1.3047E+02	**9.4868E+01**
F12	Ave	2.8669E+03	2.8692E+03	2.9130E+03	2.8974E+03	3.0592E+03	2.8625E+03	**2.8611E+03**	2.8613E+03
	Std	4.8933E+00	1.3761E+00	3.3667E+01	3.9696E+01	9.1830E+01	1.5678E+00	1.3208E+00	**1.1334E+00**

As shown in Table 10, the proposed CALA algorithm demonstrates clear superiority over the original ALA and six other metaheuristic algorithms of the CEC2022 test suite. The CALA achieves the best mean fitness on 9 out of 12 functions (F1, F2, F4, F6–F8, F10–F11), and shares the best results on F1, F3, F5, and F9. Moreover, it consistently shows lower standard deviations, indicating high solution stability and robustness. In particular, the CALA exhibits outstanding precision on F1 and F3, with negligible variance, and maintains minimal fluctuations on complex functions such as F6, F10, and F11. Compared with the original ALA, the CALA exhibits improvements in both accuracy and stability on almost all test functions, confirming that the three introduced strategies—linear inertia weight strategy, historical best-guided strategy, and crossover strategy—effectively enhance the algorithm's global search ability, convergence speed, and solution precision. Overall, the experimental results validate that the integration of the three strategies significantly strengthens the performance of the ALA, making the CALA more competitive and reliable in solving complex optimization problems.

Figure 6 illustrates the convergence curves of the CALA algorithm on selected CEC2022 test functions (F1, F4, F6, F7, F8, and F9). For unimodal functions (e.g., F1), the CALA demonstrates a rapid decline in fitness values during the early stages, approaching the optimal value within approximately 100 iterations. Throughout the entire process, its fitness values remain consistently lower than those of the compared algorithms, indicating superior convergence accuracy. In the case of multimodal functions (e.g., F4 and F7), the CALA effectively escapes local optima, with steep drops in the curves suggesting a strong ability to locate the global optimum quickly and avoid local traps, thus exhibiting a clear advantage in convergence speed. Furthermore, for functions like F6 and F9, the convergence curves remain smooth with minimal fluctuations across iterations, particularly aligning closely with low-value baselines in the later stages, indicating that the CALA maintains stable performance and is less affected by local optima, reflecting the robustness of its search mechanism.

Moreover, the combined use of the linear inertia weight strategy, the historical best-guided strategy, and the crossover strategy significantly enhances the algorithm's global exploration and local exploitation capabilities, thereby improving its overall convergence speed, solution accuracy, and stability. In summary, the convergence curves demonstrate that the CALA exhibits fast convergence, high precision, and strong stability across test functions, highlighting its potential and adaptability for solving complex problems in high-dimensional scenarios.

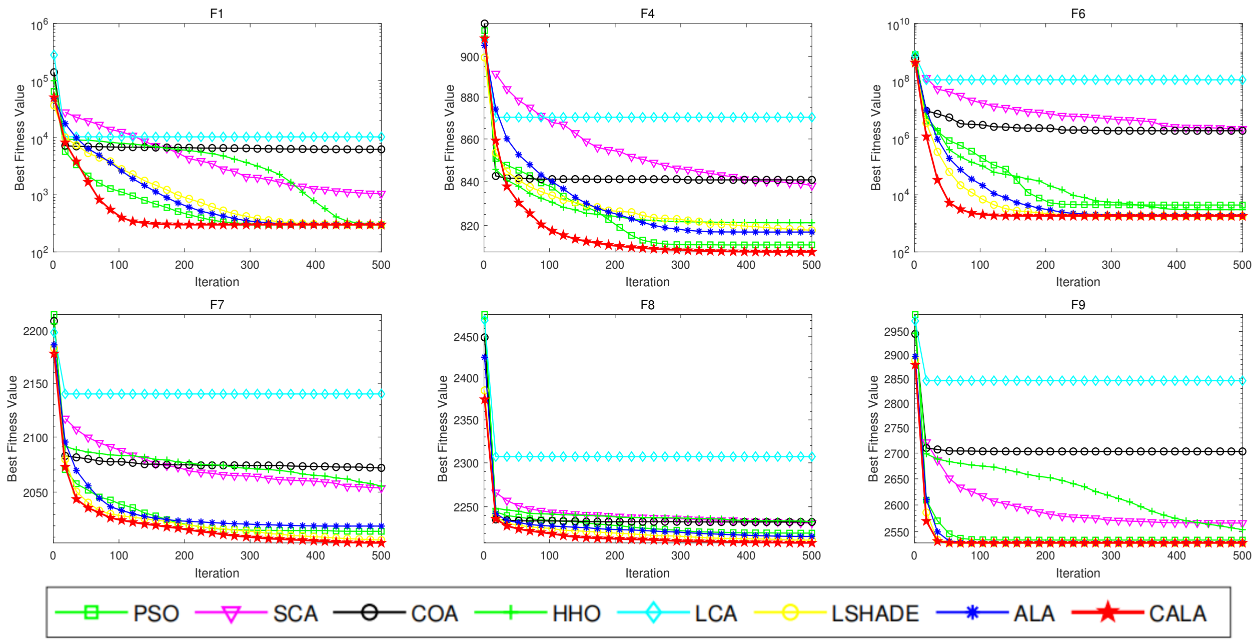

Figure 6. CEC2022 test function convergence curve (Dim = 10).

5.7. *Statistical test*

In this subsection, we carried out the Wilcoxon rank sum test and the Friedman ranking test on the CALA and other comparative algorithms. Subsequently, we evaluated the experimental results to conduct a quantitative analysis of the differences between the CALA and other comparison algorithms.

5.7.1. Wilcoxon rank sum test

To comprehensively demonstrate the advantages of the improved CALA, the Wilcoxon rank sum test is employed in this section to evaluate the statistical differences between the CALA and other algorithms at a significance level of 5% [44]. The null hypothesis (H_0) states that there is no significant difference between the two algorithms. If the p-value is less than 0.05, the null hypothesis is rejected, indicating a statistically significant difference; if the p-value is greater than 0.05, the null hypothesis is not rejected, suggesting no significant difference and implying that the algorithms exhibit similar performance. A "NaN" value indicates that the comparison could not be performed due to nearly identical performance. The Wilcoxon test results comparing the CALA with other algorithms on the CEC2017 benchmark functions are presented in Table 11, while those for the CEC2022 benchmark functions are shown in Table 12. To highlight the comparison outcomes, values with a significance level greater than 5% are displayed in bold.

Table 11. Wilcoxon rank sum test p-values between the CALA and comparison algorithms on CEC2017 benchmark functions. Bold values indicate a p-value > 0.05, suggesting no statistically significant difference at the 5% significance level.

Function	PSO	SCA	COA	HHO	LCA	LSHADE	ALA
F1	3.83E-06	3.02E-11	3.02E-11	3.02E-11	3.02E-11	2.23E-09	7.38E-10
F3	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11
F4	2.38E-07	3.02E-11	3.02E-11	2.23E-09	3.02E-11	**7.39E-01**	**1.86E-01**
F5	**6.20E-01**	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	**1.12E-01**
F6	1.60E-07	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11
F7	**5.19E-02**	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.34E-11	7.30E-04
F8	**2.46E-01**	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	2.24E-02
F9	**3.63E-01**	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.01E-11	3.83E-06
F10	**3.48E-01**	3.02E-11	3.02E-11	3.82E-09	3.02E-11	3.02E-11	1.56E-08
F11	3.65E-08	3.02E-11	3.02E-11	1.61E-10	3.02E-11	2.60E-05	9.06E-08
F12	3.69E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	2.61E-10	2.15E-10
F13	5.00E-09	3.02E-11	3.02E-11	3.02E-11	3.02E-11	1.07E-07	5.09E-06
F14	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11
F15	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11
F16	2.68E-04	3.02E-11	3.02E-11	1.78E-10	3.02E-11	3.47E-10	**5.75E-02**
F17	1.11E-03	3.02E-11	3.02E-11	3.02E-11	3.02E-11	2.78E-07	1.34E-05
F18	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11
F19	1.61E-10	3.02E-11	3.02E-11	3.02E-11	3.02E-11	2.87E-10	4.08E-11
F20	2.81E-02	3.02E-11	3.34E-11	3.34E-11	3.02E-11	3.83E-06	1.11E-04
F21	**2.34E-01**	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	1.06E-03
F22	1.25E-04	1.21E-10	3.02E-11	2.37E-10	3.02E-11	2.61E-10	6.28E-06
F23	7.39E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.34E-11	7.70E-04
F24	4.08E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.51E-02
F25	9.06E-08	3.02E-11	3.02E-11	3.02E-11	3.02E-11	4.06E-02	6.77E-05
F26	3.50E-03	3.02E-11	3.02E-11	4.62E-10	3.02E-11	3.34E-11	6.28E-06
F27	4.69E-08	3.02E-11	3.02E-11	3.02E-11	3.02E-11	2.92E-02	**8.24E-02**
F28	8.10E-10	3.02E-11	3.02E-11	4.08E-11	3.02E-11	5.75E-02	1.25E-07
F29	1.38E-02	3.02E-11	3.02E-11	3.02E-11	3.02E-11	1.10E-08	5.27E-05
F30	5.07E-10	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	1.69E-09

As shown in Table 11, for the majority of benchmark functions, the p-values obtained from the Wilcoxon rank sum test between the CALA and other optimization algorithms are less than 0.05, indicating statistically significant differences at the 5% significance level. However, when compared with PSO, the p-values for six benchmark functions exceed 0.05. This is attributed to the incorporation of the inertia weight parameter w from PSO into the CALA, suggesting that the two algorithms do not exhibit significant differences on these functions. Similarly, in comparison with the ALA algorithm, there are four functions (F4, F5, F16, and F27) for which the p-values are greater than 0.05, indicating no statistically significant differences between the CALA and ALA on these functions at the 5% significance level. This can be explained by the fact that the CALA is developed based on the ALA framework — while retaining the core advantages of the original algorithm, the CALA introduces three enhancement strategies, resulting in similar performance on certain test functions.

Table 12. *P*-value on CEC2022.

Function	PSO	SCA	COA	HHO	LCA	LSHADE	ALA
F1	2.38E-11	2.38E-11	2.38E-11	2.38E-11	2.38E-11	2.38E-11	2.38E-11
F2	2.57E-03	2.63E-11	2.63E-11	2.62E-05	2.63E-11	9.53E-04	4.83E-03
F3	6.87E-05	2.90E-11	2.90E-11	2.90E-11	2.90E-11	3.94E-04	3.21E-11
F4	3.40E-01	3.02E-11	3.02E-11	8.15E-11	3.02E-11	1.46E-10	1.03E-06
F5	3.05E-05	1.08E-11	1.08E-11	1.08E-11	1.08E-11	2.06E-10	1.08E-11
F6	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.69E-11	3.02E-11
F7	5.57E-03	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.18E-03	4.50E-11
F8	1.64E-05	3.34E-11	3.02E-11	3.02E-11	3.02E-11	1.84E-02	5.26E-04
F9	**8.15E-02**	1.21E-12	1.21E-12	1.21E-12	1.21E-12	NAN	NAN
F10	2.20E-07	5.07E-10	2.61E-10	1.96E-10	4.50E-11	1.06E-03	3.83E-05
F11	2.78E-05	1.05E-04	2.20E-11	1.67E-07	2.20E-11	6.05E-04	6.68E-06
F12	2.92E-08	2.25E-11	2.25E-11	2.25E-11	2.25E-11	**8.85E-02**	**1.33E-01**

Table 12 presents the Wilcoxon rank sum test results between the CALA algorithm and the compared algorithms on the CEC2022 benchmark functions. As shown in Table 12, the CALA exhibits statistically significant differences from most of the compared algorithms across various test functions, with only a few functions failing to pass the significance test. This indicates clear performance distinctions between the CALA and the compared algorithms. In summary, the incorporation of three strategies–linear inertia weight strategy, historical best-guided strategy, and crossover strategy–into the ALA framework has proven to be a successful enhancement of the CALA algorithm.

5.7.2. Friedman ranking test

To evaluate the performance of the CALA algorithm, we employed the non-parametric Friedman mean rank test to rank the experimental results of the CALA against other algorithms on the CEC2017 and CEC2022 benchmark suites. This method offers the advantage of not relying on assumptions about the underlying data distribution, making it particularly well-suited for assessing the performance of optimization algorithms across diverse benchmark sets.

As shown in Table 13, we present the Friedman test ranking results for all compared algorithms. The column "Ave.rank" represents the average rank obtained through the Friedman test by calculating the optimizer's ranking across all test functions, where a lower value indicates better overall performance. The "Ave.rank" values of the CALA on the CEC2017 and CEC2022 benchmark functions are 1.37 and 1.5, respectively. Overall, the CALA consistently ranks first among all compared algorithms. This result not only confirms its superior performance on the selected benchmark suites but also highlights its efficiency and reliability in solving complex optimization problems. Furthermore, the CALA's significant ranking advantage underscores its adaptability and robustness across test functions with varying dimensions and characteristics, enabling it to maintain stable optimization performance across a wide range of application scenarios.

Table 13. Friedman ranking test results of CEC2017 and CEC2022 test sets.

Test sets	CEC2017		CEC2022	
Algorithms	Ave.rank	Overall rank	Ave.rank	Overall rank
PSO	3.33	4	3.50	4
SCA	5.83	6	5.50	5
COA	6.93	7	6.75	6
HHO	5.00	5	5.50	5
LCA	8.00	8	8.00	7
LSHADE	2.63	2	2.67	3
ALA	2.90	3	2.58	2
CALA	**1.37**	**1**	**1.5**	**1**

6. The CALA is used for engineering optimization problems

As demonstrated in the experimental study and analysis in the previous section, the proposed CALA algorithm exhibits outstanding performance on benchmark optimization functions. However, the ultimate measure of a metaheuristic algorithm's value lies in its ability to address the complexities of real-world problems. In light of this, the present section aims to validate the practical effectiveness and general applicability of the CALA through a series of application cases that simulate real-world scenarios. To comprehensively assess the algorithm's practical potential and scalability, the CALA is applied to four representative engineering optimization problems, thereby illustrating its capability to efficiently handle complex real-world challenges.

6.1. Pressure vessel design problem

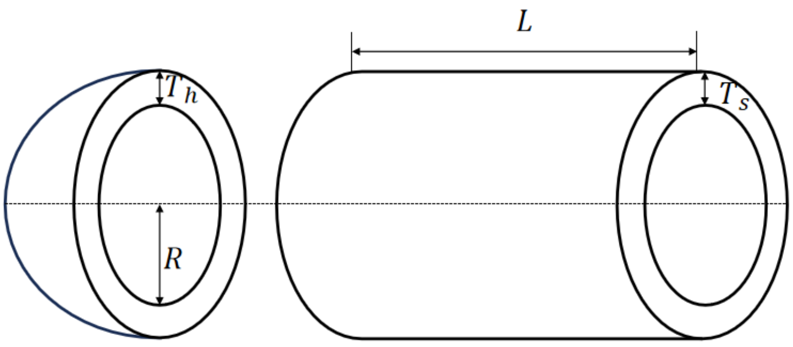

Figure 7. Pressure vessel design structure.

As shown in Figure 7, the design of pressure vessels represents a typical constrained optimization problem [45, 46], involving four decision variables: shell thickness (T_s), head thickness (T_h), internal radius (R), and the length of the cylindrical section excluding the head (L). The primary objective is to minimize the overall expense of molding, material, and welding for the pressure vessel, subject to a set of structural and functional constraints. The corresponding mathematical model is formulated in Eq (6.1).

$$\begin{aligned} &Consider\ \vec{x} = [x_1\ x_2\ x_3\ x_4] = [T_S\ T_h\ R\ L],\\ &Minimize\ f(\vec{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3,\\ &Subject\ to\ g_1(\vec{x}) = -x_1 + 0.0193x_3 \le 0,\\ &g_2(\vec{x}) = -x_2 + 0.00954x_3 \le 0,\\ &g_3(\vec{x}) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \le 0,\\ &g_4(\vec{x}) = x_4 - 240 \le 0,\\ &Parameter\ range\ 0 \le x_1, x_2 \le 99, 10 \le x_3, x_4 \le 200. \end{aligned} \tag{6.1}$$

According to the results presented in Table 14, the CALA outperforms its competitors in optimizing the pressure vessel design problem. Specifically, the algorithm achieves a parameter combination of T_s = 0.778169, T_h = 0.384649, R = 40.319619, and L = 200, yielding a total cost of 5885.33277617, while the theoretical optimum stands at 5.8853E+03. This outcome not only demonstrates the CALA's superior optimization capability but also underscores its efficacy in addressing real-world engineering challenges. The optimized parameters directly enhance the functional performance and cost-effectiveness of the pressure vessel.

Table 14. Experimental results of pressure vessel design (PVD).

Algorithm	Optimal values for variable				Optimal value	Ranking
	T_s	T_h	R	L		
PSO	1.060529	0.524220	54.949670	63.357130	6159.51238587	4
SCA	0.844987	0.500274	41.581061	193.313337	6907.61027696	6
COA	8.156754	2.417835	54.142188	68.663205	51670.25975309	7
HHO	1.174085	0.578651	60.619440	31.435695	6860.30600024	5
LCA	4.692799	56.739591	63.541105	32.401238	330484.4712261	8
LSHADE	0.778169	0.384649	40.319619	200	5885.33675995	2
ALA	0.778197	0.384663	40.321096	199.979440	5886.60005388	3
CALA	**0.778169**	**0.384649**	**40.319619**	**199.999999**	**5885.33277617**	**1**

6.2. Three-bar truss design problem

The three-bar truss design problem [47] involves two design variables: the cross-sectional area of bars 1 and 3 (x_1) and the cross-sectional area of bar 2 (x_2), as illustrated in Figure 8. The primary objective of this optimization task is to minimize the total weight of the truss structure. In addition, the design process is subject to manufacturing constraints, including stress, deflection, and buckling limits. The optimization model for this design problem is shown in Eq (6.2).

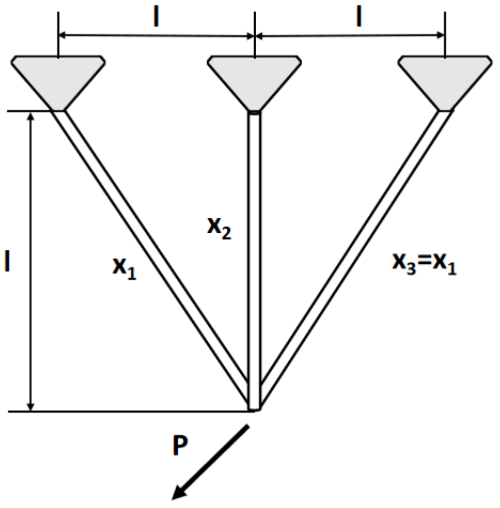

Figure 8. Schematic of the three-bar truss design problem.

$$
\begin{aligned}
&Consider\ \vec{x} = [x_1\ x_2],\\
&Minimize\ f(\vec{x}) = (2\sqrt{2}x_1 + x_2)\cdot l,\\
&Subject\ to\ R_1(\vec{x}) = \frac{(\sqrt{2}x_1 + x_2)}{(\sqrt{2}x_2(\frac{1+2x_1x_2}{P}))} - \sigma \le 0,\\
&R_2(\vec{x}) = \frac{x_2}{(\sqrt{2}x_1(\frac{1+2x_1x_2}{P}))} - \sigma \le 0,\\
&R_3(\vec{x}) = (\frac{x_2}{\sqrt{2}x_2 + x_1})(\frac{1}{P}) - \sigma \le 0,\\
&Parameter\ range\ 0 \le x_1, x_2 \le 1, l = 100\ cm, P = 2\ KN, \sigma = 2\ KN/cm^2.
\end{aligned}
\tag{6.2}
$$

Table 15. Experimental results of the three-bar truss design problem.

Algorithm	Optimal values for variable		Optimal value	Ranking
	x_1	x_2		
PSO	0.78927136	0.40656451	263.89592108	3
SCA	0.79338445	0.39533486	265.87710151	6
COA	0.79219625	0.39844231	263.93186039	4
HHO	0.79508214	0.39042165	263.97316893	5
LCA	0.67504020	0.91331727	268.33684991	7
LSHADE	**0.78867514**	**0.40824829**	**263.89584338**	**1**
ALA	0.78867820	0.40823963	263.89584340	2
CALA	**0.78867513**	**0.40824829**	**263.89584338**	**1**

Table 15 presents the optimization results of the CALA and seven competing algorithms on the three-bar truss design problem. According to the data, the CALA and LSHADE achieved the best performance with an optimal value of 263.89584338, which is very close to the theoretical optimum of 2.6389E+02. This result demonstrates the remarkable optimization capability of the CALA. Its outstanding performance can be attributed to the integration of innovative strategies, which significantly enhance the algorithm's convergence ability, allowing it to converge faster than other competing methods. The CALA not only excels in solving the three-bar truss design problem, but also exhibits strong potential for application in a wide range of complex engineering optimization tasks, particularly in identifying cost-effective design solutions.

6.3. Tension/compression spring design

This design problem [48–50] aims to minimize the weight of tension/compression springs by optimizing three critical parameters: wire diameter (*d*), coil diameter (*D*), and the number of coils (*N*). The structure of this engineering problem is illustrated in Figure 9, with the mathematical model presented in Eq (6.3).

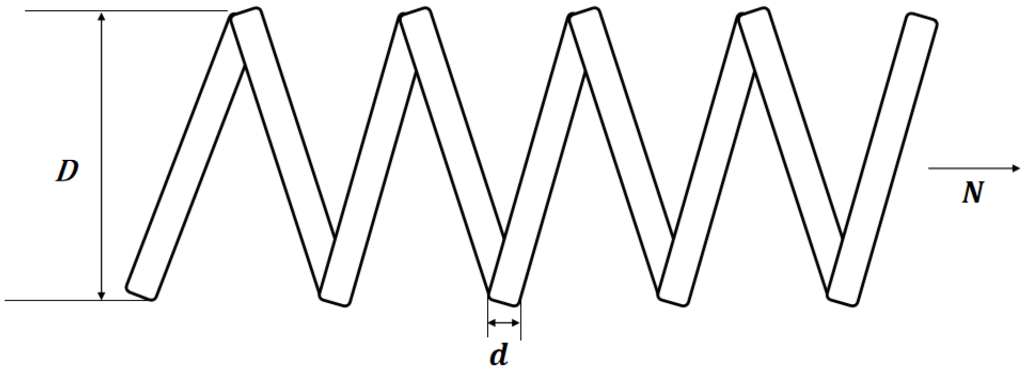

Figure 9. Tension/compression spring design structure.

$$
\begin{aligned}
&\mathit{Consider}\ \vec{x} = [x_1\ x_2\ x_3] = [d\ D\ N],\\
&\mathit{Minimize}\ f(\vec{x}) = (x_3 + 2)x_2x_1^2,\\
&\mathit{Subject\ to}\ g_1(\vec{x}) = 1 - \frac{x_2^3x_3}{71785x_1^4} \le 0,\\
&g_2(\vec{x}) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \le 0,\\
&g_3(\vec{x}) = 1 - \frac{140.45x_1}{x_2^2x_3} \le 0,\\
&g_4(\vec{x}) = \frac{x_1 + x_2}{1.5} - 1 \le 0,\\
&\mathit{Parameter\ range}\ 0.05 \le x_1 \le 2, 0.25 \le x_2 \le 1.3, 2 \le x_3 \le 15.
\end{aligned}
\tag{6.3}
$$

Table 16 compares the optimization results of the CALA with those of seven competing algorithms in the tension/compression spring design problem. According to the data, the CALA optimizes the wire diameter (d), coil diameter (D), and number of coils (N) to values of 0.051678169, 0.356455629, and 11.304353542, respectively, achieving a minimum cost of 0.0126656943, which is remarkably close to the theoretical optimum of 1.2665E-02. These results highlight the superior optimization capability of the CALA.

Table 16. Experimental results of tension/compression spring design.

Algorithm	Optimal values for variable			Optimal value	Ranking
	d	D	N		
PSO	0.054902802	0.439073022	7.705511883	0.0128624338	4
SCA	0.050000000	0.316197655	14.453633886	0.0129649952	5
COA	0.050000000	0.314498279	14.427049450	0.0129868856	6
HHO	0.060633998	0.612540093	4.221773976	0.0134032871	7
LCA	0.065570551	0.705978691	5.725365230	0.0234492137	8
LSHADE	0.051696273	0.356891268	11.278799629	0.0126657926	2
ALA	0.051480654	0.351724679	11.587810847	0.0126668832	3
CALA	**0.051678169**	**0.356455629**	**11.304353542**	**0.0126656943**	**1**

6.4. Welded beam design problem

The welded beam design problem [51], as illustrated in Figure 10, aims to determine the optimal dimensions of a welded beam that minimizes weight and reduces cost while supporting a given load. The design accounts for constraints including shear stress (τ), bending stress, buckling (σ), load on the bar (P_c), and end deflection (δ). The design process involves four variables: height (h), length (l), thickness (t), and breadth (b). The mathematical model for this problem is given by Eq (6.4).

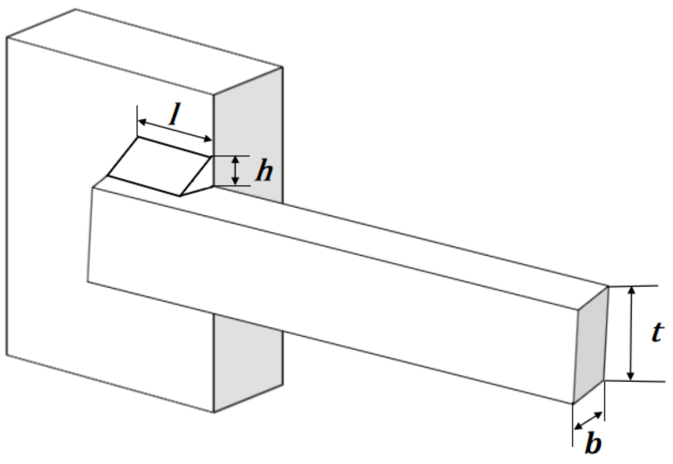

Figure 10. Welded beam design structure.

$$
\begin{aligned}
&Consider\ \vec{x} = [x_1\ x_2\ x_3\ x_4] = [h\ l\ t\ b],\\
&Minimize\ f(\vec{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 - x_2),\\
&Subject\ to\ g_1(\vec{x}) = \tau(x) - 13000 \leq 0,\\
&\qquad g_2(\vec{x}) = \sigma(x) - 30000 \leq 0,\\
&\qquad g_3(\vec{x}) = x_1 - x_4 \leq 0,\\
&\qquad g_4(\vec{x}) = 0.1047x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \leq 0,\\
&\qquad g_5(\vec{x}) = 0.125 - x_1 \leq 0,\\
&\qquad g_6(\vec{x}) = \delta(x) - 0.25 \leq 0,\\
&\qquad g_7(\vec{x}) = 6000 - P_c(x) \leq 0,\\
&where\ \tau(x) = \sqrt{(\tau')_2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2},\ \tau' = \frac{6000}{\sqrt{2}x_1x_2},\ \tau'' = \frac{MR}{J},\\
&\qquad M = 6000(14 + \frac{x_2}{2}),\ R = \sqrt{\frac{x_2^2}{4} + (\frac{x_1 + x_3}{2})^2},\\
&\qquad J = 2(\sqrt{2}x_1x_2[\frac{x_2^2}{12} + (\frac{(x_1 + x_3)}{2})^2]),\ \sigma(x) = \frac{504000}{x_4x_3^2},\ \delta(x) = \frac{2.1952}{x_3^3x_4},\\
&Parameter\ range\ 0.1 \leq x_1 \leq 2, 0.1 \leq x_2 \leq 10, 0.1 \leq x_3 \leq 10, 0.1 \leq x_4 \leq 2.
\end{aligned}
\tag{6.4}
$$

Table 17. Experimental results of tension/compression spring design.

Algorithm	Optimal values for variable				Optimal value	Ranking
	h	l	t	b		
PSO	0.205729640	3.234919327	9.036623903	0.205729640	1.6927687582	4
SCA	0.209823756	2.861645040	9.839819507	0.213334443	1.8295623593	5
COA	0.198649392	9.154041972	9.257666910	0.204681270	2.3567429784	7
HHO	0.198850641	3.364110981	9.076641260	0.206831652	1.8442131515	6
LCA	0.309381210	4.641518537	7.298865294	0.539776642	4.2168921301	8
LSHADE	0.205729640	3.234919306	9.036623910	0.205729640	1.6927682637	2
ALA	0.205729580	3.234920524	9.036623925	0.205729640	1.6927683475	3
CALA	**0.205729640**	**3.234919306**	**9.036623910**	**0.205729640**	**1.6927682636**	**1**

Table 17 presents a comparison between the CALA and several other optimization algorithms in terms of the minimum cost, decision variables, and ranking achieved in solving this design problem. As observed, the CALA consistently ranks first, attaining a minimum cost of 1.6927682636, which is remarkably close to the theoretical optimum of 1.6702E+00. This outstanding performance can be attributed to the algorithm's innovative optimization strategies, whose integration enables a more efficient convergence toward the global optimum. The superior result obtained by the CALA not only demonstrates its exceptional capability in addressing the welded beam design problem, but also highlights its potential applicability to a broad range of complex engineering optimization tasks.

7. Parameter identification problem for photovoltaic models

Solar energy, recognized for its abundant reserves and environmentally friendly nature, is widely regarded as a key renewable alternative to conventional fossil fuels. Among solar utilization technologies, photovoltaic (PV) systems represent the mainstream approach for converting solar radiation into electrical energy. The performance of PV systems is highly influenced by device selection and uncertainties in model parameters. Therefore, accurately estimating these parameters based on measured current-voltage characteristic data has become an essential optimization task for enhancing the design and operational efficiency of PV systems.

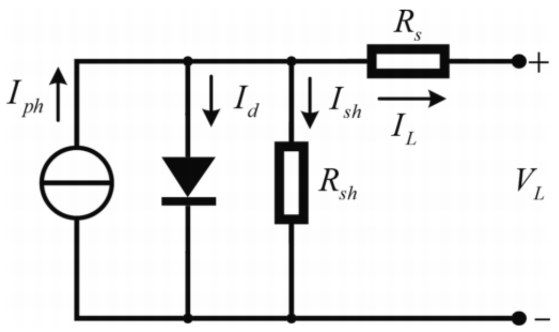

Figure 11. Equivalent circuit of an SDM photovoltaic cell.

The equivalent circuit of the single diode model (SDM) is illustrated in Figure 11. This model comprises a photo-generated current source in parallel with a diode, a series resistor to denote the ohmic losses related to load current, and a shunt resistor to present the leakage current. Thus, in terms of Kirchhoff's current law (KCL), the PV cell terminal current, I_t , can be expressed by:

$$I_t = I_{ph} - I_d - I_{sh}, \tag{7.1}$$

where I_{ph} denotes the photo-generated current, I_d denotes the diode current, and I_{sh} denotes the shunt resistor current, respectively. Additionally, in term of the Shockley equation, I_d is computed by:

$$I_d = I_{sd}[exp(q(V_t + I_tR_s)/nkT) - 1], \tag{7.2}$$

where I_{sd} is the reverse saturation current of the diode, V_t is the cell terminal voltage, R_s is the series resistance, n is the diode ideality factor, k is the Boltzmann constant ($1.38010^{-23}J/K$), q is the electronic charge ($1.60210^{-19}C$), and T is the PV cell absolute temperature in Kelvin, respectively. Moreover, using Kirchhoff's voltage law (KVL), I_{sh} is obtained as:

$$I_{sh} = (V_t + I_tR_s)/R_{sh}, \tag{7.3}$$

where R_{sh} is the shunt resistance. Therefore, the I–V relationship of the SDM can be rewritten as follows:

$$I_t = I_{ph} - I_{sd}[exp(q(V_t + I_tR_s)/nkT) - 1] - (V_t + I_tR_s)/R_{sh}. \tag{7.4}$$

The principal aim of this problem is minimizing the disparity between the experimental data estimated by an algorithm and the real measured data as much as possible, thus the root mean square error (RMSE) metric is adopted as the objective function as follows:

$$minRMSE(\vec{x}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} f_i(\vec{x}, I_t, V_t)^2}, \tag{7.5}$$

where N signifies the quantity of experimental data and $\vec{x}$ denotes the solution vector of the five parameters defined in Eq (7.6).

$$\begin{cases} f(\vec{x}, I_t, V_t) = I_{ph} - I_{sd}[exp(q(V_t + I_tR_s)/nkT) - 1] - (V_t + I_tR_s)/R_{sh} - I_t, \\ \vec{x} = [I_{ph}, I_{sd}, R_s, R_{sh}, n], \\ 0 \le I_{ph} \le 1, 0 \le I_{sd} \le 0.5, 0 \le R_s \le 0.5, 0 \le R_{sh} \le 100, 1 \le n \le 2. \end{cases} \tag{7.6}$$

The benchmark current-voltage data are obtained from [52], in which a commercial R.T.C French silicon solar cell with a 57-mm diameter was tested under 1000 W/m^2 at 33 °C.

Table 18. Optimal results of the CALA and competitive algorithms for the parameter identification problem of the SDM.

Algorithm	Optimal values for variable					RMSE	Ranking
	$I_{ph}(A)$	$I_{sd}(\mu A)$	$R_s(\Omega)$	$R_{sh}(\Omega)$	n		
PSO	0.836820	0.000000	0.000000	1.148912	1.433546	2.2286E-01	8
COA	0.832732	0.000000	0.000000	1.163695	1.000000	2.2288E-01	9
HHO	0.786891	4.607745	0.014567	7.630152	1.820015	2.1775E-02	7
LCA	0.537521	0.000000	0.075937	1.465383	1.000000	3.5890E-01	10
LSHADE	0.760776	0.323021	0.036377	53.718523	1.481184	**9.8602E-04**	**1**
ALA	0.760773	0.429205	0.035194	60.579945	1.510393	1.1305E-03	6
CALA	0.760775	0.322945	0.036378	53.713606	1.481160	**9.8602E-04**	**1**
ESCA [53]	0.760775	0.323020	0.036377	53.71852	1.4811	9.86021E-04	2
GOTLBO [54]	0.760780	0.331552	0.036265	54.115426	1.483820	9.87442E-04	5
CARO [55]	0.76079	0.31724	0.03644	53.0893	1.48168	9.8665E-04	4
SJAYA [56]	0.760776	0.323021	0.036377	53.7185	1.48118	9.86022E-04	3
R_{cr}-**IJADE** [57]	0.760776	0.323021	0.036377	53.718526	1.481184	**9.8602E-04**	**1**

Table 18 summarizes the optimal parameters (including I_{ph}, I_{sd}, R_s, R_{sh}, n) obtained by the CALA algorithm for the SDM of solar cells and their corresponding root mean square error (RMSE) objective values, along with comparative results against the traditional ALA and several other parameter estimation methods. The first six algorithms were previously used for comparison with the CALA, while the latter five algorithms (ESCA, GOTLBO, CARO, SJAYA, and R_{cr}-IJADE) were selected as benchmarks due to their excellent performance in recent literature on SDM parameter estimation for photovoltaic cells. As shown by the RMSE values in Table 18, the CALA, LSHADE, and R_{cr}-IJADE achieve the best RMSE value of 9.8602E-04. This indicates that the CALA significantly improves the performance of the conventional ALA. Therefore, the optimal parameter values obtained by the

CALA are closer to the true parameters of the SDM, demonstrating the higher accuracy of the CALA in parameter estimation.

To further investigate the quality of the parameters estimated by the proposed CALA, the estimated values of I_{ph}, I_{sd}, R_s, R_{sh}, and n are substituted into the SDM in Eq (7.4) to reconstruct the calculated current data and calculated power data at the experimental voltage point. The experimental data (voltage, current, and power), the calculated data, and the individual absolute errors (I_{IAE} and P_{IAE}) between experimental and calculated data are listed in Table 19. As shown in columns 6 and 8 and the last row of Table 19, the IAEs and their sum are so small, providing concrete evidence that the parameter values estimated by the CALA are highly accurate.

Table 19. The calculated results of the proposed CALA for the SDM solar cell.

Item	Experimental data			Calculated current data		Calculated power data	
	V(V)	I(A)	P(W)	I_{cal}(A)	I_{IAE}	P_{cal}(W)	P_{IAE}(W)
1	-0.2057	0.764	-0.157155	0.764088	8.79088E-05	-0.157173	1.80828E-05
2	-0.1291	0.762	-0.098374	0.762663	0.000663161	-0.098460	8.56141E-05
3	-0.0588	0.7605	-0.044717	0.761355	0.000855262	-0.044768	5.02894E-05
4	0.0057	0.7605	0.004335	0.760154	0.000346164	0.004333	1.97313E-06
5	0.0646	0.76	0.049096	0.759055	0.000945046	0.049035	6.105E-05
6	0.1185	0.759	0.089942	0.758042	0.000958	0.089828	0.000113523
7	0.1678	0.757	0.127025	0.757091	9.12293E-05	0.127040	1.53083E-05
8	0.2132	0.757	0.161392	0.756141	0.000859126	0.161209	0.000183166
9	0.2545	0.7555	0.192275	0.755086	0.000413662	0.192169	0.000105277
10	0.2924	0.754	0.220470	0.753663	0.000336666	0.220371	9.84412E-05
11	0.3269	0.7505	0.245338	0.751390	0.000890464	0.245630	0.000291093
12	0.3585	0.7465	0.267620	0.747353	0.000853463	0.267926	0.000305967
13	0.3873	0.7385	0.286021	0.740117	0.001617031	0.286647	0.000626276
14	0.4137	0.728	0.301174	0.727382	0.000617691	0.300918	0.000255539
15	0.4373	0.7065	0.308952	0.706973	0.00047304	0.309159	0.00020686
16	0.459	0.6755	0.310055	0.675281	0.000219196	0.309954	0.000100611
17	0.4784	0.632	0.302349	0.630759	0.001240944	0.301755	0.000593667
18	0.496	0.573	0.284208	0.571929	0.001070912	0.283677	0.000531172
19	0.5119	0.499	0.255438	0.499608	0.000607515	0.255749	0.000310987
20	0.5265	0.413	0.217445	0.413649	0.000648913	0.217786	0.000341653
21	0.5398	0.3165	0.170847	0.317510	0.001009835	0.171392	0.000545109
22	0.5521	0.212	0.117045	0.212154	0.00015435	0.117130	8.52164E-05
23	0.5633	0.1035	0.058302	0.102251	0.001249388	0.057598	0.00070378
24	0.5736	-0.01	-0.005736	-0.008718	0.001282013	-0.005001	0.000735363
25	0.5833	-0.123	-0.071746	-0.125507	0.00250734	-0.073208	0.001462531
26	0.59	-0.21	-0.123900	-0.208471	0.001528514	-0.122998	0.000901823
Sum of IAE					0.021526833		0.008730372

Figure 12 illustrates the difference between measured data as well as experimental data achieved by the CALA on the SDM. It is obvious that the experimental data derived from the CALA perfectly fit the measured data. These results demonstrate that the proposed CALA can also present great advantages in solving the more practical problem of parameter identification for PV models, which is attributed to its co-enhanced exploration and exploitation capabilities by the multi-strategy method.

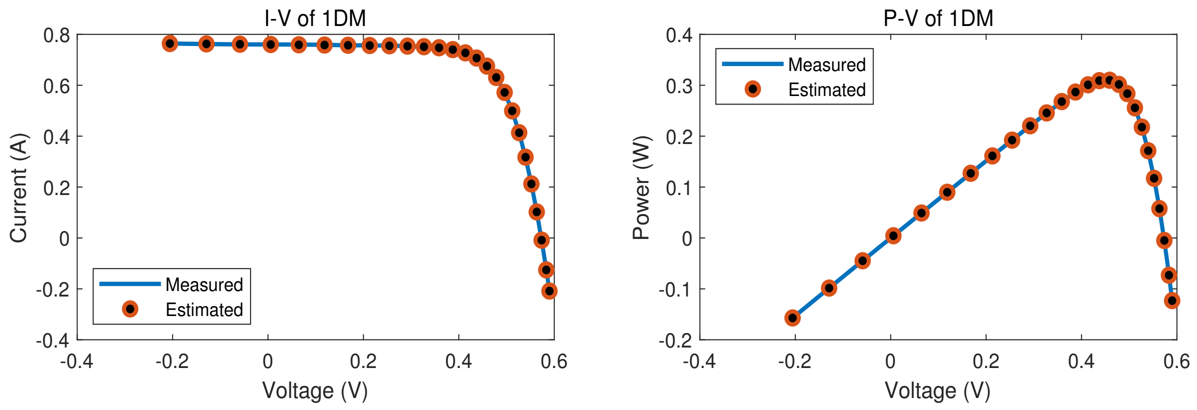

Figure 12. Curve fitting for the data derived from measurements and experiments identified by the CALA: the left side shows the $I - V$ characteristics; the right side shows the $P - V$ characteristics.

8. Conclusions and future directions

This study proposes an enhanced artificial lemming algorithm (CALA) that significantly improves the original algorithm's performance in numerical optimization and engineering applications through the innovative integration of three optimization strategies. In terms of algorithm design, we first introduce a linear inertia weight strategy during the global exploration phase, which effectively enhances both convergence speed and global search capability. Second, we implement a historical best-guided strategy in the local exploitation phase to substantially improve individual search efficiency. Finally, to address the original algorithm's tendency to fall into local optima, we innovatively incorporate a crossover strategy during iterations, enabling effective escape from local optima and further acceleration of convergence.

To comprehensively evaluate the CALA's performance advantages, we conducted systematic validation using representative benchmark sets from CEC2017 (29 test functions) and CEC2022 (12 test functions). Experimental results demonstrate that compared to current mainstream metaheuristic algorithms, the CALA exhibits superior comprehensive performance in most test cases, showing particular advantages in solution quality, convergence speed, algorithmic stability, and high-dimensional problem-solving capability. Furthermore, practical application tests on four typical engineering optimization problems and a photovoltaic model parameter estimation task further confirm the CALA's practical value and reliability in real-world engineering scenarios.

Although the CALA has demonstrated excellent optimization performance, our research has identified several areas for improvement. The primary limitation is the increased computational complexity resulting from the combined use of multiple enhancement strategies, which affects operational efficiency to some extent. To address this limitation, we plan to focus on the following aspects in future research: First, we will explore parallel computing techniques or integration with other advanced algorithms to significantly reduce computational costs while maintaining algorithmic precision. Second, we intend to incorporate advanced operators such as dynamic population evolution and quantum rotation gates to further enhance the algorithm's robustness and adaptability. Preliminary tests indicate that the CALA shows promising application potential in classical optimization problems like knapsack [58], economic dispatch [59], and traveling salesman [60] problems in daily life. Concurrently, we are developing a multi-objective version of the CALA to extend its applicability to complex multi-objective optimization problems, which will establish a foundation for broader applications of the algorithm.

Author contributions

Yan Zhong, Xiongfa Mai, and Haiyan Luo designed the algorithm. Yan Zhong conducted the experiments, and Xiongfa Mai and Li-Bin Liu guided the analysis of the experimental results. Yan Zhong wrote the main manuscript text, and Xiongfa Mai and Li-Bin Liu modified it. All authors reviewed and approved the final manuscript.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work is supported by the Guangxi Natural Science Foundation (2025GXNSFAA069738), the Guangxi Science and Technology Base and Special Talents (Guike AD23023003), the National Natural Science Foundation of China (12361087), and the Innovation Project of Guangxi Graduate Education (YCSW2025529).

Conflict of interest

Li-Bin Liu is a guest editor for *Networks and Heterogeneous Media* (NHM) and was not involved in the editorial review or the decision to publish this article. All authors declare that there are no competing interests.

References

1. Q. Li, S.Y. Liu, X.S. Yang, Influence of initialization on the performance of metaheuristic optimizers, *Appl. Soft Comput.*, **91** (2020), 106193. https://doi.org/10.1016/j.asoc.2020.106193

2. H. Su, D. Zhao, A. A. Heidari, L. Liu, X. Zhang, M. Mafarja, et al., RIME: A physics-based optimization, *Neurocomputing*, **532** (2023), 183–214. https://doi.org/10.1016/j.neucom.2023.02.010

3. S. Abbasi, A. M. Rahmani, A. Balador, A. Sahafi, A fault-tolerant adaptive genetic algorithm for service scheduling in internet of vehicles, *Appl. Soft Comput.*, **143** (2023), 110413. https://doi.org/10.1016/j.asoc.2023.110413

4. B. Zhou, Z. Zhao, An adaptive artificial bee colony algorithm enhanced by Deep Q-Learning for milk-run vehicle scheduling problem based on supply hub, *Knowledge Based Syst.*, **264** (2023), 110367. https://doi.org/10.1016/j.knosys.2023.110367

5. Z. Wang, L. Shao, S. Yang, J. Wang, D. Li, CRLM: A cooperative model based on reinforcement learning and metaheuristic algorithms of routing protocols in wireless sensor networks, *Comput. Networks*, **236** (2023), 110019. https://doi.org/10.1016/j.comnet.2023.110019

6. Y. Chen, G. Wang, B. Yin, C. Ma, Z. Wu, M. Gao, Basketball team optimization algorithm (BTOA): a novel sport-inspired meta-heuristic optimizer for engineering applications, *Sci. Rep.*, **15** (2025), 21629. https://doi.org/10.1038/s41598-025-05477-0

7. Y. Xiao, Y. Guo, H. Cui, Y. Wang, J. Li, Y. Zhang, IHAOAVOA: An improved hybrid aquila optimizer and African vultures optimization algorithm for global optimization problems, *Math. Biosci. Eng.*, **19** (2022), 10963–11017. https://doi.org/10.3934/mbe.2022512

8. D. H. Wolpert, W. G. Macready, No free lunch theorems for optimization, *IEEE Trans. Evolut. Comput.*, **1** (1997), 67–82. https://doi.org/10.1109/4235.585893

9. R. Zheng, H. Jia, L. Abualigah, S. Wang, D. Wu, An improved remora optimization algorithm with autonomous foraging mechanism for global optimization problems, *Math. Biosci. Eng.*, **19** (2022), 3994–4037. https://doi.org/10.3934/mbe.2022184

10. R. Zheng, H. Jia, L. Abualigah, Q. Liu, S. Wang, Deep ensemble of slime mold algorithm and arithmetic optimization algorithm for global optimization, *Processes*, **9** (2021), 1774. https://doi.org/10.3390/pr9101774

11. H. Liu, R. Zhou, X. Zhong, Y. Yao, W. Shan, J. Yuan, et al., Multi-strategy enhanced crested porcupine optimizer: CAPCPO, *Mathematics*, **12** (2024), 3080. https://doi.org/10.3390/math12193080

12. H. Liu, L. Liu, X. Mai, D. Guo, A new hybrid Lévy Quantum-behavior Butterfly Optimization Algorithm and its application in NL5 Muskingum model, *Electron. Res. Arch.*, **32** 2024. https://doi.org/10.3934/era.2024109

13. Y. Xiao, H. Cui, R. Abu Khurma, P. A. Castillo, Artificial lemming algorithm: a novel bionic meta-heuristic technique for solving real-world engineering optimization problems, *Artif. Intell. Rev.*, **58** (2025), 84. https://doi.org/10.1007/s10462-024-11023-7

14. M. Abdel-Basset, R. Mohamed, S. A. Abdel Azeem, M. Jameel, M. Abouhawwash, Kepler optimization algorithm: A new metaheuristic algorithm inspired by Kepler's laws of planetary motion, *Knowledge Based Syst.*, **268** (2023), 110454. https://doi.org/10.1016/j.knosys.2023.110454

15. M. H. Amiri, N. Mehrabi Hashjin, M. Montazeri, S. Mirjalili, N. Khodadadi, Hippopotamus optimization algorithm: A novel nature-inspired optimization algorithm, *Sci. Rep.*, **14** (2024), 5032. https://doi.org/10.1038/s41598-024-54910-3

16. L. Deng, S. Liu, Snow ablation optimizer: A novel metaheuristic technique for numerical optimization and engineering design, *Expert Syst. Appl.*, **225** (2023), 120069. https://doi.org/10.1016/j.eswa.2023.120069

17. G. Hu, C. Gong, X. Li, Z. Xu, CGKOA: An enhanced Kepler optimization algorithm for multi-domain optimization problems, *Comput. Method. Appl. Mech. Eng.*, **425** (2024), 116964. https://doi.org/10.1016/j.cma.2024.116964

18. S. Pei, G. Sun, L. Tong, An improved hippopotamus optimization algorithm based on adaptive development and solution diversity enhancement, *PeerJ Comput. Sci.*, **11** (2025), e2901. https://doi.org/10.7717/peerj-cs.2901

19. Y. Xiao, H. Cui, A. G. Hussien, F. A. Hashim, MSAO: A multi-strategy boosted snow ablation optimizer for global optimization and real-world engineering applications, *Adv. Eng. Inform.*, **61** (2024), 102464. https://doi.org/10.1016/j.aei.2024.102464

20. Y. Han, GALA: group merging artificial lemming algorithm for gene selection, in *2025 International Conference on Computer Science, Technology and Engineering (ICCSTE)*, 2025, 244–247. https://doi.org/10.1109/ICCSTE65902.2025.11138413

21. X. Zhu, C. Jia, J. Zhao, C. Xia, W. Peng, J. Huang, et al., An enhanced artificial lemming algorithm and its application in UAV path planning, *Biomimetics*, **10** (2025), 377. https://doi.org/10.3390/biomimetics10060377

22. Y. Xie, Z. Sun, K. Yuan, Z. Sun, 3D UAV route optimization in complex environments using an enhanced artificial lemming algorithm, *Symmetry*, **17** (2025), 946. https://doi.org/10.3390/sym17060946

23. Y. Tan, J. Wang, B. Wang, An improved bionic artificial lemming algorithm for global optimization and cloud task-scheduling problems, *Biomimetics*, **10** (2025), 572. https://doi.org/10.3390/biomimetics10090572

24. J. Zhang, Y. Li, C. Li, X. Mei, J. Zhou, Application of soft computing represented by regression machine learning model and artificial lemming algorithm in predictions for hydrogen storage in metal-organic frameworks, *Materials*, **18** (2025), 3122. https://doi.org/10.3390/ma18133122

25. P. Qu, X. Song, Z. Zhou, Research on path planning for mobile robot using the enhanced artificial lemming algorithm, *Mathematics*, **13** (2025), 3533. https://doi.org/10.3390/math13213533

26. Y. Han, Feature selection algorithm based on Q-learning and artificial lemming algorithm for medical data, in *2025 7th International Conference on Electronics and Communication, Network and Computer Technology (ECNCT)*, **1** (2025), 737–743. https://doi.org/10.1109/ECNCT66493.2025.11172427

27. R. C. Eberhart, Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, in *Proceedings of the 2000 Congress on Evolutionary Computation*, **1** (2000), 84–88. https://doi.org/10.1109/CEC.2000.870279

28. Y. Shi, R. C. Eberhart, Empirical study of particle swarm optimization, in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99*, **3** (1999), 1945–1950. https://doi.org/10.1109/CEC.1999.785511

29. J. C. Bansal, P. K. Singh, M. Saraswat, A. Verma, S. S. Jadon, A. Abraham, Inertia weight strategies in particle swarm optimization, in *2011 Third World Congress on Nature and Biologically Inspired Computing*, 2011, 633–640. https://doi.org/10.1109/NaBIC.2011.6089659

30. W. Gao, S. Liu, L. Huang, A global best artificial bee colony algorithm for global optimization, *J. Comput. Appl. Math.*, **236** (2012), 2741–2753. https://doi.org/10.1016/j.cam.2012.01.013

31. J. Liu, H. Zhu, Q. Ma, L. Zhang, H. Xu, An artificial bee colony algorithm with guide of global & local optima and asynchronous scaling factors for numerical optimization, *Appl. Soft Comput.*, **37** (2015), 608–618. https://doi.org/10.1016/j.asoc.2015.08.021

32. M. K. Naik, R. Panda, A novel adaptive cuckoo search algorithm for intrinsic discriminant analysis based face recognition, *Appl. Soft Comput.*, **38** (2016), 661–675. https://doi.org/10.1016/j.asoc.2015.10.039

33. A. B. Meng, Y. C. Chen, H. Yin, S. Z. Chen, Crisscross optimization algorithm and its application, *Knowledge Based Syst.*, **67** (2014), 218–229. https://doi.org/10.1016/j.knosys.2014.05.004

34. J. Kennedy, R. Eberhart, Particle swarm optimization, in *Proceedings of ICNN'95 - International Conference on Neural Networks*, **4** (1995), 1942–1948. https://doi.org/10.1109/ICNN.1995.488968

35. S. Mirjalili, SCA: A Sine Cosine algorithm for solving optimization problems, *Knowl-Based Syst.*, **96** (2016), 120–133. https://doi.org/10.1016/j.knosys.2015.12.022

36. H. Jia, H. Rao, C. Wen, S. Mirjalili, Crayfish optimization algorithm, *Artif. Intell. Rev.*, **56** (2023), 1919–1979. https://doi.org/10.1007/s10462-023-10567-4

37. A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, H. Chen, Harris hawks optimization: Algorithm and applications, *Future Gener. Comput. Syst.*, **97** (2019), 849–872. https://doi.org/10.1016/j.future.2019.02.028

38. E. H. Houssein, D. Oliva, N. Abdel Samee, N. F. Mahmoud, M. M. Emam, Liver cancer algorithm: A novel bio-inspired optimizer, *Comput. Biol. Med.*, **165** (2023), 107389. https://doi.org/10.1016/j.compbiomed.2023.107389

39. R. Tanabe, A. S. Fukunaga, Improving the search performance of SHADE using linear population size reduction, in *2014 IEEE Congress on Evolutionary Computation (CEC)*, 2014, 1658–1665. https://doi.org/10.1109/CEC.2014.6900380

40. K. Hussain, M. N. M. Salleh, S. Cheng, Y. Shi, On the exploration and exploitation in popular swarm-based metaheuristic algorithms, *Neural Comput. Appl.*, **31** (2019), 7665–7683. https://doi.org/10.1007/s00521-018-3592-0

41. B. Morales-Castañeda, D. Zaldivar, E. Cuevas, F. Fausto, A. Rodríguez, A better balance in metaheuristic algorithms: Does it exist?, *Swarm Evol. Comput.*, **54** (2020), 100671. https://doi.org/10.1016/j.swevo.2020.100671

42. G. Wu, R. Mallipeddi, P. N. Suganthan, Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization. Available from: https://www.researchgate.net/publication/317228117.

43. A. Kumar, K. V. Price, A. W. Mohamed, A. A. Hadi, Problem Definitions and Evaluation Criteria for the CEC 2022 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization.

44. P. B. Dao, On Wilcoxon rank sum test for condition monitoring and fault detection of wind turbines, *Appl. Energy*, **318** (2022), 119209. https://doi.org/10.1016/j.apenergy.2022.119209

45. D. Annaratone, *Pressure Vessel Design*, Springer, Berlin, Heidelberg, 2007. https://doi.org/10.1007/978-3-540-49144-6

46. E. Sandgren, Nonlinear integer and discrete programming in mechanical design optimization, *J. Mech. Des.*, **112** (1990), 223–229. https://doi.org/10.1115/1.2912596

47. M. Alabdulhafith, H. Batra, N. M. A. Samee, M. A. Al-Betar, A. Almomani, D. Izci, et al., A modified Bonobo optimizer with its application in solving engineering design problems, *IEEE Access*, **12** (2024), 134948–134984. https://doi.org/10.1109/ACCESS.2024.3455550

48. J. S. Arora, 2-Optimum design problem formulation, *Introduction to Optimum Design (Second Edition)*, Elsevier, 2004, 15–54. https://doi.org/10.1016/B978-012064155-0/50002-1

49. A. D. Belegundu, J. S. Arora, A study of mathematical programming methods for structural optimization. Part I: Theory, *Int. J. Numer. Meth. Eng.*, **21** (1985), 1583–1599. https://doi.org/10.1002/nme.1620210904

50. C. A. Coello Coello, E. Mezura Montes, Constraint-handling in genetic algorithms through the use of dominance-based tournament selection, *Adv. Eng. Inform.*, **16** (2002), 193–203. https://doi.org/10.1016/S1474-0346(02)00011-3

51. A. T. Kamil, H. M. Saleh, I. H. Abd-Alla, A multi-swarm structure for particle swarm optimization: Solving the welded beam design problem, *J. Phys.: Conf. Ser.*, **1804** (2021), 012012. https://doi.org/10.1088/1742-6596/1804/1/012012

52. T. Easwarakhanthan, J. Bottin, I. Bouhouch, C. Boutrit, Nonlinear minimization algorithm for determining the solar cell parameters with microcomputers, *Int. J. Sol. Energy*, **4** (1986), 1–12. https://doi.org/10.1080/01425918608909835

53. T. Zhou, C. Shang, Parameter identification of solar photovoltaic models by multi strategy sine–cosine algorithm, *Energy Sci. Eng.*, **12** (2024), 1422–1445. https://doi.org/10.1002/ese3.1673

54. X. Chen, K. Yu, W. Du, W. Zhao, G. Liu, Parameters identification of solar cell models using generalized oppositional teaching learning based optimization, *Energy*, **99** (2016), 170–180. https://doi.org/10.1016/j.energy.2016.01.052

55. X. Yuan, Y. He, L. Liu, Parameter extraction of solar cell models using chaotic asexual reproduction optimization, *Neural Comput. Appl.*, **26** (2015), 1227–1239. https://doi.org/10.1007/s00521-014-1795-6

56. Z. Feng, D. Zhu, H. Guo, J. Xue, C. Zhou, A Jaya algorithm based on self-adaptive method for parameters identification of photovoltaic cell and module, *Clust. Comput.*, **28** (2025), 145. https://doi.org/10.1007/s10586-024-04877-7

57. W. Gong, Z. Cai, Parameter extraction of solar cell models using repaired adaptive differential evolution, *Sol. Energy*, **94** (2013), 209–220. https://doi.org/10.1016/j.solener.2013.05.007

58. M. Abdel-Basset, R. Mohamed, M. Abouhawwash, A. M. Alshamrani, A. W. Mohamed, K. Sallam, Binary light spectrum optimizer for knapsack problems: an improved model, *Alex. Eng. J.*, **67** (2023), 609–632. https://doi.org/10.1016/j.aej.2022.12.025

59. M. Ellahi, G. Abbas, A Hybrid Metaheuristic Approach for the Solution of Renewables-Incorporated Economic Dispatch Problems, *IEEE Access*, **8** (2020), 127608–127621. https://doi.org/10.1109/ACCESS.2020.3008570

60. E. Osaba, X. S. Yang, J. Del Ser, Traveling salesman problem: a perspective review of recent research and new results with bio-inspired metaheuristics, *Nature-Inspired Computation and Swarm Intelligence*, Academic Press, 2020, 135–164, https://doi.org/10.1016/B978-0-12-819714-1.00020-8

Appendix

Table A1. CEC2017 test functions.

Type	ID	CEC2017 function name	Dimension	fmin
Unimodal	F1	Shifted and rotated bent cigar function	30	100
	F2*	Shifted and rotated sum of different power function	30	200
	F3	Shifted and rotated Zakharow function	30	300
Multimodal	F4	Shifted and rotated Rosenbrock's function	30	400
	F5	Shifted and rotated Rastrigin's function	30	500
	F6	Shifted and rotated expanded Scaffer's F6 function	30	600
	F7	Shifted and rotated Lunacek bi-rastrigin finction	30	700
	F8	Shifted and rotated non-continuous Rastrigin's function	30	800
	F9	Shifted and rotated Levy function	30	900
	F10	Shifted and rotated Schwefel's function	30	1000
Hybrid	F11	Hybrid function 1 (N=3)	30	1100
	F12	Hybrid function 2 (N = 3)	30	1200
	F13	Hybrid function 3 (N = 3)	30	1300
	F14	Hybrid function 4 (N = 4)	30	1400
	F15	Hybrid function 5 (N = 4)	30	1500
	F16	Hybrid function 6 (N = 4)	30	1600
	F17	Hybrid function 7 (N = 5)	30	1700
	F18	Hybrid function 8 (N = 5)	30	1800
	F19	Hybrid function 9 (N = 5)	30	1900
	F20	Hybrid function 10 (N = 6)	30	2000
Composition	F21	Composition function 1 (N=3)	30	2100
	F22	Composition function 2 (N = 3)	30	2200
	F23	Composition function 3 (N = 4)	30	2300
	F24	Composition function 4 (N = 4)	30	2400
	F25	Composition function 5 (N = 5)	30	2500
	F26	Composition function 6 (N = 5)	30	2600
	F27	Composition function 7 (N = 6)	30	2700
	F28	Composition function 8 (N = 6)	30	2800
	F29	Composition function 9 (N = 3)	30	2900
	F30	Composition function 10 (N = 3)	30	3000
		Search Range $[-100, 100]^D$		

*F2 has been officially removed due to its unstable behavior on high-dimensional issues.

Table A2. Experimental results of 8 algorithms on hybrid and composition functions of CEC2017 (Dim = 30).

Function	Metric	PSO	SCA	COA	HHO	LCA	LSHADE	ALA	CALA
F11	Ave	1.2880E+03	2.7391E+03	6.8310E+03	1.2827E+03	1.3872E+04	1.1924E+03	1.2172E+03	**1.1914E+03**
	Std	6.1313E+01	6.0872E+02	1.4897E+03	4.8923E+01	2.6019E+03	**2.5759E+01**	3.8619E+01	4.5732E+01
F12	Ave	3.8445E+07	1.9443E+09	9.8922E+09	2.0020E+07	1.7057E+10	4.2925E+05	9.2911E+05	**7.3266E+04**
	Std	9.9897E+07	4.8592E+08	2.4347E+09	1.9573E+07	3.7356E+09	3.4830E+05	1.0450E+06	**4.2782E+04**
F13	Ave	2.7466E+06	7.2249E+08	4.4897E+09	5.5291E+05	1.2723E+10	5.5220E+04	3.2719E+04	**2.3142E+04**
	Std	1.3100E+07	4.6706E+08	2.4765E+09	3.0432E+05	5.6964E+09	2.5461E+04	**2.0489E+04**	2.1209E+04
F14	Ave	5.2149E+04	3.7233E+05	1.3938E+06	3.4747E+05	3.0273E+07	1.8844E+03	1.7604E+03	**1.4907E+03**
	Std	6.7856E+04	3.1048E+05	1.2133E+06	4.2680E+05	2.8223E+07	1.5069E+02	1.4179E+02	**4.2444E+01**
F15	Ave	2.2939E+04	3.1138E+07	2.0469E+08	8.5648E+04	2.2198E+09	**4.1979E+03**	9.8735E+03	5.6941E+03
	Std	1.8582E+04	2.6242E+07	2.0052E+08	4.6715E+04	1.0735E+09	1.0562E+05	9.0181E+03	**8.4595E+03**
F16	Ave	2.4224E+03	3.9274E+03	5.1441E+03	3.4906E+03	7.0843E+03	2.8146E+03	2.4415E+03	**2.3777E+03**
	Std	2.7486E+02	**2.0860E+02**	6.3137E+02	4.4090E+02	1.2175E+03	4.8413E+02	2.3483E+02	2.6029E+02
F17	Ave	1.9772E+03	2.6054E+03	3.3891E+03	2.6157E+03	7.7415E+03	2.0117E+03	2.0435E+03	**1.8615E+03**
	Std	1.2435E+02	1.9837E+02	7.2389E+02	2.9340E+02	6.4804E+03	1.0642E+02	1.3877E+02	**8.8460E+01**
F18	Ave	8.2039E+05	7.2546E+06	1.8266E+07	2.1041E+06	3.1722E+08	3.2864E+05	4.9433E+04	**3.3795E+04**
	Std	7.2181E+05	4.3053E+06	1.1963E+07	2.6936E+06	1.9772E+08	1.2710E+05	**1.9944E+04**	2.1171E+04
F19	Ave	2.6483E+04	5.8711E+07	1.9235E+08	5.7781E+05	2.0183E+09	**3.2926E+03**	1.8949E+04	5.7206E+03
	Std	6.6931E+04	2.7226E+07	1.5772E+08	4.5140E+05	1.0540E+09	1.4383E+05	1.7000E+04	**5.5043E+03**
F20	Ave	2.2871E+03	2.7561E+03	2.8195E+03	2.7377E+03	3.5059E+03	2.3693E+03	2.3738E+03	**2.2479E+03**
	Std	1.5431E+02	1.6849E+02	1.8052E+02	1.7568E+02	1.7866E+02	1.5949E+02	1.7724E+02	**1.5377E+02**
F21	Ave	2.3836E+03	2.5856E+03	2.6983E+03	2.5535E+03	2.7911E+03	2.4517E+03	2.3860E+03	**2.3739E+03**
	Std	2.3647E+01	2.2676E+01	3.4659E+01	4.4659E+01	5.1824E+01	8.2770E+01	2.6859E+01	**1.8055E+01**
F22	Ave	3.8958E+03	8.8253E+03	8.5849E+03	6.3818E+03	1.0549E+04	**2.3000E+03**	5.7023E+03	2.7066E+03
	Std	1.5987E+03	2.3724E+03	1.0257E+03	1.8648E+03	1.3620E+03	**2.4816E-06**	1.8069E+03	1.0670E+03
F23	Ave	2.8216E+03	3.0487E+03	3.4616E+03	3.1000E+03	3.7511E+03	2.8012E+03	2.7476E+03	**2.7263E+03**
	Std	5.4443E+01	2.8767E+01	1.1935E+02	1.0142E+02	1.6733E+02	9.9439E+02	**2.0973E+01**	2.1430E+01
F24	Ave	3.0000E+03	3.2140E+03	3.6779E+03	3.3808E+03	4.1232E+03	2.9754E+03	2.9180E+03	**2.9063E+03**
	Std	4.9778E+01	2.7523E+01	1.1264E+02	1.2681E+02	1.4656E+02	6.0036E+01	2.8887E+01	**2.4660E+01**
F25	Ave	2.9080E+03	3.3417E+03	4.5031E+03	2.9418E+03	6.0664E+03	**2.8871E+03**	2.8943E+03	2.8985E+03
	Std	2.2117E+01	1.2161E+02	3.1379E+02	3.0797E+01	5.7541E+02	**1.9892E-01**	1.0885E+01	2.0530E+01
F26	Ave	4.7311E+03	7.3646E+03	1.0393E+04	7.4642E+03	1.2284E+04	5.0503E+03	4.6657E+03	**4.5409E+03**
	Std	5.1067E+02	5.1191E+02	8.2461E+02	8.2815E+02	1.2383E+03	8.7639E+02	**2.4285E+02**	4.1855E+02
F27	Ave	3.2444E+03	3.4873E+03	4.1886E+03	3.4122E+03	5.0805E+03	**3.2071E+03**	3.2211E+03	3.2245E+03
	Std	3.1888E+01	4.9927E+01	1.8314E+02	1.2807E+02	3.7318E+02	**6.1868E-05**	1.3596E+01	1.7860E+01
F28	Ave	3.3031E+03	4.0837E+03	6.5674E+03	3.3230E+03	8.2569E+03	**3.2068E+03**	3.3516E+03	3.2235E+03
	Std	6.4763E+01	2.4022E+02	5.9629E+02	**2.1061E+01**	6.0082E+02	2.5226E+01	5.2504E+02	2.6296E+01
F29	Ave	3.6558E+03	4.8937E+03	7.0986E+03	4.5050E+03	1.2307E+04	3.8265E+03	3.7780E+03	**3.6272E+03**
	Std	2.1263E+02	2.3725E+02	8.2203E+02	4.0134E+02	8.3800E+03	9.8426E+02	1.8779E+02	**1.7067E+02**
F30	Ave	1.9689E+05	1.2188E+08	7.4168E+08	4.0964E+06	2.4503E+09	3.9730E+04	3.8604E+04	**1.0062E+04**
	Std	5.5388E+05	4.6709E+07	3.2174E+08	2.6802E+06	1.2470E+09	1.0572E+08	3.7283E+04	**3.4511E+03**

Table A3. CEC2022 test functions

Type	ID	CEC2022 function name	Dimension	fmin
Unimodal	F1	Shifted and full rotated Zakharov function	10	300
Multimodal	F2	Shifted and full rotated Rosenbrock's function	10	400
	F3	Shifted and full rotated Rastrigin's function	10	600
	F4	Shifted and full rotated non-continuous Rastrigin's function	10	800
	F5	Shifted and full rotated Levy function	10	900
Hybrid	F6	Hybrid function 1 (N = 3)	10	1800
	F7	Hybrid function 2 (N = 6)	10	2000
	F8	Hybrid function 3 (N = 5)	10	2200
Composition	F9	Composition function 1 (N = 5)	10	2300
	F10	Composition function 2 (N = 4)	10	2400
	F11	Composition function 3 (N = 5)	10	2600
	F12	Composition function 4 (N = 6)	10	2700
		Search Range $[-100, 100]^D$		