Trend-Aware Mechanism for Metaheuristic Algorithms

Junbo Jacob Lian[a,b], Kaichen Ouyang[c], Rui Zhong[d], Yujun Zhang[e], Shipeng Luo[f], Ling Ma[a], Xincan Wu[a] and Huiling Chen[g,*]

[a]*School of Mathematics and Computer Sciences, Zhejiang A & F University, Hangzhou, China*
[b]*McCormick School of Engineering, Northwestern University, Evanston, USA*
[c]*School of Mathematics, University of Science and Technology of China, Hefei, China*
[d]*Information Initiative Center, Hokkaido University, Sapporo, Japan*
[e]*School of New Energy, Jingchu University of Technology, Jingmen, China*
[f]*School of Mechanical and Electrical Engineering, Northeast Forestry University, Harbin, China*
[g]*School of Computer Science and Artificial Intelligence, Wenzhou University, Wenzhou, China*

ARTICLE INFO

Keywords:
Trend-Aware Mechanism
Metaheuristic Algorithm
Optimization
Historical search
Evolutionary Computation

ABSTRACT

In metaheuristic algorithms, historical search-position data often remain underutilized despite their potential to reveal valuable movement trends and promising search directions. To address this limitation, we propose the Trend-Aware Mechanism (TAM), which leverages historical position information to enhance the position updating process. TAM identifies the primary direction of movement by deriving a trend line from the population's positions over the two most recent iterations. It evaluates candidate optimal positions by assessing the fitness of the K nearest points along this trend line. To effectively balance exploration and exploitation, TAM employs an adaptive covariance mechanism to generate high-dimensional random vectors, dynamically adjusting the update strategies. We integrate TAM with four prominent meta-heuristic algorithms—PSO, SHADE, JaDE, and CMA-ES—and conduct an extensive parameter sensitivity analysis to ensure robustness. Comparative evaluations across five performance metrics demonstrate that TAM significantly improves search efficiency and consistently achieves superior results on standard benchmark functions. Moreover, TAM's practical applicability is validated through real-world problems in engineering design, feature selection, and photovoltaic model parameter extraction. The open-source implementation of TAM will be publicly available at https://github.com/junbolian/Trend-Aware-Mechanism.

1. Introduction

Metaheuristic algorithms have emerged as effective solutions for addressing complex and high-dimensional optimization challenges, frequently surpassing conventional optimization methods in performance [1]. They are used in medical image segmentation [2], photovoltaic models [3, 4], quality analysis [5, 6], AI model optimization [7, 8], engineering design [9, 10], and many other fields. These algorithms typically balance exploration and exploitation processes by leveraging stochasticity to explore the search space while exploiting promising areas based on historical information. However, a significant challenge remains: how to effectively harness the wealth of search history data accumulated during the iterative search process. Despite the abundance of position and fitness information available from previous iterations, many existing algorithms underutilize this historical data, limiting their ability to fully capture the underlying search dynamics and trends [11].

These algorithms can be broadly divided into two types: individual-based and population-based. Individual-based methods, exemplified by hill climbing and simulated annealing (SA) [12], prioritize refining a single solution at a time. In contrast, population-based strategies operate on multiple solutions concurrently, enabling a more comprehensive exploration of the solution space [13]. Population-based optimization algorithms can be broadly classified into three main categories: Population-based optimization algorithms, however, are typically categorized into three major groups:

*Corresponding author

junbolian@qq.com (J.J. Lian); Jacoblian@u.northwestern.edu (J.J. Lian); oykc@mail.ustc.edu.cn (K. Ouyang); zhongrui@iic.hokudai.ac.jp (R. Zhong); zhangyj069@gmail.com (Y. Zhang); donglinluo2426@gmail.com (S. Luo); maling@stu.zafu.edu.cn (L. Ma); y.yr.r@qq.com (X. Wu); chenhuiling.jlu@gmail.com (H. Chen)
ORCID(s): 0000-0001-7602-0022 (J.J. Lian); 0009-0003-5937-5229 (K. Ouyang); 0000-0003-4605-5579 (R. Zhong); 0000-0003-3016-8843 (Y. Zhang); 0009-0000-7577-7453 (S. Luo); 0009-0006-5874-3511 (L. Ma); 0009-0003-8698-6084 (X. Wu); 0000-0002-7714-9693 (H. Chen)

(1) Evolutionary algorithms, such as differential evolution (DE) [14], genetic algorithms (GA) [15], evolutionary strategy (ES) [16], and evolutionary programming (EP) [17]. These algorithms simulate the natural process of survival of the fittest (Darwinian evolution) to guide the population toward optimal solutions [18]. (2) Swarm intelligence algorithms, including ant colony optimization (ACO) [19], Parrot Optimizer (PO) [2], and educational competition optimizer (ECO) [20], which derive global optima by mimicking group intelligence behaviors [21]. (3) Algorithms based on science and mathematical principles, such as the Gravitational Search Algorithm (GSA) [22] and the RIME algorithm [23], which achieve optimization by leveraging mathematical laws and principles.

All metaheuristic algorithms share three typical characteristics: (1) The ability to escape local optima, a fundamental requirement often achieved by incorporating randomization. (2) The need for hyperparameters, with the number of such parameters varying across algorithms. (3) The challenge of balancing global exploration and local exploitation. Excessive global exploration may prevent the algorithm from converging, while too much local exploitation can lead to entrapment in local optima. However, many traditional algorithms, despite their simplicity, suffer from limitations in their structure and performance [2]. The No Free Lunch theorem [24] further underscores that no single algorithm can solve every optimization problem. As a result, improving existing algorithms to enhance their performance or broaden their applicability is crucial. Such improvements typically take two forms: incorporating novel strategies into the algorithm or hybridizing multiple algorithms to create a more robust solution. These approaches ultimately aim to enhance the algorithm's overall effectiveness across a variety of optimization tasks.

In recent years, several researchers have introduced strategies to improve the efficiency of metaheuristic algorithms by incorporating mechanisms that adaptively adjust the balance between exploration and exploitation. Techniques such as jumping strategy [1], Q-learning [25], multiple search preferences [26], guided learning strategy [27], Latin hypercube sampling [28], vector-encirclement strategy [29], opposite learning method [30], mutation strategy [31], and historical knowledge transfer [32] have demonstrated the effectiveness of adaptive mechanisms in improving convergence speed and solution quality. However, a deeper understanding of search trends—specifically how historical search positions can inform future movements—remains underexplored.

To bridge this gap, we propose the Trend-Aware Mechanism (TAM), which explicitly leverages historical search position data to guide the optimization process. TAM analyzes position data from previous iterations to estimate movement trends and pinpoint regions in the search space with a higher likelihood of containing optimal solutions. By evaluating the fitness of nearby points along these trends, TAM dynamically refines the position update strategy, effectively balancing trend-driven guidance with random exploration. The core of the proposed method is the calculation of a trend line between the positions of individuals across consecutive iterations. TAM evaluates the fitness landscape around this trend and applies an adaptive covariance mechanism to generate random vectors that blend historical movement trends with stochastic exploration. This allows the algorithm to dynamically shift between guided exploitation and exploratory behavior based on current search conditions. Moreover, TAM does not increase the number of evaluations of the original algorithm.

Methods that leverage historical information have achieved remarkable success in evolutionary computation: Adaptive Differential Evolution with Optional External Archive (JaDE) [33] and Success History-Based Adaptive Differential Evolution (SHADE) [34] improve performance by recording successful search steps to adjust control parameters, while the Memory Backtracking Strategy (MBS) [11] recalls group-level memories to expand exploration. Opposition-based learning (OBL) accelerates convergence by evaluating each candidate solution together with its opposite [35, 36], and historical knowledge-transfer models reuse solutions or information from previous tasks or iterations [37]. Building on—but differing fundamentally from—these memory-based or transfer-based strategies, the Trend-Aware Mutation (TAM) proposed in this paper fits a trend line through each individual's historical positions and exploits that trajectory to generate new candidate directions in real time, embedding within-task search history directly into the update rule without generating opposite points, transferring external knowledge, or incurring additional function evaluations.

In this study, we integrate the TAM into four leading metaheuristic algorithms: Particle Swarm Optimization (PSO), originally proposed by Kennedy and Eberhart, which is characterized by its use of social and cognitive components to guide particles towards optimal solutions [38]; Success History-Based Adaptive Differential Evolution (SHADE), introduced by Tanabe and Fukunaga , which incorporates success history to adaptively adjust control parameters and improve convergence [34]; Adaptive Differential Evolution with Optional External Archive (JaDE), proposed by Zhang et al., which enhances differential evolution by using an external archive to store and reuse promising solutions [33]; and Covariance Matrix Adaptation Evolution Strategy (CMA-ES), introduced by Hansen et al., which adapts the covariance matrix of the search distribution to guide the population toward regions of higher fitness [39]. Through extensive experiments on a suite of standard benchmark functions, we demonstrate that integrating TAM significantly enhances the population's search capabilities, improving the performance of each algorithm without increasing the number of function evaluations. The key contributions of this paper are outlined as follows:

- **Introduction of the Trend-Aware Mechanism (TAM):** We propose TAM as a novel approach to utilizing historical search data to guide future movements by estimating search trends and dynamically adjusting position updates based on local fitness evaluations.
- **Adaptive Covariance Mechanism:** We introduce an adaptive covariance mechanism that generates high-dimensional random vectors to balance historical search trends with stochastic exploration, allowing for a dynamic adjustment of position updates.
- **Integration with Famous Algorithms:** TAM is integrated with four leading metaheuristic algorithms—PSO, SHADE, JaDE, and CMA-ES—and its performance is extensively evaluated across a set of standard benchmark functions.
- **Performance Improvement:** Our experimental findings reveal that TAM enhances search efficiency substantially, achieving notable improvements in convergence speed and solution precision compared to the baseline algorithms.

This paper explores the TAM methodology and its integration with various algorithms, presenting experimental results and their implications for metaheuristic optimization. The structure is as follows: Section 2 describes the mechanisms and framework of TAM; Section 3 examines the parameter sensitivity of TAM; Section 4 compares TAM's performance before and after improvement using five evaluation metrics; Section 5 highlights TAM's practical applications in engineering design, feature selection, and photovoltaic model parameter identification; and Section 6 concludes with a summary and future research directions.

2. Trend-Aware Mechanism

In this section, we present the TAM, a trend-aware update strategy aimed at improving the search efficiency and accuracy of population-based optimization algorithms. This mechanism leverages historical search positions and fitness information to incorporate trend awareness into the search process. TAM optimizes individual position updates by analyzing the population's search trajectory during previous iterations. Additionally, an adaptive covariance mechanism generates high-dimensional trend vectors that combine stochasticity and stability, facilitating a more intelligent update strategy.

2.1. Overview of the mechanism

This update mechanism relies on two historical matrices: the historical search position matrix, `search_history`, and the fitness matrix, `fitness_history`. The `search_history` is a three-dimensional matrix of size (N, Max_iter, dim) that represents the search positions of N individuals in a dim-dimensional problem across Max_iter iterations. In contrast, `fitness_history` is a two-dimensional matrix of size (N, Max_iter) that records the fitness values of the corresponding individuals at each iteration. The update mechanism comprises the following steps:

1. **Iteration Constraint:** The mechanism is activated only when the number of iterations $i > n$, ensuring that at least n generations of historical data are available for trend estimation.

2. **Trend Point Computation:** A straight line is constructed by connecting the search positions from the last iteration, `search_history`$(j, i-1, :)$, with those from the previous iteration, `search_history`$(j, i-2, :)$. The K nearest points to this line are identified from the historical data, with the selection limited to the first n most recent searches to minimize computation.

3. **Vector Generation:** A vector $\mathbf{V}$ is computed to point toward the historical location, and a random vector $\mathbf{V}'$ is generated using an adaptive covariance mechanism.

4. **Fitness Comparison and Update:** The fitness values of the K nearest points are compared with those of both the current and historical positions to determine the direction for updating the algorithm.

2.2. Distance calculation

The parametric equation of the line is constructed, and the nearest point is identified by calculating the distance. Let P represent a point in the search history, and the distance d from P to the line can be computed using the following equation:

$$d(P) = \frac{\|(\text{search\_history}(j, i-1, :) - P) \times (\text{search\_history}(j, i-2, :) - \text{search\_history}(j, i-1, :))\|}{\|\text{search\_history}(j, i-2, :) - \text{search\_history}(j, i-1, :)\|} \tag{1}$$

In this equation, $\times$ denotes the cross product, and $\|\cdot\|$ represents the magnitude of the vector. The distance from each point P to the line is calculated by iterating through all search history positions, and the K points with the smallest distances are selected.

2.3. Adaptive covariance mechanism

In high-dimensional spaces, randomly generated offset vectors can significantly impact search efficiency, as the complexity of such spaces increases the challenge of balancing exploration and exploitation. If the direction and magnitude of these offsets are not adaptively adjusted, the algorithm may suffer from inefficiency, either due to excessive exploration or premature convergence to local optima. To address this, the proposed mechanism incorporates an adaptive covariance-based approach that adjusts the offset vector $\mathbf{V}'$ according to the current population distribution. This adaptation helps to improve both search efficiency and accuracy. The working principle and implementation steps of this mechanism are detailed below:

First, the vector $\mathbf{V}$, which points in the direction of the historical position, is computed as follows:

$$\mathbf{V} = \text{search\_history}(j, i-2, :) - \text{search\_history}(j, i-1, :) \tag{2}$$

An adaptive covariance mechanism is used to generate a high-dimensional vector $\mathbf{V}'$, where $Z \sim \mathcal{N}(0, \Sigma)$ is a Gaussian random vector with zero mean, and α and β are weighting parameters that control the balance between the historical trend and stochastic exploration. The covariance matrix Σ describes the correlation between dimensions in multidimensional data, such as population locations. In population optimization algorithms, individual positions are often distributed across different dimensions, and these dimensions may exhibit dependencies. Calculating the covariance matrix from the population positions can capture these correlations and fluctuations, enabling a more focused search.

The covariance matrix Σ is computed based on the current positions $\mathbf{X}$ of the population:

$$\Sigma = \frac{1}{N} \sum_{j=1}^{N} \left((\mathbf{X}(j, :) - \mu)(\mathbf{X}(j, :) - \mu)^T \right) \tag{3}$$

$$\mathbf{X}(j, :) = \text{search\_history}(j, i-1, :) \tag{4}$$

where μ represents the population center (the mean position across all dimensions), and N is the number of individuals in the population.

Once the covariance matrix Σ is calculated, a random vector $Z \sim \mathcal{N}(0, \Sigma)$ is generated, whose direction and magnitude are influenced by the population's distribution. Using $\mathbf{V}$ as a directional reference, the random vector Z is guided along the direction of $\mathbf{V}$, resulting in the offset vector $\mathbf{V}'$.

$$\mathbf{V}' = \alpha \mathbf{V} + \beta Z \tag{5}$$

Here, α and β are the weights used to balance the historical trend and stochastic exploration:

$$\alpha = \frac{i}{\text{Max\_iter}}, \quad \beta = 1 - \alpha \tag{6}$$

To ensure that the magnitude of $\mathbf{V}'$ is equal to $\mathbf{V}$, $\mathbf{V}'$ is normalized as follows:

$$\mathbf{V}' = \frac{\mathbf{V}'}{\|\mathbf{V}'\|} \times \|\mathbf{V}\| \tag{7}$$

2.4. Adaptation Comparison and Update

The fitness value corresponding to the ith point among the K nearest points is f_i, and it is compared with the fitness values of $\texttt{search\_history}(j, i-1, :)$ and $\texttt{search\_history}(j, i-2, :)$. The comparison is performed as follows:

- If $f_i > \max(\texttt{fitness\_history}(j, i-1, :), \texttt{fitness\_history}(j, i-2, :))$, then set $F_i = 1$.
- Else if $f_i > \min(\texttt{fitness\_history}(j, i-1, :), \texttt{fitness\_history}(j, i-2, :))$ and $\texttt{fitness\_history}(j, i-1, :) < \texttt{fitness\_history}(j, i-2, :)$, then set $F_i = 0.4$.
- Else if $f_i > \min(\texttt{fitness\_history}(j, i-1, :), \text{fitness\_history}(j, i-2, :))$ and $f_i < \max(\texttt{fitness\_history}(j, i-1, :), \texttt{fitness\_history}(j, i-2, :))$, then set $F_i = 0$.
- Else if $f_i < \min(\texttt{fitness\_history}(j, i-1, :), \texttt{fitness\_history}(j, i-2, :))$, then set $F_i = -1$.

If $\sum_{i=1}^{K} F_i < 0$, this indicates a high probability that a better fitness value lies between the two points. In this case, the update strategy is:

$$\mathbf{X}_{\text{new}}(j, :) = \mathbf{X}_{\text{new}}(j, :) + \mathbf{Vec} \tag{8}$$

where

$$\mathbf{Vec} = \mathbf{V}' \times \frac{1 - i/\text{Max\_iter}}{10} \tag{9}$$

This adjustment offsets $\mathbf{X}_{\text{new}}(j, :)$ in the direction where a lower fitness value is likely to exist. Otherwise, if $\sum_{i=1}^{K} F_i > 0$, suggesting that a poorer fitness value exists between the two points, the update strategy is:

$$\mathbf{X}_{\text{new}}(j, :) = \mathbf{X}_{\text{new}}(j, :) - \mathbf{Vec} \tag{10}$$

This causes the point $\mathbf{X}_{\text{new}}(j, :)$ to shift away from the current trend, ensuring that the individual's position moves in a direction where better fitness values are more likely to be found.

As illustrated in Fig. 1(a), points A (fitness_history$(j, i-2, :)$) and B (fitness_history$(j, i-1, :)$) are used for trend perception, while points C, D, E, and F represent historical data. With $K = 3$, C, D, and E are identified as the nearest points, and the proximity count $\sum_{i=1}^{K} F_i = 0.9$ is computed. Fig. 1(b)is a 2D projection of Fig. 1(a), where the position of each point remains consistent. Consequently, the green vector in the figure, representing the main perception direction, aligns with the $\overrightarrow{AB}$. The yellow vector denotes a random vector generated based on covariance, and the grey vector indicates the final output of the TAM.

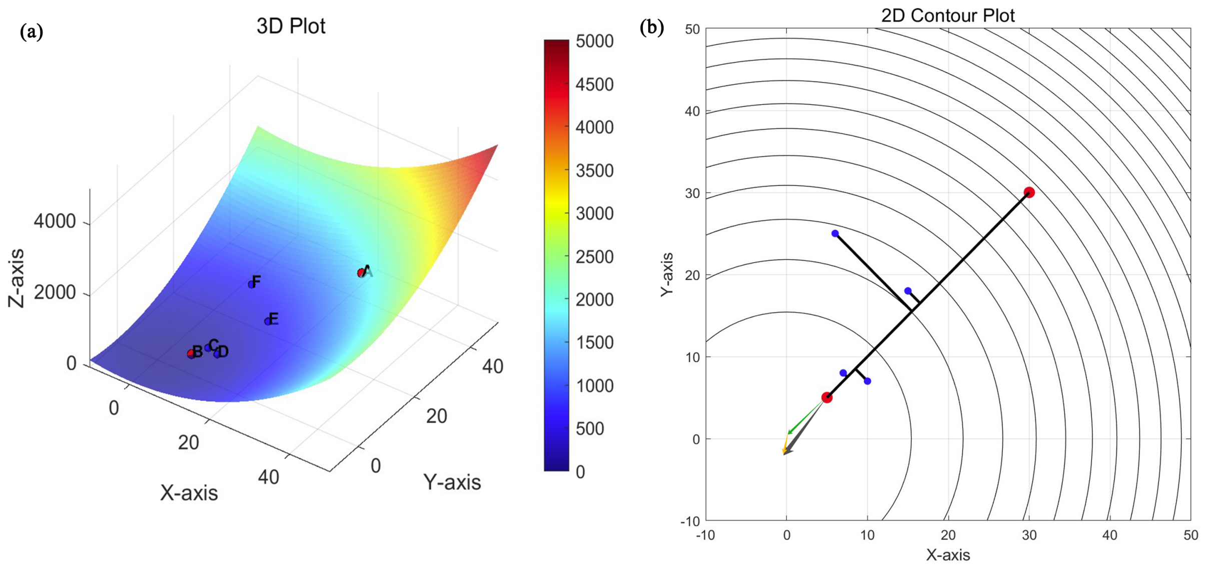

Figure 1: TAM Application Example: (a) 3D plot; (b) 2D contour plot

2.5. Pseudo-code of the Trend-Aware Mechanism

The pseudo-code of the TAM is provided in Algorithm 1, with its workflow illustrated in Fig. 2. Here, i denotes the current iteration index, and Max_iter represents the maximum iteration limit. As observed from both the pseudo-code and the flowchart, the parameter K plays a key role in determining the selection of nearest points for TAM, with larger K values expanding the neighborhood size and potentially improving exploration capabilities. Meanwhile, the parameters α and β control the balance between trend weighting and randomness, where a higher α emphasizes historical search trends.

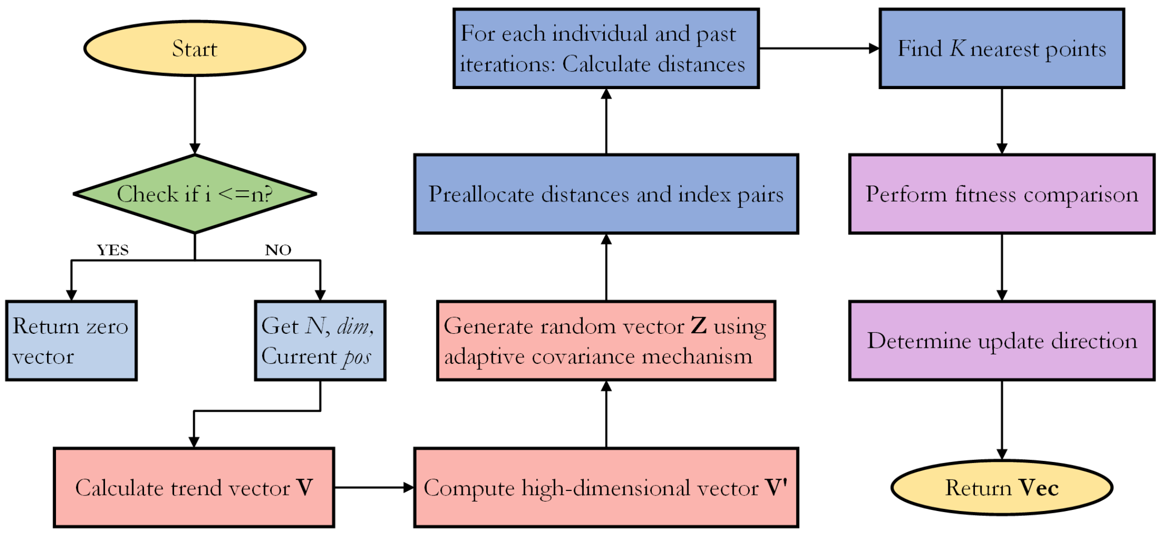
Start
Check if i <=n?
YES
NO
Return zero vector
Get N, dim, Current pos
Calculate trend vector V
Compute high-dimensional vector V'
Generate random vector Z using adaptive covariance mechanism
Preallocate distances and index pairs
For each individual and past iterations: Calculate distances
Find K nearest points
Perform fitness comparison
Determine update direction
Return Vec

Figure 2: Flowchart of TAM

Algorithm 1: Trend-Aware Mechanism (TAM)

```
1  Input: search_history, fitness_history, $Max\_iter$, $i$, $j$, $K$
   Output: Trend-aware vector and direction
2  if $i \leq n$ then
3      Return zero vector of size $1 \times dim$;
4  end
5  Get dimensions;
6  $[N, Max\_iter, dim]$ = SIZE(search_history);
7  Current and previous positions;
8  $pos\_curr$ = search_history$[j, i-1, :]$;
9  $pos\_prev$ = search_history$[j, i-2, :]$;
10 Compute trend vector $V$ (Eq. (2));
11 $V = pos\_prev - pos\_curr$;
12 Generate random vector $Z$ using covariance (Eq. (3)-Eq. (7));
13 $X$ = search_history$[:, i-1, :]$;
14 $\Sigma = \mathrm{COV}(X)$;
15 $Z = \mathrm{mvnrnd}(zeros(dim, 1), \Sigma)$;
16 Compute $V'$ (Eq. (5));
17 $\alpha = \frac{i}{Max\_iter}$;
18 $V' = \alpha \cdot V + \beta \cdot Z'$;
19 Preallocate for distances and index pairs;
20 $dists$ = zero vector of size $N \times (i-1)$;
21 $index\_pairs$ = zero matrix of size $N \times (i-1, 2)$;
22 $count = 0$;
23 Calculate distances;
24 for $p = 1$ to $N$ do
25     for $it = i - n$ to $i - 1$ do
26         $P$ = search_history$[p, it, :]$;
27         $d$ = CALCULATE_DISTANCE$(P, pos\_curr, pos\_prev)$;
28         $count = count + 1$;
29         $dists[count] = d$;
30         $index\_pairs[count, :] = [p, it]$;
31     end
32 end
33 Find $K$ nearest points (Eq. (8)-Eq. (10));
34 $nearest\_indices$ = FIND_K_NEAREST$(dists, index\_pairs, K)$;
35 Update strategy;
36 $F$ = EVALUATE_FITNESS$(nearest\_indices, fitness\_history, j, i)$;
37 Vec = DETERMINE_UPDATE_DIRECTION$(F, V', i, Max\_iter)$;
38 Return Vec with direction
```

2.6. Computational Complexity of TAM

The TAM procedure introduces three extra computational steps per iteration in addition to the baseline optimizer's work: (i) trend-line construction, (ii) K-nearest-neighbour evaluation, and (iii) covariance-based perturbation. Deriving the primary movement trend line from the two most recent positions of each individual costs $\mathcal{O}(ND)$ time, where N is the population size and D the search-space dimension. Computing distances from the current individual to N historical points likewise costs $\mathcal{O}(ND)$, after which selecting the K closest neighbours requires at most $\mathcal{O}(N \log N)$ time (naïve heap selection). The adaptive covariance mechanism then forms a $D \times D$ sample covariance matrix from those K neighbours, incurring $\mathcal{O}(KD^2)$, and draws a multivariate normal sample; a Cholesky factorisation gives a worst-case cost of $\mathcal{O}(D^3)$. Combining these terms, one TAM-augmented iteration executes in

$$\mathcal{O}\big(ND + N \log N + KD^2 + D^3\big).$$

A standard iteration of the underlying algorithm (without TAM) is typically $\mathcal{O}(ND)$, spent on updating the N solutions and evaluating their fitness. Thus, TAM adds an asymptotic overhead of

$$\mathcal{O}\big(N\log N + KD^2 + D^3\big)$$

per generation. For moderate dimensions and small K this extra cost is usually negligible, but the $\mathcal{O}(D^3)$ component can dominate as D becomes very large.

3. Parameters sensitivity analysis

To examine the parameter sensitivity of the Trend-Aware Mechanism (TAM), we selected the Particle Swarm Optimization (PSO) algorithm [38] as a representative framework for experimentation. This section investigates the impact of two critical parameters in TAM: the number of proximity points (K) and the memory rounds (n). The K parameter, analogous to that in the K-Nearest Neighbors (KNN) algorithm, affects both decision-making complexity and algorithmic efficiency. TAM aims to balance capturing local relationships between data points and identifying overarching trends, making the selection of K crucial; in this study, n is fixed at 3 by default. As n increases, decision-making tends to become more cautious and precise, albeit at the cost of reduced operational efficiency, underscoring the need to optimize its value. Additionally, the assignment of F_i under conditions such as $f_i > \min(\texttt{fitness\_history}(j,i-1,:),\texttt{fitness\_history}(j,i-2,:))$ and $\texttt{fitness\_history}(j,i-1,:) < \texttt{fitness\_history}(j,i-2,:)$ (defaulted to 0.4 in this study) offers intriguing insights into sensitivity analysis. This parameter is pivotal in balancing the resolution of single-modal and multi-modal functions, a topic to be further explored in future research.

To validate our experiments, we employed an improved PSO algorithm with the following parameters: $V_{\max} = 2$, $w_{\max} = 0.9$, $w_{\min} = 0.2$, and $c_1 = c_2 = 2$. The formulation of the TAM-PSO model is as follows:

$$\text{vel}_{\text{new}}(i,j) = w \times \text{vel}(i,j) + c_1 \times \text{rand} \times (\text{pBest}(i,j) - \text{pos}(i,j)) + c_2 \times \text{rand} \times (\text{gBest}(j) - \text{pos}(i,j)) \tag{11}$$

$$\text{pos}_{\text{new}}(i,:) = \text{pos}(i,:) + \text{vel}_{\text{new}}(i,j) \pm \mathbf{Vec} \tag{12}$$

Here, $\text{vel}_{\text{new}}(i,j)$ presents the updated velocity of particle i in dimension j, with w representing the inertia weight. The term $\text{pBest}(i,j)$ refers to the optimal position previously identified by particle i, while $\text{gBest}(j)$ indicates the global best position. $\text{pos}(i,j)$ corresponds to the current location of particle i, and **Vec** introduces the adjustment derived from the TAM.

The PSO-based models are evaluated with a population size of 50 on the 20-dimensional CEC2022 test functions. The algorithm runs for 4000 iterations, adhering to the standard practice where the maximum number of function evaluations is set to $20,000 \times \text{dim}$.To ensure the reproducibility and reliability of the results, each experiment was conducted independently 30 times.

3.1. The K value

In this subsection, we examine the sensitivity of the parameter K within the mechanism model. To illustrate the convergence patterns and stability across different K values, we present box-and-whisker plots in Fig. 3. Additionally, Table 1 provides a comprehensive performance analysis for various K values across the test functions, with rankings based on mean performance and overall evaluation summarized in Table 2.

The results demonstrate that the TAM modification enhances the performance of the PSO in test functions F_3, F_4, F_5, F_7, F_8, F_{10}, and F_{12}. In contrast, TAM has minimal impact on F_2, and F_9, and slightly negatively affects F_1, F_6, and F_{11}. Overall, TAM demonstrates significant potential for optimizing PSO performance. The analysis also reveals that different K values yield varied impacts on algorithm outcomes: the best performance, balancing stability and convergence, is achieved at $K = 10$, with $K = 5$ also showing good convergence. When $K = 3$, stability diminishes due to insufficient proximity points, as observed in F_6 and F_{11}. Conversely, larger K values can introduce distant points that distort trend estimation, seen in F_3 and F_{11}. Smaller K values are preferable for estimating exploratory phases involving larger inter-point distances, while larger K values enhance performance in more refined, densely sampled phases. Designing adaptive strategies for K is thus essential for further improving TAM's effectiveness in future studies.

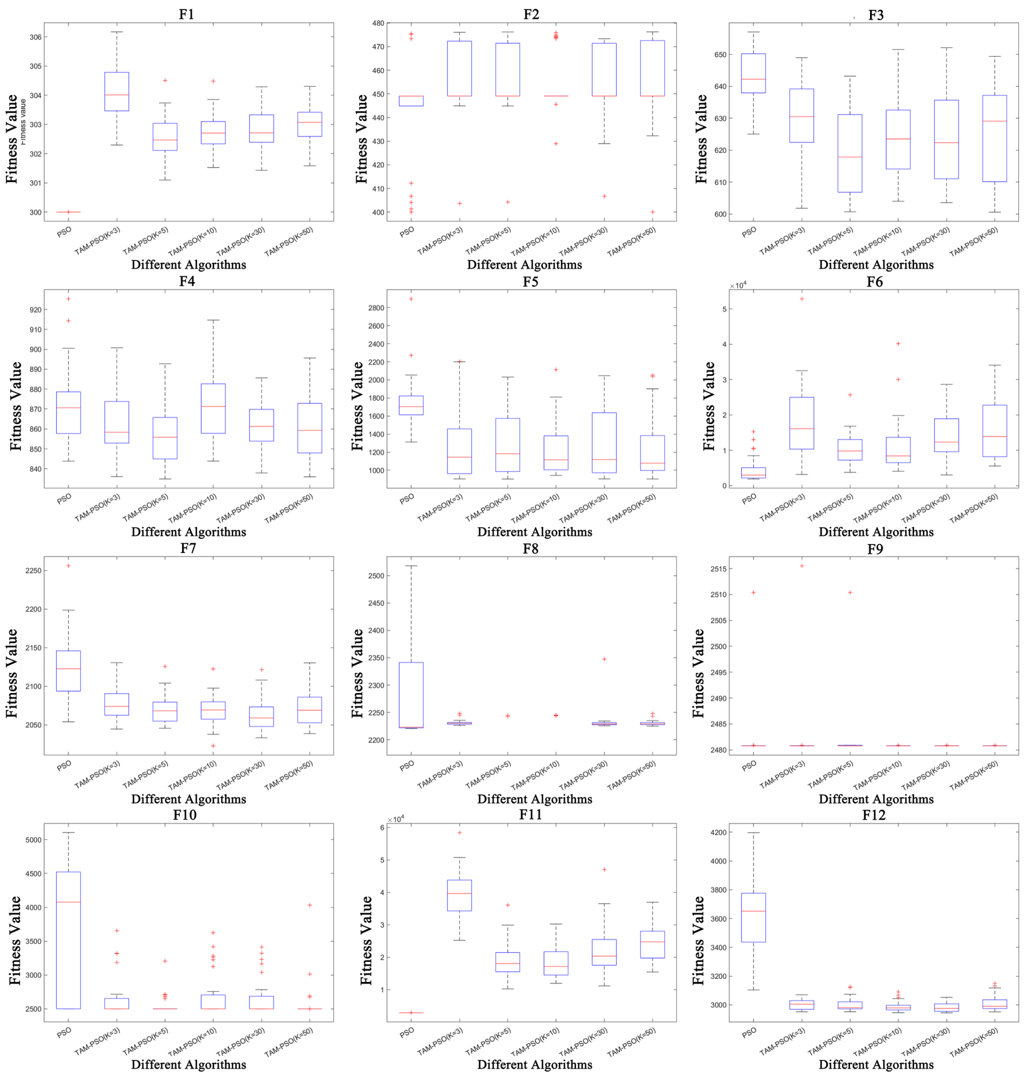

Figure 3: Comparison of K-values

3.2. The n value

In this subsection, we discuss the sensitivity of the parameter n. Setting n too high, representing the number of rounds to be memorized, can lead to excessive time spent searching for the shortest point, while setting n too low may result in the loss of potentially better points. Therefore, determining an appropriate value for n is essential. When $n = 1$, no memory is utilized, corresponding to the standard PSO version. The following experiments compare the standard PSO model with TAM-enhanced PSO variants for $n = 2, 3, \ldots, 5$ to analyze and identify the optimal value of n for future algorithm design.

Fig. 4 presents box plots for different algorithms on the CEC2022 test functions, while Table 3 shows the performance results using evaluation metrics, and Table 4 summarizes the overall rankings. The results show that the optimal number of memory rounds is not maximized but reaches an ideal value at $n = 3$. Higher values of n can lead to repeated use of previously incorrect positions and significantly increase computational time. Conversely, smaller values of n may cause algorithmic instability, as observed in the results for functions $F3$ and $F5$. Although $n = 4$ shows better performance in functions $F3$, $F6$, $F11$, and $F12$, an overall evaluation demonstrates that $n = 3$ is the most effective parameter for TAM. Therefore, this value will be adopted in future algorithm design and comparative studies.

Table 1
Performance of different K-values in CEC2022

Functions	Items	PSO	TAM-PSO(K=3)	TAM-PSO(K=5)	TAM-PSO(K=10)	TAM-PSO(K=30)	TAM-PSO(K=50)
F1	Worst	**3.000E+02**	3.051E+02	3.051E+02	3.068E+02	3.045E+02	3.038E+02
	Best	**3.000E+02**	3.012E+02	3.015E+02	3.013E+02	3.015E+02	3.012E+02
	STD	**5.684E-14**	9.979E-01	8.062E-01	9.686E-01	7.482E-01	5.856E-01
	Median	**3.000E+02**	3.029E+02	3.029E+02	3.023E+02	3.027E+02	3.025E+02
	Mean	**3.000E+02**	3.031E+02	3.029E+02	3.025E+02	3.028E+02	3.026E+02
F2	Worst	**4.725E+02**	4.755E+02	4.744E+02	4.740E+02	4.753E+02	4.747E+02
	Best	**4.000E+02**	4.065E+02	4.323E+02	4.023E+02	4.449E+02	4.001E+02
	STD	2.192E+01	1.785E+01	1.240E+01	1.547E+01	**1.210E+01**	2.223E+01
	Median	**4.491E+02**	**4.491E+02**	**4.491E+02**	**4.491E+02**	**4.491E+02**	**4.491E+02**
	Mean	**4.457E+02**	4.557E+02	4.564E+02	4.549E+02	4.575E+02	4.504E+02
F3	Worst	6.600E+02	6.526E+02	**6.437E+02**	6.498E+02	6.519E+02	6.523E+02
	Best	6.287E+02	6.033E+02	6.020E+02	6.008E+02	**6.005E+02**	6.023E+02
	STD	**8.217E+00**	1.477E+01	1.238E+01	1.441E+01	1.552E+01	1.571E+01
	Median	6.398E+02	**6.227E+02**	6.249E+02	6.229E+02	6.283E+02	6.263E+02
	Mean	6.421E+02	6.235E+02	6.225E+02	**6.214E+02**	6.248E+02	6.248E+02
F4	Worst	9.254E+02	9.008E+02	9.156E+02	**8.897E+02**	8.977E+02	8.976E+02
	Best	8.438E+02	8.361E+02	**8.210E+02**	8.250E+02	8.339E+02	8.329E+02
	STD	1.920E+01	1.576E+01	2.071E+01	**1.554E+01**	1.465E+01	1.516E+01
	Median	8.706E+02	**8.583E+02**	8.657E+02	8.618E+02	8.628E+02	8.599E+02
	Mean	8.723E+02	8.630E+02	8.703E+02	**8.620E+02**	8.630E+02	8.599E+02
F5	Worst	2.897E+03	2.206E+03	2.035E+03	**1.793E+03**	1.934E+03	2.056E+03
	Best	1.313E+03	**9.036E+02**	9.108E+02	9.107E+02	**9.032E+02**	9.067E+02
	STD	2.949E+02	3.727E+02	**2.689E+02**	2.865E+02	2.972E+02	3.207E+02
	Median	1.703E+03	1.144E+03	1.058E+03	1.047E+03	1.121E+03	**1.009E+03**
	Mean	1.744E+03	1.242E+03	**1.193E+03**	1.160E+03	1.206E+03	1.180E+03
F6	Worst	**1.526E+04**	5.283E+04	2.341E+04	2.574E+04	2.918E+04	3.813E+04
	Best	**1.895E+03**	3.187E+03	3.375E+03	4.392E+03	2.857E+03	4.770E+03
	STD	**3.543E+03**	1.032E+04	6.106E+03	5.397E+03	7.186E+03	8.430E+03
	Median	**3.001E+03**	1.611E+04	1.100E+04	9.983E+03	1.102E+04	1.282E+04
	Mean	**4.627E+03**	1.806E+04	1.145E+04	1.116E+04	1.340E+04	1.501E+04
F7	Worst	2.200E+03	2.121E+03	2.107E+03	2.105E+03	**2.102E+03**	2.123E+03
	Best	2.064E+03	2.035E+03	2.034E+03	2.030E+03	2.029E+03	**2.027E+03**
	STD	3.143E+01	2.475E+01	**1.789E+01**	1.936E+01	2.243E+01	2.536E+01
	Median	2.103E+03	2.071E+03	**2.064E+03**	2.067E+03	2.073E+03	2.075E+03
	Mean	2.110E+03	2.076E+03	2.066E+03	**2.065E+03**	2.069E+03	2.074E+03
F8	Worst	2.455E+03	2.247E+03	**2.244E+03**	**2.244E+03**	2.245E+03	**2.244E+03**
	Best	**2.221E+03**	2.226E+03	2.223E+03	2.224E+03	2.225E+03	2.225E+03
	STD	7.107E+01	4.111E+00	5.126E+00	3.482E+00	3.872E+00	**3.357E+00**
	Median	2.238E+03	2.230E+03	**2.229E+03**	**2.229E+03**	**2.229E+03**	**2.229E+03**
	Mean	2.286E+03	2.231E+03	2.230E+03	**2.229E+03**	2.230E+03	**2.229E+03**
F9	Worst	2.510E+03	2.510E+03	**2.481E+03**	**2.481E+03**	2.510E+03	2.510E+03
	Best	**2.481E+03**	**2.481E+03**	**2.481E+03**	**2.481E+03**	**2.481E+03**	**2.481E+03**
	STD	5.402E+00	5.392E+00	5.362E-02	**4.167E-02**	5.479E+00	7.491E+00
	Median	**2.481E+03**	**2.481E+03**	**2.481E+03**	**2.481E+03**	**2.481E+03**	**2.481E+03**
	Mean	2.482E+03	2.482E+03	**2.481E+03**	**2.481E+03**	2.482E+03	2.483E+03
F10	Worst	5.104E+03	3.656E+03	**3.151E+03**	4.041E+03	3.699E+03	3.780E+03
	Best	**2.500E+03**	2.501E+03	**2.500E+03**	**2.500E+03**	**2.500E+03**	**2.500E+03**
	STD	9.633E+02	3.179E+02	**1.562E+02**	4.884E+02	3.385E+02	4.038E+02
	Median	3.919E+03	**2.501E+03**	**2.501E+03**	2.597E+03	**2.501E+03**	**2.501E+03**
	Mean	3.732E+03	2.664E+03	**2.584E+03**	2.901E+03	2.692E+03	2.707E+03
F11	Worst	**2.900E+03**	4.644E+04	4.802E+04	3.395E+04	3.634E+04	3.415E+04
	Best	**2.900E+03**	2.680E+04	1.151E+04	1.046E+04	1.048E+04	1.364E+04
	STD	**4.222E-13**	4.932E+03	7.219E+03	4.909E+03	6.226E+03	5.610E+03
	Median	**2.900E+03**	3.796E+04	1.834E+04	1.625E+04	1.900E+04	2.321E+04
	Mean	**2.900E+03**	3.709E+04	1.956E+04	1.766E+04	2.119E+04	2.256E+04
F12	Worst	4.195E+03	3.141E+03	3.189E+03	**3.082E+03**	3.091E+03	3.321E+03
	Best	3.129E+03	2.946E+03	2.952E+03	2.950E+03	2.952E+03	**2.943E+03**
	STD	2.243E+02	**3.987E+01**	6.581E+01	4.091E+01	4.540E+01	8.496E+01
	Median	3.486E+03	2.993E+03	2.984E+03	2.987E+03	**2.979E+03**	2.986E+03
	Mean	3.489E+03	**2.998E+03**	3.008E+03	**2.998E+03**	2.999E+03	3.010E+03

Table 2
Ranking of different K-values in CEC2022

Functions	PSO	TAM-PSO(K=3)	TAM-PSO(K=5)	TAM-PSO(K=10)	TAM-PSO(K=30)	TAM-PSO(K=50)
F1	1	6	5	2	4	3
F2	1	4	5	3	6	2
F3	6	3	2	1	4	5
F4	6	3	5	2	4	1
F5	6	5	3	1	4	2
F6	1	6	3	2	4	5
F7	6	5	2	1	3	4
F8	6	5	4	1	3	2
F9	3	4	2	1	5	6
F10	6	2	1	5	3	4
F11	1	6	3	2	4	5
F12	6	2	4	1	3	5
Average Rank	**4.083**	**4.250**	**3.250**	**1.833**	**3.917**	**3.667**
Final Ranking	**5**	**6**	**2**	**1**	**4**	**3**

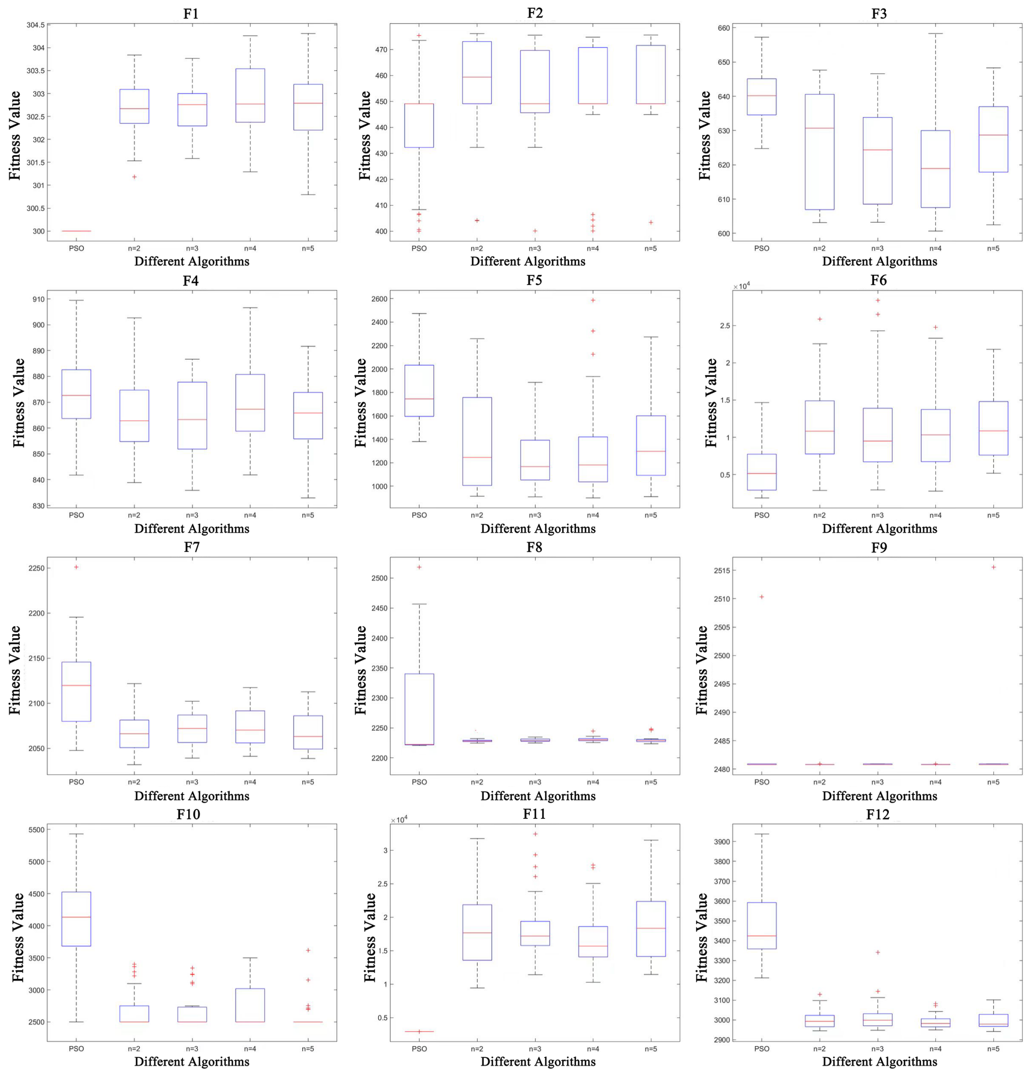

Figure 4: Comparison of n-values

Table 3
Performance of different n-values in CEC2022

Functions	Items	PSO	n=2	n=3	n=4	n=5
	Worst	**3.000E+02**	3.038E+02	3.038E+02	3.043E+02	3.043E+02
	Best	**3.000E+02**	3.012E+02	3.016E+02	3.013E+02	3.008E+02
F1	STD	**3.338E-14**	6.677E-01	5.287E-01	8.047E-01	7.606E-01
	Median	**3.000E+02**	3.027E+02	3.028E+02	3.028E+02	3.028E+02
	Mean	**3.000E+02**	3.026E+02	3.027E+02	3.029E+02	3.027E+02
	Worst	4.754E+02	4.761E+02	4.755E+02	**4.747E+02**	4.755E+02
	Best	**4.000E+02**	4.041E+02	4.001E+02	4.001E+02	4.034E+02
F2	STD	2.369E+01	2.041E+01	**1.567E+01**	2.198E+01	1.548E+01
	Median	**4.491E+02**	4.593E+02	**4.491E+02**	**4.491E+02**	**4.491E+02**
	Mean	**4.431E+02**	4.563E+02	4.535E+02	4.510E+02	4.565E+02
	Worst	6.572E+02	6.476E+02	**6.466E+02**	6.583E+02	6.483E+02
	Best	6.247E+02	6.031E+02	6.032E+02	**6.007E+02**	6.025E+02
F3	STD	**8.485E+00**	1.538E+01	1.451E+01	1.545E+01	1.320E+01
	Median	6.402E+02	6.307E+02	6.244E+02	**6.189E+02**	6.287E+02
	Mean	6.405E+02	6.260E+02	6.232E+02	**6.222E+02**	6.267E+02
	Worst	9.094E+02	9.027E+02	**8.867E+02**	9.066E+02	8.917E+02
	Best	8.418E+02	8.389E+02	8.359E+02	8.418E+02	**8.330E+02**
F4	STD	1.737E+01	1.492E+01	1.528E+01	1.692E+01	**1.394E+01**
	Median	8.726E+02	8.628E+02	8.633E+02	8.673E+02	**8.658E+02**
	Mean	8.735E+02	8.656E+02	**8.634E+02**	8.683E+02	8.646E+02
	Worst	2.473E+03	2.258E+03	**1.885E+03**	2.585E+03	2.273E+03
	Best	1.380E+03	9.144E+02	9.091E+02	**9.002E+02**	9.105E+02
F5	STD	2.905E+02	4.145E+02	**2.947E+02**	4.318E+02	3.746E+02
	Median	1.745E+03	1.245E+03	**1.167E+03**	1.181E+03	1.297E+03
	Mean	1.818E+03	1.372E+03	**1.258E+03**	1.317E+03	1.368E+03
	Worst	**1.466E+04**	2.583E+04	2.836E+04	2.476E+04	2.180E+04
	Best	**1.862E+03**	2.872E+03	2.934E+03	2.778E+03	5.173E+03
F6	STD	**3.556E+03**	5.969E+03	6.912E+03	5.527E+03	5.007E+03
	Median	**5.143E+03**	1.082E+04	9.496E+03	1.033E+04	1.086E+04
	Mean	**5.738E+03**	1.194E+04	1.151E+04	1.097E+04	1.177E+04
	Worst	2.251E+03	2.122E+03	**2.102E+03**	2.117E+03	2.113E+03
	Best	2.048E+03	2.032E+03	**2.039E+03**	2.041E+03	**2.039E+03**
F7	STD	4.575E+01	2.156E+01	**1.820E+01**	2.240E+01	2.394E+01
	Median	2.120E+03	**2.066E+03**	2.072E+03	2.070E+03	**2.063E+03**
	Mean	2.119E+03	**2.068E+03**	2.071E+03	2.075E+03	2.069E+03
	Worst	2.518E+03	**2.246E+03**	2.249E+03	**2.245E+03**	2.248E+03
	Best	**2.221E+03**	2.224E+03	2.225E+03	2.225E+03	2.223E+03
F8	STD	8.410E+01	**3.599E+00**	4.310E+00	**3.599E+00**	6.187E+00
	Median	**2.222E+03**	2.228E+03	2.229E+03	2.230E+03	2.229E+03
	Mean	2.276E+03	**2.229E+03**	**2.230E+03**	**2.231E+03**	**2.230E+03**
	Worst	2.510E+03	**2.481E+03**	**2.481E+03**	**2.481E+03**	2.516E+03
	Best	**2.481E+03**	**2.481E+03**	**2.481E+03**	**2.481E+03**	**2.481E+03**
F9	STD	5.389E+00	**4.489E-02**	5.485E-02	**4.849E-02**	6.341E+00
	Median	**2.481E+03**	**2.481E+03**	**2.481E+03**	**2.481E+03**	**2.481E+03**
	Mean	2.482E+03	**2.481E+03**	**2.481E+03**	**2.481E+03**	2.482E+03
	Worst	5.431E+03	3.397E+03	**3.341E+03**	3.499E+03	3.617E+03
	Best	**2.501E+03**	**2.500E+03**	**2.500E+03**	**2.500E+03**	**2.500E+03**
F10	STD	8.272E+02	3.072E+02	2.786E+02	3.431E+02	**2.376E+02**
	Median	4.133E+03	**2.501E+03**	**2.501E+03**	**2.501E+03**	**2.501E+03**
	Mean	3.997E+03	2.680E+03	2.667E+03	2.743E+03	**2.595E+03**
	Worst	**2.900E+03**	3.175E+04	3.242E+04	2.779E+04	3.152E+04
	Best	**2.900E+03**	9.427E+03	1.140E+04	1.027E+04	1.144E+04
F11	STD	**4.137E-13**	5.510E+03	4.913E+03	4.409E+03	5.147E+03
	Median	**2.900E+03**	1.767E+04	1.718E+04	1.567E+04	1.833E+04
	Mean	**2.900E+03**	1.775E+04	1.843E+04	1.685E+04	1.891E+04
	Worst	3.938E+03	3.128E+03	3.341E+03	**3.083E+03**	3.102E+03
	Best	3.212E+03	**2.945E+03**	2.948E+03	2.950E+03	2.942E+03
F12	STD	1.779E+02	**4.506E+01**	7.668E+01	**3.456E+01**	4.501E+01
	Median	3.424E+03	2.993E+03	2.999E+03	**2.982E+03**	**2.979E+03**
	Mean	3.475E+03	3.001E+03	3.017E+03	**2.990E+03**	2.998E+03

Table 4
Ranking of different n-values in CEC2022

Function	PSO	n=2	n=3	n=4	n=5
F1	1	2	3	5	4
F2	1	4	3	2	5
F3	5	3	2	1	4
F4	5	3	1	4	2
F5	5	4	1	2	3
F6	1	5	3	2	4
F7	5	1	3	4	2
F8	5	1	2	4	3
F9	4	1	3	2	5
F10	5	3	2	4	1
F11	1	3	4	2	5
F12	5	3	4	1	2
Average Rank	**3.583**	**2.750**	**2.583**	**2.750**	**3.333**
Final Ranking	**5**	**2**	**1**	**2**	**4**

4. Comparison of algorithms

In this section, we integrate Trend-Aware Mechanism (TAM) into four high-performance, widely recognized algorithms to demonstrate its feasibility and broad applicability. The selected algorithms are PSO [38], SHADE [34], JaDE [33], and CMA-ES [39].

4.1. Design of improved algorithms

To enable meaningful comparisons, we provide a detailed explanation of the improvements introduced by TAM and the parameter settings for each algorithm in the following sections.

PSO: The design of the PSO, as well as the parameter control, are consistent with Section 3.1.

SHADE: The TAM is integrated into the mutation step of the TAM-SHADE algorithm to enhance its search capabilities by considering historical information from previous generations. Specifically, TAM introduces an additional term to the differential mutation operation, leveraging past `search_history` and `fitness_history` to guide the search process. In the standard SHADE algorithm, the mutation step for an individual x_i is defined as:

$$v_i = x_i + F \cdot (x_{\text{best}} - x_i) + F \cdot (x_{r1} - x_{r2}), \tag{13}$$

where x_{best} is a randomly selected best solution from the top p fraction of the population, $r1$ and $r2$ are indices of distinct, randomly selected individuals, and F is the mutation factor. The parameter settings are as follows:

- **Mutation factor (F):** Adaptively adjusted over generations, with an initial value of 0.5.
- **Crossover probability (CR):** Adaptively adjusted over generations, with an initial value of 0.5.
- ***p*-value:** The proportion for the "p-best" strategy is set to 0.04.

In TAM-SHADE, this mutation equation is modified to include a trend-aware component, TAM, as follows:

$$v_i = x_i + F \cdot (x_{\text{best}} - x_i) + F \cdot (x_{r1} - x_{r2}) \pm \text{Vec}, \tag{14}$$

where `Vec` represents the trend-aware adjustment term.

JaDE: builds upon the foundational Differential Evolution (DE) algorithm. It incorporates an adaptive control mechanism for its mutation factor (F) and crossover probability (CR), allowing them to evolve over generations. The algorithm uses a "p-best" strategy, where individuals are biased towards the best solutions in the current population to balance exploration and exploitation. The integration of TAM follows the same approach as in the SHADE algorithm. The parameter settings are as follows:

- **Mutation factor (F):** Adaptively adjusted over generations, with an initial value of 0.5.
- **Crossover probability (CR):** Adaptively adjusted over generations, with an initial value of 0.5.
- ***p*-value:** The "p-best" strategy proportion is set to 0.05.

CMA-ES: CMA-ES is a cutting-edge technique for solving continuous optimization challenges. To improve its performance, we integrate a historical trend-awareness component, enabling the algorithm to adjust its search direction based on information from previous generations. This enhancement guides the mutation process more effectively. The standard mutation process in CMA-ES generates new candidate solutions by sampling from a multivariate normal distribution, with the evolving covariance matrix shaping the distribution. The mutation step for an individual x_i is given by:

$$x_i = \mu + \sigma \cdot z_i, \tag{15}$$

The μ denotes the mean of the population (serving as the current best estimate of the solution), σ represents the step size (or standard deviation), which controls the dispersion of candidate solutions, and $z_i \sim \mathcal{N}(0, C)$ is a sample drawn from a normal distribution with covariance matrix C, which is updated adaptively. The covariance matrix C is adjusted using the weighted sum of the outer products of the deviations of the population members from the mean, thereby capturing the correlations between variables and steering the search towards the most promising directions.

To integrate the TAM into CMA-ES, we modify the mutation step by introducing an additional term that incorporates information from past generations. The new term, referred to as `Vec`, adjusts the mutation step based on the historical trend of the population. The modified mutation step with TAM becomes:

$$x_i' = \mu + \sigma \cdot z_i \pm \text{Vec}, \tag{16}$$

where `Vec` is the trend-aware component, calculated using the `search_history` and `fitness_history` of previous generations. This term directs the algorithm towards areas of the search space that have historically shown improvements, potentially accelerating convergence. In addition, the algorithm utilizes the population size to control the number of iterations consistently, replacing the original approach, which was based on dimensional computations. Furthermore, the improved algorithm incorporates boundary constraints to prevent solutions from exceeding the defined search space.

4.2. Comparison of algorithms

Fig. 5 illustrates the performance of the eight algorithms on the CEC2022, with a detailed set of evaluation metrics depicted in Table 5. It is evident that TAM effectively enhances not only the PSO algorithm but also SHADE, JaDE, and CMA-ES. The improvements brought by TAM are primarily reflected in two aspects: the stability and convergence of the algorithms. The models are evaluated with a population size of 50 on the 20-dimensional CEC2022 test functions. The algorithm runs for 4000 iterations, adhering to the standard practice where the maximum number of function evaluations is set to $20,000 \times$ dim. To ensure the reproducibility and reliability of the results, each experiment was conducted independently 30 times.

For instance, in F2, F6, F8, F9, and F10, TAM significantly enhances the stability of the algorithms, while in F2, F3, F4, F5, F7, and F12, the convergence of the TAM-enhanced algorithms is notably improved. These findings confirm the strong optimization potential of TAM.

Additionally, the experimental results indicate that TAM provides notable improvements on multimodal high-dimensional functions, whereas its performance on unimodal functions is comparatively limited. This discrepancy arises naturally from TAM's design philosophy, which primarily focuses on leveraging historical position information to prevent algorithms from becoming trapped in local optima. Specifically, at the start of optimization, the trend vector $\mathbf{v}_i^{(t)}$ is computed from relatively scattered historical positions; hence, its magnitude is initially large and directionally diverse, promoting **exploration** of the search space. As optimization progresses and the population converges, the positions of individuals become increasingly correlated, automatically causing a reduction in $\|\mathbf{v}_i^{(t)}\|$. Consequently, the update step gradually shortens and becomes biased towards the current best solution, transitioning the search focus to **exploitation**. Unlike classical PSO—where exploration-exploitation balance is controlled explicitly through inertia and social coefficients—or SHADE, which relies on adaptive parameter memories updated only after successful trials, TAM inherently modulates the exploration-exploitation trade-off in real-time via the evolving geometry of trajectories.

However, further refinement is needed to enhance TAM's effectiveness on unimodal functions, as discussed in detail in Section 2.4.

Wilcoxon signed-rank comparisons between each base algorithm and its TAM variant indicate statistically significant improvements ($p < 0.05$) on the majority of benchmark functions, consistent with the ranking in Table 5, and the Friedman test corroborates this ranking. However, these results underscore the potential of TAM for solving complex multimodal high-dimensional problems in engineering design, feature selection, and photovoltaic modeling applications. The relatively modest performance gains of TAM in enhancing SHADE are likely due to SHADE's inherent use of historical information, which reduces TAM's effectiveness in further improving an algorithm already designed to utilize such information.

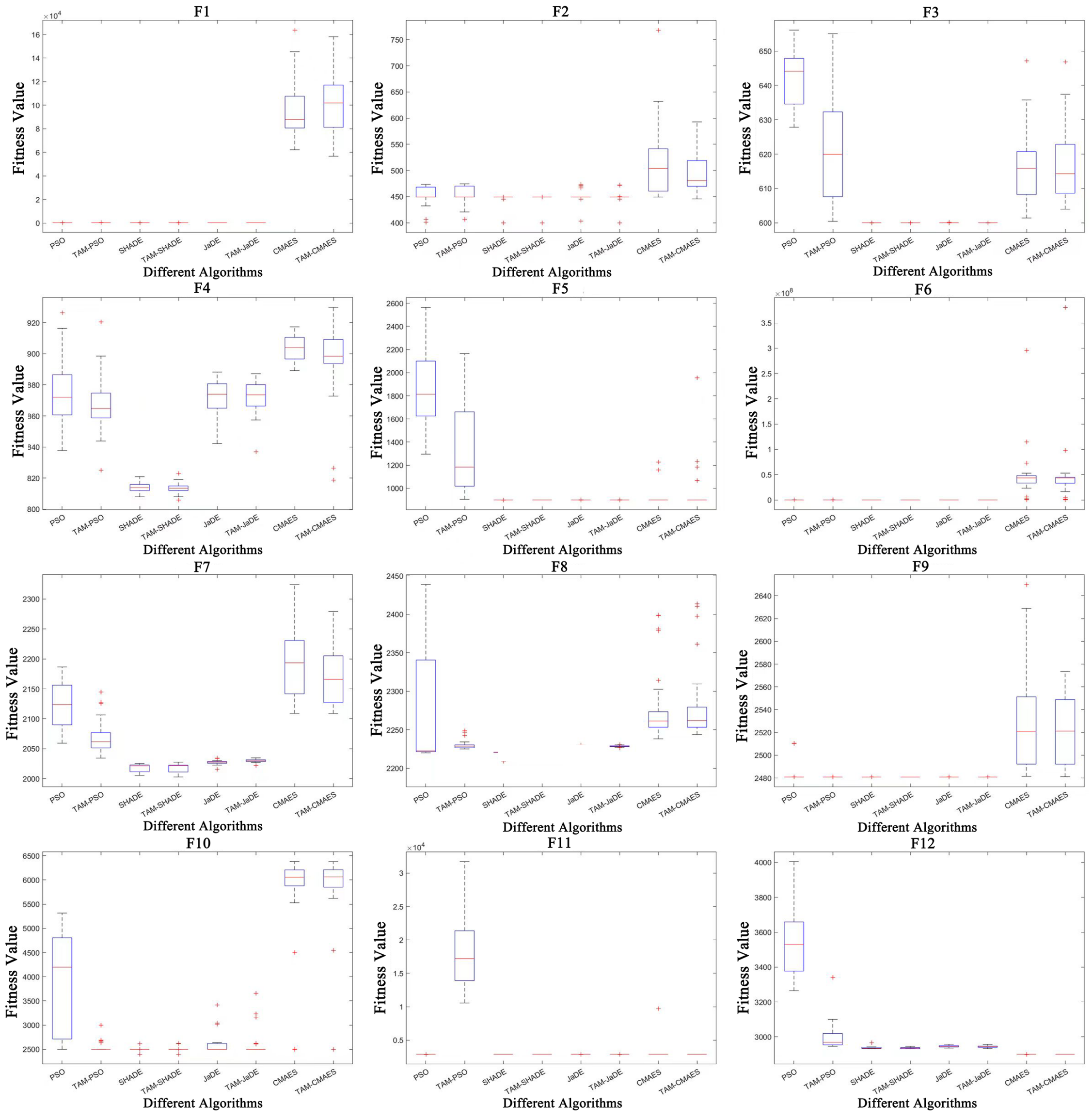

Figure 5: Comparison of different algorithms

Table 5
Performance of different algorithms in CEC2022

Functions	Items	PSO	TAM-PSO	SHADE	TAM-SHADE	JaDE	TAM-JaDE	CMAES	TAM-CMAES
	Worst	**3.000E+02**	3.051E+02	**3.000E+02**	**3.000E+02**	**3.000E+02**	**3.000E+02**	1.638E+05	1.581E+05
	Best	**3.000E+02**	3.018E+02	**3.000E+02**	**3.000E+02**	**3.000E+02**	**3.000E+02**	6.214E+04	5.667E+04
F1	STD	2.586E-14	7.448E-01	1.493E-14	1.338E-14	**0.000E+00**	**0.000E+00**	2.572E+04	2.300E+04
	Median	**3.000E+02**	3.027E+02	**3.000E+02**	**3.000E+02**	**3.000E+02**	**3.000E+02**	8.788E+04	1.018E+05
	Mean	**3.000E+02**	3.028E+02	**3.000E+02**	**3.000E+02**	**3.000E+02**	**3.000E+02**	9.629E+04	1.005E+05
	Worst	4.734E+02	4.745E+02	**4.491E+02**	**4.491E+02**	4.735E+02	4.723E+02	7.683E+02	5.929E+02
	Best	**4.000E+02**	4.067E+02	**4.000E+02**	**4.000E+02**	4.033E+02	**4.000E+02**	4.491E+02	4.456E+02
F2	STD	1.711E+01	1.620E+01	1.242E+01	**8.962E+00**	1.173E+01	1.412E+01	6.729E+01	3.871E+01
	Median	**4.491E+02**	4.593E+02	**4.491E+02**	**4.491E+02**	**4.491E+02**	**4.491E+02**	5.041E+02	4.805E+02
	Mean	4.515E+02	4.533E+02	4.455E+02	**4.454E+02**	4.500E+02	4.471E+02	5.123E+02	4.947E+02
	Worst	6.561E+02	6.551E+02	**6.000E+02**	**6.000E+02**	6.002E+02	**6.000E+02**	6.472E+02	6.469E+02
	Best	6.278E+02	6.004E+02	**6.000E+02**	**6.000E+02**	**6.000E+02**	**6.000E+02**	6.014E+02	6.040E+02
F3	STD	7.445E+00	1.496E+01	**1.097E-13**	**1.097E-13**	3.442E-02	4.370E-06	1.101E+01	1.052E+01
	Median	6.441E+02	6.199E+02	**6.000E+02**	**6.000E+02**	**6.000E+02**	**6.000E+02**	6.159E+02	6.143E+02
	Mean	6.423E+02	6.204E+02	**6.000E+02**	**6.000E+02**	**6.000E+02**	**6.000E+02**	6.176E+02	6.174E+02
	Worst	9.264E+02	9.205E+02	**8.209E+02**	8.229E+02	8.882E+02	8.872E+02	9.173E+02	9.300E+02
	Best	8.378E+02	8.250E+02	8.080E+02	**8.060E+02**	8.422E+02	8.369E+02	8.891E+02	8.187E+02
F4	STD	2.100E+01	1.750E+01	**3.334E+00**	3.415E+00	1.060E+01	9.883E+00	8.286E+00	2.331E+01
	Mediam	8.721E+02	8.648E+02	**8.139E+02**	**8.135E+02**	8.739E+02	8.731E+02	9.041E+02	8.985E+02
	Mean	8.758E+02	8.667E+02	**8.142E+02**	**8.137E+02**	8.722E+02	8.713E+02	9.038E+02	8.967E+02
	Worst	2.565E+03	2.164E+03	**9.000E+02**	**9.000E+02**	9.005E+02	9.001E+02	1.226E+03	1.956E+03
	Best	1.296E+03	9.051E+02	**9.000E+02**	**9.000E+02**	**9.000E+02**	**9.000E+02**	**9.000E+02**	**9.000E+02**
F5	STD	3.181E+02	3.814E+02	2.111E-14	**0.000E+00**	1.161E-01	2.271E-02	7.470E+01	2.052E+02
	Mediam	1.813E+03	1.184E+03	**9.000E+02**	**9.000E+02**	**9.000E+02**	**9.000E+02**	**9.000E+02**	**9.000E+02**
	Mean	1.863E+03	1.326E+03	**9.000E+02**	**9.000E+02**	9.000E+02	**9.000E+02**	9.195E+02	9.613E+02
	Worst	**1.440E+04**	3.145E+04	1.919E+03	1.952E+03	1.504E+04	1.705E+04	2.955E+08	3.808E+08
	Best	1.857E+03	4.268E+03	**1.806E+03**	1.814E+03	3.534E+03	3.085E+03	2.844E+05	8.800E+04
F6	STD	3.488E+03	7.385E+03	**2.713E+01**	3.368E+01	3.028E+03	3.672E+03	5.263E+07	6.351E+07
	Mediam	3.007E+03	8.640E+03	**1.849E+03**	1.857E+03	8.500E+03	7.189E+03	4.355E+07	4.348E+07
	Mean	4.449E+03	1.133E+04	**1.848E+03**	1.860E+03	8.541E+03	7.943E+03	4.645E+07	4.643E+07
	Worst	2.187E+03	2.145E+03	**2.025E+03**	2.027E+03	2.034E+03	2.035E+03	2.325E+03	2.279E+03
	Best	2.059E+03	2.034E+03	2.005E+03	**2.003E+03**	2.016E+03	2.022E+03	2.109E+03	2.109E+03
F7	STD	4.575E+01	2.156E+01	1.820E+01	6.898E+00	3.341E+00	**2.492E+00**	5.790E+01	4.925E+01
	Mediam	2.124E+03	2.062E+03	**2.022E+03**	**2.022E+03**	2.027E+03	2.030E+03	2.193E+03	2.166E+03
	Mean	2.122E+03	2.069E+03	**2.018E+03**	**2.018E+03**	2.027E+03	2.030E+03	2.191E+03	2.171E+03
	Worst	2.439E+03	**2.249E+03**	**2.221E+03**	**2.221E+03**	2.230E+03	2.231E+03	2.399E+03	2.414E+03
	Best	2.220E+03	2.225E+03	**2.209E+03**	2.220E+03	2.227E+03	2.227E+03	2.238E+03	2.241E+03
F8	STD	6.617E+01	5.841E+00	2.185E+00	**3.510E-01**	9.189E-01	9.102E-01	4.681E+01	4.669E+01
	Mediam	2.223E+03	2.229E+03	**2.221E+03**	**2.221E+03**	2.228E+03	2.228E+03	2.262E+03	2.262E+03
	Mean	2.276E+03	2.230E+03	**2.220E+03**	**2.221E+03**	2.228E+03	2.228E+03	2.279E+03	2.279E+03
	Worst	2.510E+03	**2.481E+03**	**2.481E+03**	**2.481E+03**	**2.481E+03**	**2.481E+03**	2.650E+03	2.573E+03
	Best	**2.481E+03**	**2.481E+03**	**2.481E+03**	**2.481E+03**	**2.481E+03**	**2.481E+03**	**2.481E+03**	**2.481E+03**
F9	STD	9.018E+00	4.146E-02	8.444E-14	**0.000E+00**	8.444E-14	4.547E-13	4.476E+01	3.042E+01
	Median	**2.481E+03**	**2.481E+03**	**2.481E+03**	**2.481E+03**	**2.481E+03**	**2.481E+03**	2.521E+03	2.521E+03
	Mean	2.484E+03	**2.481E+03**	**2.481E+03**	**2.481E+03**	**2.481E+03**	**2.481E+03**	2.532E+03	2.522E+03
	Worst	5.316E+03	2.999E+03	**2.619E+03**	2.621E+03	3.419E+03	3.657E+03	6.380E+03	6.380E+03
	Best	2.501E+03	2.500E+03	**2.400E+03**	**2.400E+03**	2.500E+03	2.500E+03	2.501E+03	2.500E+03
F10	STD	1.032E+03	1.043E+02	5.853E+01	**4.959E+01**	2.049E+02	2.666E+02	9.516E+02	7.256E+02
	Median	4.197E+03	**2.501E+03**	**2.500E+03**	**2.500E+03**	**2.500E+03**	**2.500E+03**	6.057E+03	6.065E+03
	Mean	3.973E+03	2.539E+03	**2.492E+03**	2.517E+03	2.607E+03	2.601E+03	5.773E+03	5.880E+03
	Worst	**2.900E+03**	3.171E+04	**2.900E+03**	**2.900E+03**	**2.900E+03**	**2.900E+03**	9.697E+03	**2.900E+03**
	Best	**2.900E+03**	1.058E+04	**2.900E+03**	**2.900E+03**	**2.900E+03**	**2.900E+03**	**2.900E+03**	**2.900E+03**
F11	STD	4.222E-13	5.096E+03	2.733E-13	2.960E-13	4.468E-13	1.194E-13	1.241E+03	**0.000E+00**
	Median	**2.900E+03**	1.721E+04	**2.900E+03**	**2.900E+03**	**2.900E+03**	**2.900E+03**	**2.900E+03**	**2.900E+03**
	Mean	**2.900E+03**	1.823E+04	**2.900E+03**	**2.900E+03**	**2.900E+03**	**2.900E+03**	3.127E+03	**2.900E+03**
	Worst	4.005E+03	3.340E+03	2.966E+03	**2.945E+03**	2.958E+03	2.957E+03	**2.900E+03**	**2.900E+03**
	Best	3.265E+03	2.946E+03	2.931E+03	2.931E+03	2.935E+03	2.932E+03	**2.900E+03**	**2.900E+03**
F12	STD	1.961E+02	7.844E+01	6.770E+00	**4.002E+00**	5.706E+00	5.604E+00	7.151E-05	**7.086E-05**
	Median	3.529E+03	2.968E+03	2.934E+03	**2.936E+03**	2.945E+03	2.943E+03	**2.900E+03**	**2.900E+03**
	Mean	3.541E+03	2.999E+03	2.936E+03	2.936E+03	2.946E+03	2.944E+03	**2.900E+03**	**2.900E+03**
Original vs. TAM-enhanced	**+/=/-**	**8/0/4**		**6/1/5**		**7/2/3**		**8/1/3**	

5. Practical Applications

In this section, we analyze the real-world impact of the improved TAM algorithms to demonstrate their powerful applications. This analysis includes comparisons with widely cited and efficient algorithms, including PSO, SHADE, JaDE, and CMA-ES, alongside the TAM-improved variants discussed in previous chapters. The feasibility and limitations of TAM are highlighted through an in-depth discussion and analysis of six engineering design problems proposed by CEC2020, six feature selection problems based on Extreme Learning Machines (ELMs), and three

Table 6
Six engineering design challenges

Problems	Name	dim	$f_{\min}$
RW01	Hydro-Static Thrust Bearing Design	4	1625.443
RW02	Optimal Design of Industrial Refrigeration System	14	0.032213
RW03	Pressure Vessel Design	4	5885.333
RW04	Multiple Disk Clutch Brake Design	5	0.235242
RW05	Cantilever Beam Design	5	1.339956
RW06	Weight Minimization of a Speed Reducer	7	2994.424

photovoltaic model parameter identification challenges.To ensure the reproducibility and reliability of the results, each experiment was conducted independently 20 times.

5.1. Engineering design problems

Metaheuristic algorithms have long been applied to tackle engineering optimization challenges, and this remains a vibrant area of research [1, 7, 27]. In this section, we examine the performance of the Trend-Aware Mechanism by testing it on six engineering design optimization problems. These challenges, sourced from the CEC single-objective constrained optimization competition, are summarized in Table 6. Detailed descriptions of these problems can be found in [40]. The experiments use a population size of 50 and run for 2000 iterations, following the common convention that the total number of function evaluations is capped at $10,000 \times \text{dim}$.

Engineering-design optimization is notoriously difficult because real-world objective functions are highly nonlinear and must satisfy multiple, tightly coupled constraints. Although numerous constraint-handling techniques (CHTs) have been proposed—e.g. superiority of feasible solutions, adaptive or self-tuning penalties, and the ε-constraint method—we employ the following static quadratic penalty for every inequality constraint $g_j(\mathbf{x}) \le 0$:

$$F(\mathbf{x}) = f(\mathbf{x}) + \lambda \sum_{j=1}^{m} \left[\max\{0,\ g_j(\mathbf{x})\}\right]^2, \qquad \lambda = 10^{20},$$

where $f(\mathbf{x})$ is the original objective and m is the number of constraints. Only violated constraints contribute, since $\max\{0,\ g_j(\mathbf{x})\} = 0$ whenever $g_j(\mathbf{x}) \le 0$. The extremely large constant λ ensures that any infeasible solution is dominated by a feasible one, eliminating the need for parameter tuning. Variable bounds are enforced by reflection: if $x_d \notin [L_d, U_d]$, we project immediately by $x_d \leftarrow \min\big(U_d, \max(L_d, x_d)\big)$, keeping all evaluations within the prescribed domain. The results on the six real-world problems, together with algorithm rankings, are summarized in Table 7.

Tables illustrate the effectiveness of TAM-enhanced algorithms in addressing engineering optimization problems. Notably, while TAM-enhanced SHADE and JaDE demonstrate greater stability and improved convergence in RW01 and RW02, the enhancements are modest, as SHADE and JaDE already perform well in solving such engineering design problems. In contrast, TAM's integration with PSO shows a more pronounced impact, particularly in RW03 and RW06, significantly improving optimization results. For CMA-ES, TAM also plays an important role, but its effectiveness is limited to three of the engineering problems due to conflicts between the penalty function handling of engineering constraints and the matrix positivity requirements in CMA-ES. Overall, TAM successfully enhances four widely recognized metaheuristic algorithms and demonstrates satisfactory performance, confirming its value as an effective improvement strategy.

5.2. Feature selection problems

To assess the effectiveness of the proposed TAM in feature selection challenges, six datasets from the UCI database [1] and Kaggle database [2] were selected for testing, with relevant information provided in Table 8. For all datasets, the ELM classifier was employed for classification. The hidden layer of the ELM was configured with 50 nodes, and the Sigmoid activation function was applied. Additionally, the random seed for disrupting the dataset was set to 42, and the parameters for the individual algorithms, as well as the design details, were consistent with those described above. The population size and the number of iterations were both set to 50. Table 9 compares the classification accuracies of the PSO, SHADE, JaDE, and CMA-ES algorithms before and after improvement.

[1]`https://archive.ics.uci.edu`
[2]`https://www.kaggle.com`

Table 7
Results of six engineering constrained optimization challenges

Problems	PSO		TAM-PSO		SHADE		TAM-SHADE	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD
RW01	**1.969E+03**	3.277E+02	2.038E+03	**2.203E+02**	**1.616E+03**	**0.000E+00**	**1.616E+03**	**0.000E+00**
RW02	N/A	N/A	N/A	N/A	3.221E-02	5.666E-18	**3.221E-02**	**4.626E-18**
RW03	6.200E+03	1.778E+02	**6.157E+03**	7.67E+01	**5.885E+03**	**0.000E+00**	**5.885E+03**	**0.000E+00**
RW04	**2.352E-01**	**0.000E+00**	**2.352E-01**	**0.000E+00**	**2.352E-01**	**0.000E+00**	**2.352E-01**	**0.000E+00**
RW05	**1.340E+00**	**8.034E-07**	1.340E+00	2.408E-04	**1.340E+00**	**2.341E-16**	**1.340E+00**	**2.341E-16**
RW06	3.120E+03	1.076E+02	**3.082E+03**	**7.222E+01**	**2.994E+03**	**0.000E+00**	**2.994E+03**	**0.000E+00**
+/=/-			**3/1/1**				**1/5/0**	

Problems	JaDE		TAM-JaDE		CMAES		TAM-CMAES	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD
RW01	1.697E+03	5.496E+01	**1.617E+03**	**1.364E+00**	N/A	N/A	N/A	N/A
RW02	4.528E-02	1.014E-02	**4.243E-02**	**7.644E-02**	N/A	N/A	N/A	N/A
RW03	**5.885E+03**	**0.000E+00**	**5.885E+03**	**0.000E+00**	N/A	N/A	N/A	N/A
RW04	**2.352E-01**	**0.000E+00**	**2.352E-01**	**0.000E+00**	**2.352E-01**	**0.000E+00**	**2.352E-01**	**0.000E+00**
RW05	**1.340E+00**	**4.853E-16**	**1.340E+00**	**4.853E-16**	1.340E+00	1.406E-05	**1.340E+00**	**1.405E-05**
RW06	**2.994E+03**	**0.000E+00**	**2.994E+03**	**0.000E+00**	3.034E+03	**5.144E+01**	**3.024E+03**	7.094E+01
+/=/-			**2/4/0**				**1/2/0**	

Table 8
Six datasets used in the experiment

No.	Datasets	Features	Samples
D01	DARWIN	451	174
D02	Breast Cancer (Diagnostic)	30	569
D03	Students Performance	14	2392
D04	Congress	16	435
D05	M-of-n	13	1000
D06	Lung Cancer	56	32

Table 9
Comparison of classification accuracy (%)

Algorithms	D01	D02	D03	D04	D05	D06
PSO	65.28	**100.00**	72.64	96.00	97.13	67.14
TAM-PSO	**73.02**	100.00	**72.86**	**96.54**	**97.77**	**74.29**
SHADE	75.09	**100.00**	73.42	96.31	98.30	70.00
TAM-SHADE	**75.29**	**100.00**	**73.87**	**96.62**	**98.77**	**72.86**
JaDE	**77.17**	**100.00**	73.21	97.00	99.03	71.43
TAM-JaDE	76.61	**100.00**	**73.91**	**97.08**	**99.27**	**80.00**
CMAES	68.48	**100.00**	71.33	95.85	94.20	**77.14**
TAM-CMAES	**74.34**	**100.00**	**71.45**	**96.62**	**98.67**	71.43

As shown in Table 9, the TAM strategy demonstrates effectiveness in feature selection, achieving significant improvements across six test sets. However, the improvement is less pronounced for JaDE on the D01 dataset and CMA-ES on the D06 dataset. These results show the overall effectiveness and broad applicability of TAM strategy.

5.3. Parameters identification of photovoltaic models

To conduct a more comprehensive evaluation of TAM's performance, we utilized the algorithm for parameter extraction tasks across three aspects: the single-diode, dual-diode, and PV module models. Data for the single- and dual-diode models were obtained from a commercial silicon R.T.C. France solar cell with a 57 mm diameter at 33°C,

Table 10
Range of parameters

Parameters	Single-Diode		Dual-Diode	
	LB	UB	LB	UB
I_{ph}/A	0	1	0	2
I_{sd}, I_{sd1}, I_{sd2}/μA	0	1	0	50
R_s/Ω	0	0.5	0	2
R_{sh}/Ω	0	100	0	2000
n, n_1, n_2	1	2	1	50

Table 11
Comparison of Various Parameter Extraction Methods for the single-diode model

Algorithms	I_{ph}/A	I_{sd}/μA	R_s/Ω	R_{sh}/Ω	n	RMSE
PSO	7.604E-01	6.743E-07	3.336E-02	9.214E+01	1.559E+00	1.299E-02
TAM-PSO	7.613E-01	2.958E-07	3.674E-02	4.721E+01	1.472E+00	**2.573E-03**
SHADE	7.608E-01	3.230E-07	3.638E-02	5.372E+01	1.481E+00	**9.860E-04**
TAM-SHADE	7.608E-01	3.230E-07	3.638E-02	5.372E+01	1.481E+00	**9.860E-04**
JaDE	7.608E-01	3.230E-07	3.638E-02	5.372E+01	1.481E+00	9.879E-04
TAM-JaDE	7.608E-01	3.230E-07	3.638E-02	5.372E+01	1.481E+00	**9.873E-04**
CMAES	7.613E-01	7.613E-01	3.189E-02	8.910E+01	1.592E+00	5.182E-03
TAM-CMAES	7.622E-01	1.000E-06	3.089E-02	8.084E+01	1.604E+00	**3.202E-03**

Table 12
Comparison of Various Parameter Extraction Methods for the dual-diode model

Algorithms	I_{ph}/A	I_{sd1}/μA	R_s/Ω	R_{sh}/Ω	n_1	I_{sd2}/μA	n_2	RMSE
PSO	7.608E-01	1.000E-09	3.729E-02	6.394E+01	1.140E+00	7.232E-07	1.600E+00	1.179E-02
TAM-PSO	7.616E-01	1.000E-06	3.106E-02	6.782E+01	1.605E+00	1.005E-09	2.000E+00	**9.444E-03**
SHADE	7.608E-01	1.521E-07	3.647E-02	5.396E+01	1.911E+00	2.925E-07	1.473E+00	1.220E-03
TAM-SHADE	7.608E-01	3.913E-07	3.654E-02	5.461E+01	1.998E+00	2.702E-07	1.466E+00	**1.216E-03**
JaDE	7.607E-01	7.472E-08	3.640E-02	5.465E+01	1.924E+00	3.103E-07	1.478E+00	1.280E-03
TAM-JaDE	7.607E-01	2.497E-07	3.665E-02	5.301E+01	1.460E+00	1.616E-07	1.775E+00	**1.249E-03**
CMAES	7.639E-01	1.000E-06	2.958E-02	1.000E+02	2.000E+00	1.000E-06	1.614E+00	2.683E-02
TAM-CMAES	7.661E-01	1.000E-06	2.606E-02	3.064E+01	1.757E+00	1.000E-06	1.654E+00	**2.148E-02**

Table 13
Comparison of Various Parameter Extraction Methods for the PV module model

Algorithms	I_{ph}/A	I_{sd1}/μA	R_s/Ω	R_{sh}/Ω	n	RMSE
PSO	7.608E-01	1.000E-09	3.729E-02	6.394E+01	1.140E+00	**9.163E-02**
TAM-PSO	7.616E-01	1.000E-06	3.106E-02	6.782E+01	1.605E+00	9.445E-02
SHADE	7.608E-01	1.521E-07	3.647E-02	5.396E+01	1.911E+00	**2.425E-03**
TAM-SHADE	7.608E-01	3.913E-07	3.654E-02	5.461E+01	1.998E+00	**2.425E-03**
JaDE	7.607E-01	7.472E-08	3.640E-02	5.465E+01	1.924E+00	**2.425E-03**
TAM-JaDE	7.607E-01	2.497E-07	3.665E-02	5.301E+01	1.460E+00	**2.425E-03**
CMAES	1.026E+00	4.883E-06	1.179E+00	2.000E+03	5.000E+01	8.264E-02
TAM-CMAES	1.026E+00	4.903E-06	1.165E+00	1.835E+03	5.000E+01	**7.219E-02**

while the data for the PV module model were sourced from a polycrystalline Photowatt-PWP201 cell at 45°C [41]. The extracted parameter ranges are outlined in Table 10. The TAM was incorporated into the four algorithms, including their respective variants, as discussed earlier. The experimental outputs are presented in Tables 11, 12, and 13. All algorithms were set with a population size of 50 and a maximum of 500 iterations. For a full description of the experimental setup, refer to the literature [42].

The results presented in the table indicate that TAM-SHADE outperformed all other algorithms, while PSO and CMA-ES performed the worst. In all three experiments, TAM proved effective in enhancing the performance of the four popular algorithms, except in the PV module model, where CMA-ES slightly outperformed TAM-CMAES. This highlights the effectiveness and power of TAM.

6. Conclusions and Future Works

This paper introduced the Trend-Aware Mechanism (TAM), a novel framework designed to enhance the exploration and exploitation capabilities of metaheuristic algorithms by effectively leveraging historical search trajectories. Unlike traditional strategies, TAM dynamically integrates individual and population-level search trends to identify promising directions, achieving a balanced exploration-exploitation trade-off.

Empirical evaluations demonstrated significant performance improvements when integrating TAM with state-of-the-art algorithms, including Particle Swarm Optimization (PSO), Success-History Adaptive Differential Evolution (SHADE), Adaptive Differential Evolution (JADE), and Covariance Matrix Adaptation Evolution Strategy (CMA-ES). Extensive experiments across diverse benchmarks highlighted TAM's effectiveness in convergence stability, accuracy, and robustness. Its practical applications in feature selection, engineering optimization, and photovoltaic parameter estimation further validate its broad applicability.

Nevertheless, TAM has certain limitations. It exhibits sensitivity to parameter tuning and incurs increased computational costs in high-dimensional spaces. Additionally, its benefits diminish for unimodal functions, where trend analysis provides minimal advantage.

Future research directions to address these limitations and further enhance TAM include:

- **Spatial Partitioning of Historical Data:** Investigating spatial partitioning methods, such as k-d trees, octrees, or grid encoding, to efficiently segment the historical search space, enabling rapid identification of nearby candidate solutions.
- **Binary Encoding for Efficient Computation:** Exploring compact binary or hash-based representations, including locality-sensitive hashing, to significantly accelerate distance computations and reduce complexity.
- **Hybridization with History-Based Adaptive Mechanisms:** Developing hybrid strategies by integrating TAM with other history-informed adaptive algorithms, including JaDE, Memory Backtracking Strategy (MBS), Historical Knowledge-Transfer (HKT), and archive-based Differential Evolution variants, aiming to further enhance adaptive performance and robustness.

In conclusion, TAM establishes a robust foundation for historical trend-informed metaheuristic optimization. The proposed future enhancements promise to expand TAM's applicability and computational efficiency, providing clear and valuable directions for ongoing research and practical advancements.

CRediT authorship contribution statement

Junbo Jacob Lian: Formal analysis, Conceptualization, Resources, Investigation, Methodology, Software, Data curation, Funding acquisition, Visualization, Writing - original draft, Writing - review & editing. **Kaichen OuYang:** Formal analysis, Investigation, Resources, Validation. **Rui Zhong:** Validation, Writing - review & editing. **Yujun Zhang:** Validation, Writing - review & editing. **Shipeng Luo:** Validation, Writing - review & editing. **Ling Ma:** Data curation, Validation, Writing - review & editing. **Xincan Wu:** Visualization. **Huiling Chen:** Formal analysis, Supervision, Funding acquisition, Writing - review & editing.

Statement and Declarations

Competing interest

he authors declare that there are no competing interests.

Data availability

The open code will be available at `https://github.com/junbolian/Trend-Aware-Mechanism`.

Acknowledgment

This research is financially supported by the National Natural Science Foundation of China (Grant No. 62076185, 62301367).

References

[1] J. Lian, G. Hui, Human evolutionary optimization algorithm, Expert Systems with Applications 241 (2024) 122638.

[2] J. Lian, G. Hui, L. Ma, T. Zhu, X. Wu, A. A. Heidari, Y. Chen, H. Chen, Parrot optimizer: Algorithm and applications to medical problems, Computers in Biology and Medicine 172 (2024) 108064.

[3] Y. Zhang, S. Li, Y. Wang, Y. Yan, J. Zhao, Z. Gao, Self-adaptive enhanced learning differential evolution with surprisingly efficient decomposition approach for parameter identification of photovoltaic models, Energy Conversion and Management 308 (2024) 118387.

[4] Y. J. Zhang, Y. F. Wang, Y. X. Yan, J. Zhao, Z. M. Gao, Self-adaptive hybrid mutation slime mould algorithm: case studies on uav path planning, engineering problems, photovoltaic models and infinite impulse response, Alexandria Engineering Journal 98 (2024) 364–389.

[5] W. Zhou, J. Lian, J. Zhang, Z. Mei, Y. Gao, G. Hui, Tomato storage quality predicting method based on portable electronic nose system combined with woa-svm model, Journal of Food Measurement and Characterization 17 (4) (2023) 3654–3664.

[6] J. Lian, L. Ma, X. Wu, T. Zhu, Q. Liu, Y. Sun, X. Lou, Visualized pattern recognition optimization for apple mechanical damage by laser relaxation spectroscopy, International Journal of Food Properties 26 (1) (2023) 1566–1578.

[7] J. Lian, P. Wu, W. Han, Y. Xie, Y. Zheng, Y. Xu, G. Hui, Discrimination of chinese prickly ash origin place using electronic nose system and feature extraction with support vector boosting machine, Cogent Food & Agriculture 11 (1) (2025) 2464939.

[8] S. Yang, H. Ma, Y. Bi, Y. Tian, L. Zhang, Y. Jin, X. Zhang, An evolutionary multi-objective neural architecture search approach to advancing cognitive diagnosis in intelligent education, IEEE Transactions on Evolutionary Computation (2024).

[9] Y. Zhang, Y. Wang, L. Tao, Y. Yan, J. Zhao, Z. Gao, Self-adaptive classification learning hybrid jaya and rao-1 algorithm for large-scale numerical and engineering problems, Engineering Applications of Artificial Intelligence 114 (2022) 105069.

[10] R. Zhong, Z. Wang, Y. Zhang, J. J. Lian, J. Yu, H. Chen, Integrating competitive framework into differential evolution: Comprehensive performance analysis and application in brain tumor detection, Applied Soft Computing 174 (2025) 112995.

[11] H. Jia, C. Lu, Z. Xing, Memory backtracking strategy: An evolutionary updating mechanism for meta-heuristic algorithms, Swarm and Evolutionary Computation 84 (2024) 101456.

[12] P. J. Van Laarhoven, E. H. Aarts, Simulated annealing, Springer, 1987.

[13] R. V. Rao, V. J. Savsani, D. P. Vakharia, Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems, Computer-Aided Design 43 (3) (2011) 303–315.

[14] S. Das, P. N. Suganthan, Differential evolution: A survey of the state-of-the-art, IEEE Transactions on Evolutionary Computation 15 (1) (2010) 4–31.

[15] J. H. Holland, Genetic algorithms, Scientific American 267 (1) (1992) 66–73.

[16] H.-G. Beyer, H.-P. Schwefel, Evolution strategies–a comprehensive introduction, Natural Computing 1 (2002) 3–52.

[17] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Transactions on Evolutionary Computation 3 (2) (1999) 82–102.

[18] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (abc) algorithm, Journal of Global Optimization 39 (2007) 459–471.

[19] M. Dorigo, M. Birattari, T. Stutzle, Ant colony optimization, IEEE Computational Intelligence Magazine 1 (4) (2006) 28–39.

[20] J. Lian, T. Zhu, L. Ma, X. Wu, A. A. Heidari, Y. Chen, G. Hui, The educational competition optimizer, International Journal of Systems Science 55 (15) (2024) 3185–3222.

[21] J. Xue, B. Shen, Dung beetle optimizer: A new meta-heuristic algorithm for global optimization, The Journal of Supercomputing 79 (7) (2023) 7305–7336.

[22] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, Gsa: A gravitational search algorithm, Information Sciences 179 (13) (2009) 2232–2248.

[23] H. Su, et al., Rime: A physics-based optimization, Neurocomputing 532 (2023) 183–214.

[24] D. H. Wolpert, W. G. Macready, No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 67–82.

[25] R. Zhong, F. Peng, J. Yu, M. Munetomo, Q-learning based vegetation evolution for numerical optimization and wireless sensor network coverage optimization, Alexandria Engineering Journal 87 (2024) 148–163.

[26] R. Zhong, C. Zhang, J. Yu, Hierarchical rime algorithm with multiple search preferences for extreme learning machine training, Alexandria Engineering Journal 110 (2025) 77–98.

[27] H. Jia, C. Lu, Guided learning strategy: A novel update mechanism for metaheuristic algorithms design and improvement, Knowledge-Based Systems 286 (2024) 111402.

[28] R. Zhong, J. Yu, C. Zhang, M. Munetomo, Srime: A strengthened rime with latin hypercube sampling and embedded distance-based selection for engineering optimization problems, Neural Computing and Applications 36 (12) (2024) 6721–6740.

[29] J. Hong, B. Shen, J. Xue, A. Pan, A vector-encirclement-model-based sparrow search algorithm for engineering optimization and numerical optimization problems, Applied Soft Computing 131 (2022) 109777.

[30] Y. Xu, Z. Yang, X. Li, H. Kang, X. Yang, Dynamic opposite learning enhanced teaching–learning-based optimization, Knowledge-Based Systems 188 (2020) 104966.

[31] Y. Xu, H. Chen, J. Luo, Q. Zhang, S. Jiao, X. Zhang, Enhanced moth-flame optimizer with mutation strategy for global optimization, Information Sciences 492 (2019) 181–203.

[32] Y. Zhang, Y. Wang, Y. Yan, J. Zhao, Z. Gao, Historical knowledge transfer driven self-adaptive evolutionary multitasking algorithm with hybrid resource release for solving nonlinear equation systems, Swarm and Evolutionary Computation (2024) 101754.

[33] J. Zhang, A. C. Sanderson, Jade: Adaptive differential evolution with optional external archive, IEEE Transactions on Evolutionary Computation 13 (5) (2009) 945–958.

[34] R. Tanabe, A. Fukunaga, Success-history based parameter adaptation for differential evolution, in: 2013 IEEE Congress on Evolutionary Computation, IEEE, 2013, pp. 71–78.

[35] H. R. Tizhoosh, Opposition-based learning: A new scheme for machine intelligence, in: Proc. of 2005 International Conference on Computational Intelligence for Modelling, Control and Automation (CIMCA 2005) and Intelligent Agents, Web Technologies and Internet Commerce (IAWTIC 2005), IEEE Computer Society, 2005, pp. 695–701.

[36] S. Mahdavi, S. Rahnamayan, K. Deb, Opposition based learning: A literature review, Swarm and Evolutionary Computation 39 (2018) 1–23.

[37] F. Liang, K. C. Tan, F. Feng, et al., Historical knowledge transfer in multi-task evolutionary optimization, IEEE Transactions on Evolutionary Computation 25 (3) (2021) 456–470.

[38] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95–International Conference on Neural Networks, Vol. 4, IEEE, 1995, pp. 1942–1948.

[39] N. Hansen, S. D. Müller, P. Koumoutsakos, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es), Evolutionary Computation 11 (1) (2003) 1–18.

[40] A. Kumar, G. Wu, M. Z. Ali, R. Mallipeddi, P. N. Suganthan, S. Das, A test-suite of non-convex constrained optimization problems from the real-world and some baseline results, Swarm and Evolutionary Computation 56 (2020) 100693.

[41] T. Easwarakhanthan, J. Bottin, I. Bouhouch, C. Boutrit, Nonlinear minimization algorithm for determining the solar cell parameters with microcomputers, International Journal of Solar Energy 4 (1) (1986) 1–12.

[42] K. Yu, J. Liang, B. Qu, X. Chen, H. Wang, Parameters identification of photovoltaic models using an improved jaya optimization algorithm, Energy Conversion and Management 150 (2017) 742–753.