Knowledge-Based Systems 310 (2025) 113020

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Crested ibis algorithm and its application in human-powered aircraft design

Yuefeng Xu [a], Rui Zhong [b], Chao Zhang [c], Jun Yu [d,*]

[a] Graduate School of Science and Technology, Niigata University, Niigata, Japan
[b] Information Initiative Center, Hokkaido University, Sapporo, Japan
[c] Faculty of Engineering, University of Toyama, Toyama, Japan
[d] Institute of Science and Technology, Niigata University, Niigata, Japan

ARTICLE INFO

Keywords:
Evolutionary computation (EC)
Bio-inspired algorithm
Crested ibis algorithm (CIA)
Predatory behaviors

ABSTRACT

Inspired by the observation of crested ibis foraging behavior, we propose a novel bio-inspired optimization algorithm called Crested Ibis Algorithm (CIA). We designed different exploration strategies by simulating the success or failure of ibis foraging and the escape behavior of fish from ibis foraging. Moreover, the dynamic balance between exploration and exploitation was realized through the information interaction mechanism of the two populations. To validate the performance of the proposed CIA algorithm, we conducted comparative experiments with 14 competitive algorithms on different dimensions of the CEC2017 and CEC2022 benchmark suites and showed excellent performance. In addition, we extend the CIA algorithm to the problem of Human-Powered Aircraft Design optimization (HPA). Experiments and statistical tests demonstrate the proposed CIA's superb performance and outstanding robustness. The source code is available at https://github.com/RuiZhong961230/CIA.

1. Introduction

Evolutionary computation (EC) [1–4] is an advanced optimization approach for solving complex optimization problems by simulating evolutionary mechanisms such as natural selection and genetic variation in organisms. Unlike traditional optimization methods, such as the Newton–Raphson method [5], Quasi-Newton methods [6], Integer Programming [7], and Gradient Descent [8], EC does not rely on derivative information or the continuity of the objective function. Instead, EC gradually approximates the optimal solution through iterative processes and updates to the feasible solutions. Based on these properties, EC effectively solves complex optimization problems [9,10], particularly in non-convex optimization [11], multi-modal optimization [12], and non-smooth objective functions [13]. Consequently, EC has become increasingly indispensable in various fields, such as engineering design [14], resource allocation [15], and scheduling [16], achieving notable success.

Since the success and widespread adoption of the Genetic Algorithm (GA) [17,18], researchers have proposed numerous Meta-heuristic Algorithms (MAs), significantly enriching the field of EC. Based on their sources of inspiration [19], MAs can be broadly categorized into four classes: evolutionary algorithms, bio-inspired algorithms, human-inspired algorithms, and law-based algorithms [20–33]. Developing these algorithms addresses diverse requirements for algorithm design across various fields, offering multiple options and strategies for tackling complex optimization problems (see Fig. 1).

With the rapid advancement of artificial intelligence technology and the exponential growth of information, the complexity and diversity of optimization problems have increased significantly [34]. In emerging domains such as neural network training [35,36], neural architecture search [37], and hyper-parameter optimization for deep learning [38,39], traditional EC algorithms are increasingly demonstrating their limitations. Similarly, these algorithms often struggle to address the diverse requirements of cross-disciplinary optimization challenges, including data science [40], bio-informatics [41], financial modeling [42], supply chain network optimization [43], and gene mutation prediction [44]. Therefore, proposing novel EC algorithms has become imperative to enhance their robustness, optimization efficiency, and adaptability. This effort is essential for solving real-world optimization problems and pivotal for advancing algorithmic theory and sustaining the vitality of the EC field.

This paper proposes a bio-inspired meta-heuristic algorithm, Crested Ibis Algorithm (CIA), inspired by the foraging behavior of the crested ibis. We simplified the model by analyzing the ibis' behavior. We designed the CIA algorithm as a mechanism for two-population interaction by simulating the behavior of the predator ibis and the

* Corresponding author.
E-mail addresses: f24c024f@mail.cc.niigata-u.ac.jp (Y. Xu), rui.zhong.u5@elms.hokudai.ac.jp (R. Zhong), zhang@eng.u-toyama.ac.jp (C. Zhang), yujun@ie.niigata-u.ac.jp (J. Yu).

https://doi.org/10.1016/j.knosys.2025.113020
Received 12 July 2024; Received in revised form 28 November 2024; Accepted 11 January 2025
Available online 21 January 2025

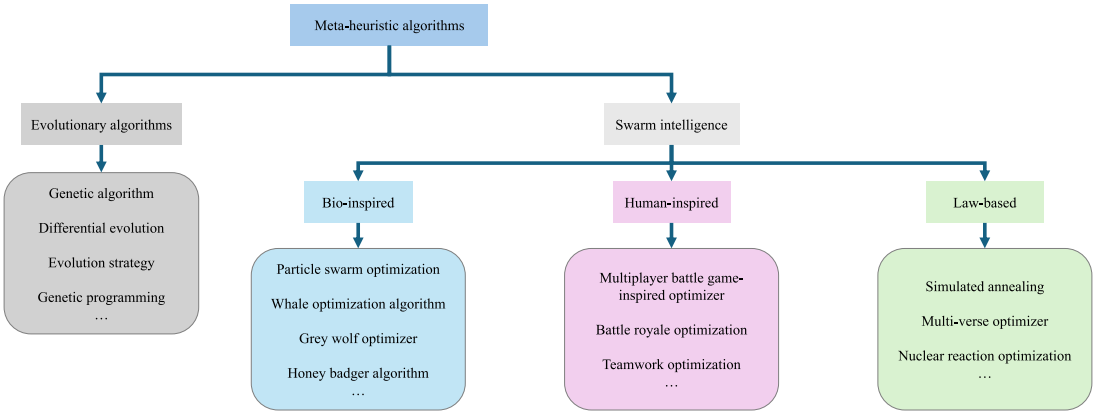

Fig. 1. The categorization of meta-heuristic algorithms.

fish of the prey. The two populations are updated independently, and the interactions between the populations ensure the convergence of the algorithm and maintain the diversity of the populations, enabling CIA to solve complex optimization problems characterized by numerous local optima efficiently. The population diversity further enhances the algorithm's ability to escape from local optima when stagnation occurs. To evaluate the performance and robustness of CIA, we design experiments using two distinct suites of benchmark functions. The CEC2022 suite assesses the algorithm's effectiveness in low-dimensional optimization scenarios, whereas the CEC2017 suite provides a comprehensive benchmarking framework across various problem types. We select a recent practical challenge, the Human-Powered Aircraft Design problem (HPA) [45], to further validate the CIA's applicability in real-world contexts.

For comparative evaluation, evaluation experiments include four classical optimization algorithms widely recognized in the field: Differential Evolution (DE), Particle Swarm Optimization (PSO), Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [46], and the Whale Optimization Algorithm (WOA). To ensure robustness in the comparison, five well-established improved variants are considered: Comprehensive Learning Particle Swarm Optimization (CL-PSO) [47], Differential Evolution with Strategy Adaptation (SaDE) [48], Adaptive Differential Evolution with Optional External Archive (JADE) [49], Success-History Based Adaptive Differential Evolution with Linear Population Size Reduction (L-SHADE) [50], and Phasor Particle Swarm Optimization (PPSO) [51]. Additionally, five state-of-the-art algorithms proposed in the last two years are included to benchmark the proposed approach against cutting-edge methodologies, namely the Fata Morgana Algorithm (FATA) [52], the Mantis Search Algorithm (MSA) [53], the Hippopotamus Optimization Algorithm (HO) [54], the Coati Optimization Algorithm (COA) [55], and the Multiplayer Battle Game-Inspired Optimizer (MBGO). This comprehensive selection ensures a balanced evaluation by covering classical and contemporary evolutionary computation developments.

Through comprehensive experiments, we evaluated the performance and robustness of the proposed CIA algorithm. The results indicate that the proposed CIA algorithm performs well on the benchmark suites and demonstrates high robustness and efficiency when applied to real-world engineering optimization problems. The key contributions of this study include:

- We propose a novel bio-inspired meta-heuristic algorithm named Crested Ibis Algorithm motivated by the foraging behavior of the crested ibis.
- The balance between exploration and exploitation is achieved through bi-specific independent evolution and inter-population information exchange.
- CIA dynamically adjusts the scope and intensity of its exploration process by incorporating adaptive factors.
- Extensive experimental evaluations and the application in HPA Design have demonstrated the high performance and robustness of the proposed algorithm.

The structure of the remaining sections is outlined as follows. Section 2 provides an overview of the biological characteristics of the crested ibis, which serve as the inspiration for the proposed CIA algorithm, and presents its mathematical model. Section 3 examines the sensitivity of the algorithm's tunable parameters. Section 4 details the performance evaluation of the CIA algorithm across multiple dimensions using two benchmark suites: CEC2017 and CEC2022. Section 5 explores the application of the algorithm to a real-world engineering problem, the human-powered aircraft (HPA) design. In Section 6, we analyze the experimental results, highlighting the algorithm's effectiveness and scalability. Finally, Section 7 concludes the paper with a summary of our findings.

2. Crested ibis algorithm

This section discusses the inspiration and mathematical model of the proposed CIA algorithm.

2.1. Crested ibis general biology

Once a widely distributed bird species, the crested ibis (Nipponia nippon) has faced significant population declines over the past century due to habitat destruction driven by intensive agricultural practices. Outside of China, the species became critically endangered. However, collaborative conservation efforts between China and Japan have successfully brought the crested ibis back from the brink of extinction in the wild. This species primarily inhabits rice fields and riverbanks, feeding on aquatic organisms such as shrimp, crabs, frogs, and small fish found in submerged mud. Its diet is occasionally supplemented with plant-based foods, including rice, beans, buckwheat, grass seeds, and other vegetation [56].

The crested ibis is a tactile feeder with numerous tactile cells at the tip of its bill [57]. Its body structure makes it particularly adept at capturing smaller prey, as larger fish are often too difficult to handle and swallow. Furthermore, catching larger fish demands greater physical strength and skill, which poses significant challenges for a relatively small bird like the crested ibis. Attempting to capture large fish is complex and risky, as the bird may be struck or injured during the struggle.

Fig. 2. The pictures of crested ibis.

2.2. Inspiration

Our observations of the crested ibis' foraging behavior highlight its remarkable adaptability and sophisticated decision-making abilities, which surpass previous assumptions. Its foraging success is shaped by the dynamic interaction between predator (crested ibis) and prey (fish). As a predator, the crested ibis carefully evaluates factors such as the size of the prey and the difficulty of capture to maximize efficiency. When faced with a challenging target, it strategically abandons the current prey in favor of one that is easier to capture [58,59]. Conversely, fish, as prey, remain acutely aware of the predator's presence, exhibiting vigilance and maintaining a safe distance to evade capture [60]. This interaction exemplifies the balance between exploration and exploitation in natural systems, where predators continuously adapt their strategies in response to environmental feedback. Inspired by these behaviors, the proposed CIA incorporates similar decision-making processes to balance global exploration and local exploitation during optimization [61–63] (see Fig. 2).

2.3. Mathematical model

As previously described, the core concept of the proposed CIA is to simulate the predation behavior of the crested ibis hunting fish. The algorithm's flowchart is presented in Fig. 3, and its detailed procedural steps are outlined in Algorithm 1. The following sections explain each step of the CIA in detail:

Initialization

In this phase, CIA initializes the crested ibis population using Eq. (1) and the fish population using Eq. (2).

$$X=\begin{bmatrix}\boldsymbol{x_1}\\ \boldsymbol{x_2}\\ \boldsymbol{x_3}\\ \vdots\\ \boldsymbol{x_{N_1}}\end{bmatrix}=\begin{bmatrix}x_1^1 & x_1^2 & \cdots & x_1^D\\ x_2^1 & x_2^2 & \cdots & x_2^D\\ x_3^1 & x_3^2 & \cdots & x_3^D\\ \vdots & \vdots & \ddots & \vdots\\ x_{N_1}^1 & x_{N_1}^2 & \cdots & x_{N_1}^D\end{bmatrix} \tag{1}$$

$$x_i^j=\mathrm{lb}^j+r_1\cdot(\mathrm{ub}^j-\mathrm{lb}^j)$$

$$Y=\begin{bmatrix}\boldsymbol{y_1}\\ \boldsymbol{y_2}\\ \boldsymbol{y_3}\\ \vdots\\ \boldsymbol{y_{N_2}}\end{bmatrix}=\begin{bmatrix}y_1^1 & y_1^2 & \cdots & y_1^D\\ y_2^1 & y_2^2 & \cdots & y_2^D\\ y_3^1 & y_3^2 & \cdots & y_3^D\\ \vdots & \vdots & \ddots & \vdots\\ y_{N_2}^1 & y_{N_2}^2 & \cdots & y_{N_2}^D\end{bmatrix} \tag{2}$$

$$y_i^j=lb^j+r_2\cdot(ub^j-lb^j)$$

where D denotes the dimensionality of the optimization problem. x_i^j and y_i^j represent the jth component of the ith individual in populations of sizes N_1 and N_2, respectively. The terms r_1 and r_2 are independent random variables uniformly distributed over $(0,1)$. The vectors $\boldsymbol{lb}$ and $\boldsymbol{ub}$ define the lower and upper bounds of the search domain, with lb^j and ub^j indicating the bounds for the jth dimension.

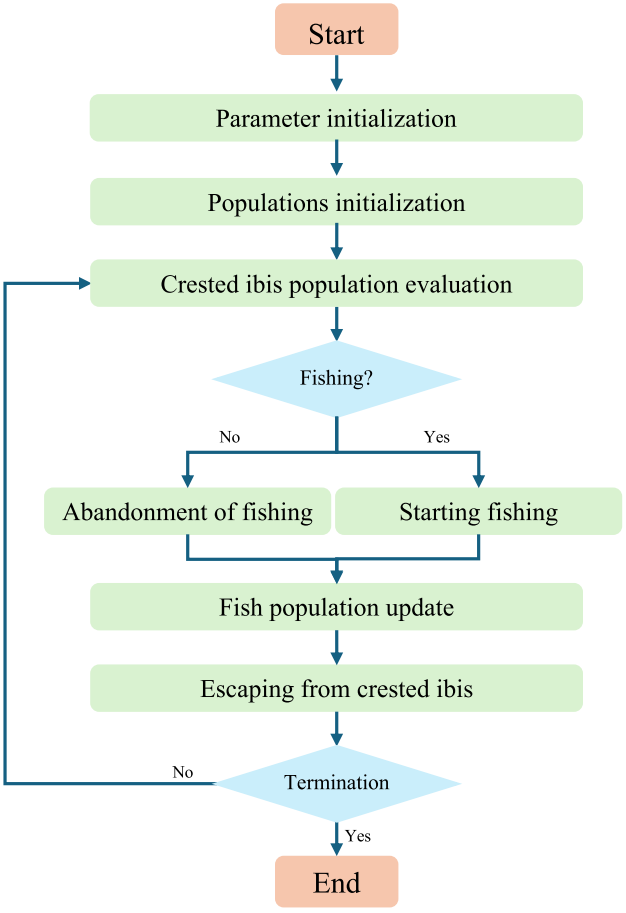

Fig. 3. The framework of proposed CIA.

Updating the population of crested ibis

As previously explained, the crested ibis evaluates a target fish based on its size and the capture difficulty. If the target fish exceeds the ibis' capabilities, the ibis abandons the attempt. Conversely, if the fish is

Algorithm 1 The pseudo-code of CIA.

Set the hyper-parameters, including crested ibis population size N_1, fish population size N_2, maximum number of evaluations t_{max}, and current number of evaluations $t = 0$.
Initialize the crested ibis population using Eq. (1) and initialize the fish population using Eq. (2).
while $t \leq t_{max}$ **do**
  Update crested ibis population:
  for $i = 1$ to N_1 **do**
    Use Eq. (4) calculate the adaptive factor.
    Select a fish randomly as the prey.
    Compare the fitness between the i-th individual and the prey fish.
    if the fitness of the i-th individual is worse **then**
      // Abandonment of fishing:
      Use Eq. (5) to select another prey.
      Use Eq. (6) to generate $\mathbf{x}_{new}$.
    else
      // Starting fishing:
      Use Eq. (7) to generate $\mathbf{x}_{new}$.
    end if
    Evaluate the fitness of individual $\mathbf{x}_{new}$ and compare it with the i-th individual of the crested ibis population. Take the better one to update the i-th individual of the crested ibis population.
  end for
  Update fish population:
  // Escaping from crested ibis:
  for $i = 1$ to N_2 **do**
    Use Eq. (8) to generate $\mathbf{y}_{new}$.
    Evaluate the fitness of individual $\mathbf{y}_{new}$ and compare it with the i-th individual of the fish population. Take the better one to update the i-th individual of the fish population.
  end for
  Increment the number of evaluations t.
end while
Select the best individual from the two populations as the optimal solution.
Output the optimal solution found.

deemed catchable, the ibis proceeds with the capture. To simulate this behavior, we randomly select a fish from the population as the target, denoted by $\boldsymbol{y_t}$. The fitness value of the target fish is then compared with that of the crested ibis to determine whether to abandon or proceed with the capture.

Adaptive factor I To effectively regulate the exploration range of the crested ibis population, we introduced an adaptive parameter defined by Eqs. (3) and (4). This parameter enables the exploration behavior of the crested ibis to dynamically adjust based on the evolutionary stage and the population's diversity.

$$div = \frac{1}{D} \cdot \sum_{k=1}^{D} \sqrt{\frac{1}{N_1} \sum_{i=1}^{N_1} \cdot (x_i^k - \bar{x}^k)^2} \tag{3}$$

$$I = (1 - \frac{Cur}{Max}) \cdot div \tag{4}$$

where $\bar{x}^k$ denotes the average value of the kth dimension across all crested ibis in the population. Cur represents the current iteration index, and Max refers to the total number of iterations.

Abandonment of fishing If the fitness of the target fish surpasses that of the crested ibis, Eq. (6) is employed to simulate the scenario where the crested ibis abandons its attempt to catch the target fish and moves on to pursue other prey, as defined by Eq. (5).

$$\boldsymbol{prey} = \begin{cases} \boldsymbol{x_{best}}, & \text{if} \quad \text{rand}(0,\ 1) < 0.5, \\ \boldsymbol{y_{best}}, & \text{otherwise} \end{cases} \tag{5}$$

$$\boldsymbol{x}_{new} = \boldsymbol{x}_i + r_3 \cdot (\boldsymbol{prey} - \boldsymbol{x_i}) \cdot \sin(2 \cdot \pi \cdot r_4) \cdot I \tag{6}$$

where $\boldsymbol{prey}$ refers to other prey of the crested ibis, $\boldsymbol{x}_{best}$ denotes the best-performing individual in the crested ibis population, $\boldsymbol{y}_{best}$ denotes the best-performing individual in the fish population, and r_3 and r_4 are independent random variables uniformly distributed over the range $(0, 1)$.

Starting fishing If the fitness of the target fish is inferior to that of the crested ibis, the ibis proceeds with the fishing attempt, which is modeled using Eq. (7).

$$x_{new}^k = \begin{cases} x_i^k + r_{\text{norm}} \cdot I, & \text{if} \quad \text{rand}\ (0,1) < 0.5, \\ y_t^k + r_{\ \text{norm}} \cdot I, & \text{otherwise} \end{cases} \tag{7}$$

where r_{norm} denotes a random variable sampled from a standard normal distribution $N(0, 1)$, and I is an adaptive factor as previously defined.

Update the population of fish

Escaping from crested ibis We use Eq. (8) to model the fleeing behavior of fish in the face of a predator.

$$\boldsymbol{y}_{new} = \boldsymbol{y}_{best} - C \cdot r_4 \cdot (\boldsymbol{x_{best}} - \boldsymbol{y_i}) + C \cdot r_5 \cdot (\boldsymbol{x_{rand}} - \boldsymbol{y_i}) \tag{8}$$

where C is a constant in the range $(0, 0.5)$, $\boldsymbol{x}_{rand}$ denotes a randomly selected individual from the crested ibis population, r_4 is a random variable uniformly distributed over the range $(-1, 1)$, and r_5 is a random variable uniformly distributed over the range $(0, 1)$.

Offspring selection

After each evolutionary iteration, the elite selection was employed to update the newly generated crested ibis and fish individuals. The update decision for each individual was based on a fitness comparison between the freshly generated individual and its corresponding original counterpart. If the new individual exhibited superior fitness, it replaced the original; otherwise, the original individual was retained. This strategy ensures that only the fittest individuals are propagated to subsequent generations, enhancing the population's overall quality over time.

3. Sensitivity analysis

To evaluate the influence of different parameters on the algorithm's performance, we conducted experiments using 50-dimensional (50D) functions from the CEC2020 benchmark suite. Parameters such as population size, proportion, and constant C were varied to assess their impact. The total number of function evaluations was 1000 times the problem dimension. Each function was executed 30 times using the same seed to ensure consistency, and the average performance across these runs was calculated. A comprehensive description of the test functions from the CEC2020 benchmark suite is provided in Table 11.

3.1. Population size N

To evaluate the impact of population size on the performance of the CIA, we tested seven different population sizes (N): 50, 100, 500, 1000, 2000, 5000, and 10000. The results, presented in Table 1 and Fig. 4, indicate that the overall performance is optimal when the population size N is set to 100. Consequently, $N = 100$ was selected as the default population size for subsequent experiments.

3.2. Constant C

To assess the impact of the constant C on the performance of the CIA, we tested five different values of C within the range $[0.1, 0.5]$. The results, presented in Table 2 and Fig. 5, indicate that the overall performance is optimal when C is set to 0.2. Consequently, $C = 0.2$ was selected as the default value for subsequent experiments.

Table 1
The result of sensitivity experiment of population size N.

Fun. \ N	50	100	500	1000	2000	5000	10000
$f1$	4.045×10^3	$\mathbf{2.772 \times 10^3}$	1.015×10^4	3.437×10^4	3.065×10^5	3.685×10^7	1.984×10^8
$f2$	3.033×10^5	$\mathbf{2.896 \times 10^5}$	1.849×10^6	6.721×10^6	4.517×10^7	4.489×10^9	2.373×10^{10}
$f3$	7.392×10^4	$\mathbf{6.586 \times 10^4}$	3.920×10^5	1.863×10^6	1.110×10^7	1.396×10^9	7.003×10^9
$f4$	1.917×10^3	$\mathbf{1.915 \times 10^3}$	1.919×10^3	1.924×10^3	1.931×10^3	1.947×10^3	1.953×10^3
$f5$	6.897×10^5	5.786×10^5	$\mathbf{5.183 \times 10^5}$	6.257×10^5	5.966×10^5	8.310×10^5	1.388×10^6
$f6$	2.446×10^4	1.373×10^4	1.271×10^4	8.694×10^3	$\mathbf{8.414 \times 10^3}$	1.881×10^4	2.553×10^4
$f7$	1.484×10^6	1.517×10^6	1.212×10^6	1.072×10^6	$\mathbf{7.696 \times 10^5}$	2.547×10^6	4.652×10^6
$f8$	2.738×10^3	2.666×10^3	$\mathbf{2.451 \times 10^3}$	2.462×10^3	2.481×10^3	2.490×10^3	2.518×10^3
$f9$	2.686×10^3	2.666×10^3	2.702×10^3	$\mathbf{2.665 \times 10^3}$	2.747×10^3	3.175×10^3	3.795×10^3
$f10$	$\mathbf{3.340 \times 10^3}$	3.358×10^3	3.373×10^3	3.400×10^3	3.499×10^3	3.489×10^3	3.647×10^3
Avg.Rank	3.4	**2.5**	2.7	3.1	3.9	5.6	6.8

Table 2
The result of sensitivity experiment of constant C.

Fun. \ C	0.1	0.2	0.3	0.4	0.5
f_1	3.568×10^3	$\mathbf{3.307 \times 10^3}$	4.098×10^3	4.118×10^3	5.178×10^3
f_2	2.442×10^5	$\mathbf{1.550 \times 10^5}$	2.860×10^5	3.196×10^5	3.453×10^5
f_3	6.480×10^4	$\mathbf{2.888 \times 10^4}$	5.524×10^4	1.061×10^5	4.551×10^4
f_4	1.923×10^3	1.916×10^3	1.915×10^3	$\mathbf{1.914 \times 10^3}$	1.915×10^3
f_5	$\mathbf{4.541 \times 10^5}$	5.361×10^5	6.074×10^5	7.423×10^5	7.500×10^5
f_6	1.329×10^4	1.689×10^4	1.473×10^4	1.199×10^4	$\mathbf{9.430 \times 10^3}$
f_7	$\mathbf{1.044 \times 10^6}$	1.503×10^6	1.798×10^6	1.715×10^6	1.941×10^6
f_8	3.059×10^3	2.545×10^3	2.441×10^3	2.434×10^3	$\mathbf{2.428 \times 10^3}$
f_9	2.740×10^3	2.684×10^3	2.684×10^3	2.659×10^3	$\mathbf{2.640 \times 10^3}$
f_{10}	3.372×10^3	3.350×10^3	3.335×10^3	$\mathbf{3.323 \times 10^3}$	3.346×10^3
Aver.Rank	3.3	**2.7**	3.1	2.8	3.1

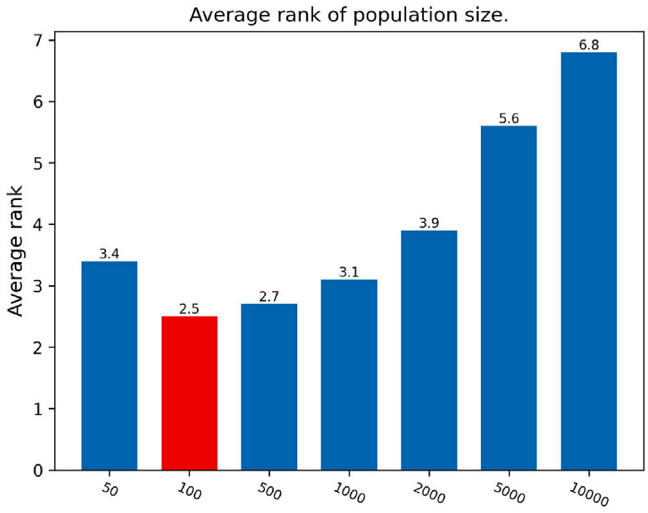

Fig. 4. The average rank of the population size N.

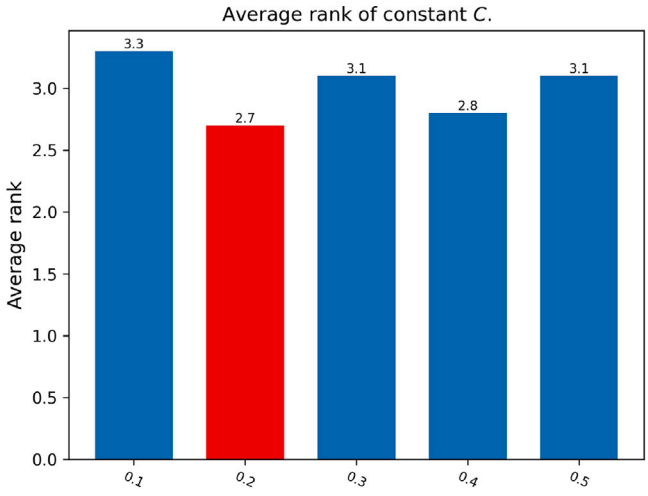

Fig. 5. The average rank of results on constant C.

3.3. Population proportion

The CIA algorithm utilizes two populations with user-defined sizes, and varying the population sizes significantly impacts the algorithm's exploration and exploitation capabilities. To achieve a more balanced performance with a constant C value of 0.2, we evaluated 11 different population proportions. The performance outcomes for each configuration with varying population sizes are presented in Table 3 and Fig. 6.

The results indicate that the CIA algorithm achieves superior performance within the population proportion interval of 40/60 to 60/40. This interval was subdivided to refine the analysis further, and additional experiments were conducted by varying the population size 1. The detailed outcomes of these experiments are presented in Table 4 and Fig. 6.

The results reveal that the performance of the CIA algorithm does not exhibit significant variation within the population proportion interval of 40/60 to 60/40. Consequently, the configuration with a population of 60 ibises and 40 fish were selected for subsequent experiments to ensure consistency and computational efficiency.

3.4. Runtime of the algorithm

To accurately compare the runtimes of the proposed CIA algorithm with those of mainstream optimization algorithms, we selected fifteen widely recognized algorithms, including DE, PSO, CMA-ES,

Table 3
The result of sensitivity experiment of population proportion (1/99 to 99/1).

Fun. \ Pop.	1/99	10/90	20/80	30/70	40/60	50/50	60/40	70/30	80/20	90/10	99/1
f_1	6.832×10^{10}	2.392×10^{4}	3.791×10^{3}	2.719×10^{3}	3.134×10^{3}	3.602×10^{3}	2.683×10^{3}	$\mathbf{2.141\times10^{3}}$	3.547×10^{3}	5.480×10^{4}	4.586×10^{8}
f_2	7.715×10^{12}	3.126×10^{6}	3.337×10^{5}	2.264×10^{5}	2.848×10^{5}	2.719×10^{5}	$\mathbf{2.012\times10^{5}}$	3.189×10^{5}	4.241×10^{5}	1.102×10^{7}	5.548×10^{10}
f_3	2.479×10^{12}	9.030×10^{5}	9.904×10^{4}	7.213×10^{4}	$\mathbf{3.696\times10^{4}}$	4.492×10^{4}	4.768×10^{4}	5.610×10^{4}	7.660×10^{4}	2.421×10^{6}	1.625×10^{10}
f_4	9.911×10^{5}	1.935×10^{3}	1.923×10^{3}	1.919×10^{3}	1.919×10^{3}	1.918×10^{3}	$\mathbf{1.915\times10^{3}}$	1.916×10^{3}	1.919×10^{3}	1.922×10^{3}	1.935×10^{3}
f_5	6.270×10^{7}	2.126×10^{6}	7.854×10^{5}	6.445×10^{5}	6.037×10^{5}	5.268×10^{5}	5.583×10^{5}	$\mathbf{4.951\times10^{5}}$	5.335×10^{5}	7.235×10^{5}	2.214×10^{6}
f_6	2.308×10^{8}	2.816×10^{4}	1.958×10^{4}	1.554×10^{4}	1.638×10^{4}	$\mathbf{1.523\times10^{4}}$	1.806×10^{4}	2.074×10^{4}	2.236×10^{4}	5.197×10^{4}	4.929×10^{5}
f_7	6.894×10^{8}	3.703×10^{6}	2.055×10^{6}	1.814×10^{6}	1.635×10^{6}	1.374×10^{6}	$\mathbf{1.290\times10^{6}}$	1.468×10^{6}	1.345×10^{6}	2.348×10^{6}	5.965×10^{6}
f_8	1.214×10^{4}	2.716×10^{3}	2.581×10^{3}	2.736×10^{3}	$\mathbf{2.452\times10^{3}}$	2.862×10^{3}	2.530×10^{3}	2.945×10^{3}	2.497×10^{3}	2.462×10^{3}	2.519×10^{3}
f_9	5.448×10^{4}	2.763×10^{3}	2.725×10^{3}	2.670×10^{3}	2.693×10^{3}	2.673×10^{3}	2.690×10^{3}	$\mathbf{2.650\times10^{3}}$	2.674×10^{3}	2.679×10^{3}	4.503×10^{3}
f_{10}	1.232×10^{4}	3.444×10^{3}	3.388×10^{3}	3.353×10^{3}	3.390×10^{3}	3.344×10^{3}	$\mathbf{3.343\times10^{3}}$	3.362×10^{3}	3.365×10^{3}	3.419×10^{3}	4.123×10^{3}
Rank	11.0	8.4	6.8	4.3	4.2	3.4	**2.8**	3.8	4.6	7.3	9.4

Table 4
The result of sensitivity experiment of population proportion (41/59 to 59/41).

Fun. \ Prop.	41/59	42/58	43/57	44/56	45/55	46/54	47/53	48/52	49/51
f_1	2.386×10^{3}	2.499×10^{3}	2.892×10^{3}	2.909×10^{3}	$\mathbf{2.053\times10^{3}}$	2.209×10^{3}	2.850×10^{3}	2.742×10^{3}	2.515×10^{3}
f_2	2.296×10^{5}	2.342×10^{5}	3.770×10^{5}	4.286×10^{5}	2.478×10^{5}	3.770×10^{5}	3.016×10^{5}	2.425×10^{5}	3.693×10^{5}
f_3	5.878×10^{4}	$\mathbf{3.701\times10^{4}}$	4.058×10^{4}	4.342×10^{4}	4.303×10^{4}	6.903×10^{4}	5.636×10^{4}	4.143×10^{4}	5.081×10^{4}
f_4	1.917×10^{3}	1.919×10^{3}	1.918×10^{3}	1.918×10^{3}	1.919×10^{3}	1.918×10^{3}	1.917×10^{3}	1.920×10^{3}	1.921×10^{3}
f_5	5.056×10^{5}	5.535×10^{5}	5.302×10^{5}	5.659×10^{5}	5.525×10^{5}	5.263×10^{5}	5.721×10^{5}	5.220×10^{5}	$\mathbf{4.875\times10^{5}}$
f_6	1.561×10^{4}	1.539×10^{4}	1.487×10^{4}	1.618×10^{4}	$\mathbf{1.469\times10^{4}}$	1.653×10^{4}	1.589×10^{4}	1.484×10^{4}	1.629×10^{4}
f_7	1.461×10^{6}	1.252×10^{6}	1.370×10^{6}	1.449×10^{6}	1.318×10^{6}	1.305×10^{6}	1.571×10^{6}	1.456×10^{6}	$\mathbf{1.221\times10^{6}}$
f_8	$\mathbf{2.445\times10^{3}}$	2.460×10^{3}	2.594×10^{3}	2.555×10^{3}	2.689×10^{3}	2.691×10^{3}	2.537×10^{3}	2.527×10^{3}	2.598×10^{3}
f_9	2.721×10^{3}	2.675×10^{3}	2.685×10^{3}	2.678×10^{3}	2.645×10^{3}	2.674×10^{3}	2.653×10^{3}	2.642×10^{3}	2.704×10^{3}
f_{10}	3.358×10^{3}	3.357×10^{3}	$\mathbf{3.341\times10^{3}}$	3.349×10^{3}	3.382×10^{3}	3.346×10^{3}	3.369×10^{3}	3.360×10^{3}	3.376×10^{3}
Rank	8.9	**7.1**	8.9	11.8	8.4	10.6	11.5	7.8	11.4

Fun. \ Prop.	50/50	51/49	52/48	53/47	54/46	55/45	56/44	57/43	58/42	59/41
f_1	3.602×10^{3}	3.099×10^{3}	2.564×10^{3}	2.592×10^{3}	2.937×10^{3}	3.123×10^{3}	3.933×10^{3}	3.073×10^{3}	2.139×10^{3}	3.425×10^{3}
f_2	2.719×10^{5}	2.324×10^{5}	3.447×10^{5}	2.870×10^{5}	2.810×10^{5}	3.830×10^{5}	$\mathbf{2.205\times10^{5}}$	2.753×10^{5}	3.578×10^{5}	3.077×10^{5}
f_3	4.492×10^{4}	4.418×10^{4}	5.451×10^{4}	4.463×10^{4}	4.368×10^{4}	4.152×10^{4}	5.055×10^{4}	6.425×10^{4}	5.783×10^{4}	4.781×10^{4}
f_4	1.918×10^{3}	1.918×10^{3}	1.917×10^{3}	1.916×10^{3}	1.917×10^{3}	1.918×10^{3}	1.918×10^{3}	1.917×10^{3}	$\mathbf{1.916\times10^{3}}$	1.917×10^{3}
f_5	5.268×10^{5}	5.384×10^{5}	5.580×10^{5}	5.261×10^{5}	5.124×10^{5}	5.008×10^{5}	5.465×10^{5}	5.755×10^{5}	5.354×10^{5}	5.552×10^{5}
f_6	1.523×10^{4}	1.540×10^{4}	1.603×10^{4}	1.668×10^{4}	1.589×10^{4}	1.497×10^{4}	1.768×10^{4}	1.540×10^{4}	1.774×10^{4}	1.726×10^{4}
f_7	1.374×10^{6}	1.255×10^{6}	1.397×10^{6}	1.578×10^{6}	1.349×10^{6}	1.282×10^{6}	1.448×10^{6}	1.334×10^{6}	1.342×10^{6}	1.356×10^{6}
f_8	2.862×10^{3}	2.693×10^{3}	2.462×10^{3}	2.630×10^{3}	2.519×10^{3}	2.503×10^{3}	2.636×10^{3}	2.708×10^{3}	2.630×10^{3}	2.453×10^{3}
f_9	2.673×10^{3}	2.665×10^{3}	2.696×10^{3}	2.677×10^{3}	2.660×10^{3}	2.661×10^{3}	2.699×10^{3}	2.688×10^{3}	2.717×10^{3}	$\mathbf{2.625\times10^{3}}$
f_{10}	3.344×10^{3}	3.356×10^{3}	3.356×10^{3}	3.360×10^{3}	3.382×10^{3}	3.375×10^{3}	3.352×10^{3}	3.360×10^{3}	3.353×10^{3}	3.363×10^{3}
Rank	9.8	9.5	10.4	10.7	9.0	8.9	12.1	12.1	10.6	10.5

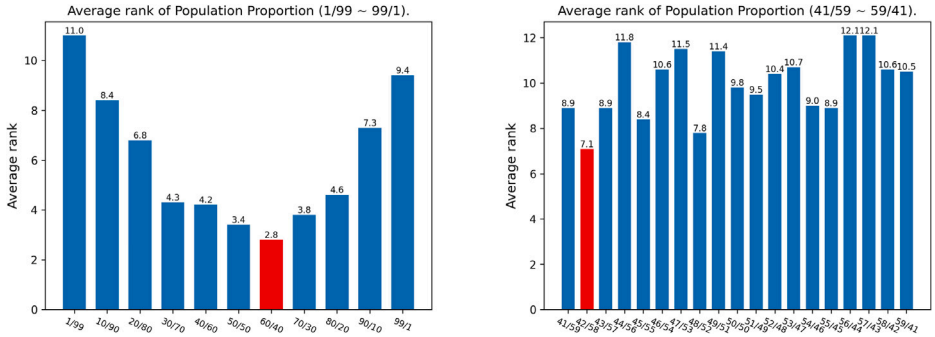

Fig. 6. The average rank of the population proportion.

WOA, MBGO, CL-PSO, SaDE, JADE, L-SHADE, PPSO, COA, FATA, HO, and MSA. These algorithms were evaluated alongside CIA through average runtime experiments conducted on the same computational platform. The experimental device had an AMD Ryzen Threadripper PRO 7975WX 4.0 GHz CPU and 64 GB of memory. The results of the runtime comparison are presented in Table 5 and Fig. 7. Furthermore, these algorithms will be utilized in subsequent comparative experiments.

4. Evaluation experiments on CEC benchmark suites

4.1. Experimental setup

To evaluate the performance of the proposed CIA algorithm, we compared it against fifteen evolutionary algorithms. These include four classical optimization algorithms widely recognized in the field: DE, PSO, CMA-ES, and WOA. Five well-established improved variants were

Table 5
Average runtime of CIA and other algorithms in 30 times. The values in the table are results obtained in unit measurement of seconds.

FunNum	DE	PSO	CMA-ES	CL-PSO	SaDE	JADE	L-SHADE
$f1$	1.80	2.06	39.10	5.84	2.01	4.64	5.02
$f2$	1.83	2.14	38.39	5.95	2.07	4.20	5.17
$f3$	1.82	2.14	39.10	6.01	2.05	4.13	5.19
$f4$	23.78	26.76	61.94	27.82	23.34	39.82	36.40
$f5$	4.76	4.37	43.15	9.08	4.98	8.96	9.25
$f6$	7.48	9.54	46.08	11.86	7.62	13.02	13.02
$f7$	6.51	7.73	44.99	11.09	6.83	9.16	9.03
$f8$	8.51	11.14	46.70	12.57	8.83	11.15	11.02
$f9$	9.57	10.26	44.19	10.89	7.12	9.39	9.23
$f10$	7.82	5.24	42.30	9.65	5.87	8.03	7.95
FunNum	WOA	PPSO	MBGO	COA	FATA	HO	MSA	CIA
$f1$	**1.16**	1.92	6.72	13.02	3.78	7.20	4.10	2.91
$f2$	**1.24**	2.23	5.94	8.96	12.38	1.26	4.11	3.61
$f3$	**1.25**	2.03	5.81	11.09	21.09	4.20	4.99	3.43
$f4$	**22.90**	32.17	42.27	26.68	31.91	30.07	23.76	24.32
$f5$	**3.98**	5.40	12.75	4.82	5.55	32.61	7.08	5.37
$f6$	**6.68**	9.36	8.90	26.38	8.14	14.20	8.57	8.40
$f7$	**5.81**	7.94	8.24	7.85	6.84	8.69	6.83	7.66
$f8$	**7.79**	9.66	10.01	17.56	9.80	8.80	9.64	9.86
$f9$	**6.27**	6.84	8.74	6.33	7.80	8.30	8.89	9.31
$f10$	5.03	**5.01**	7.60	9.86	6.26	7.03	5.79	7.37

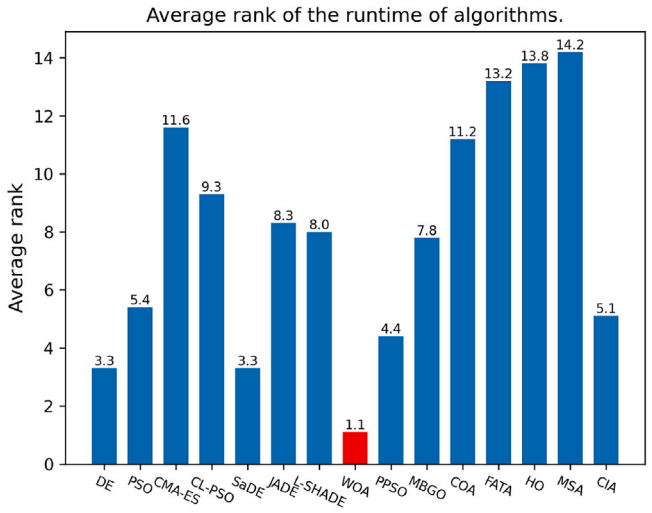

Fig. 7. The runtime of CIA and other algorithms.

included: CL-PSO, SaDE, JADE, L-SHADE, and PPSO. Furthermore, five state-of-the-art algorithms proposed within the last two years: FATA, MSA, HO, COA, and MBGO were evaluated. To ensure the accuracy and robustness of the assessment, we utilized two benchmark suites: CEC2017 and CEC2022.

For a comprehensive evaluation, each benchmark suite was tested across multiple dimensions. Specifically, the CEC2017 suite was evaluated using 10, 30, 50, and 100 dimensions, while the CEC2022 suite was tested with 10 and 20 dimensions. The experiments were conducted using the algorithm library developed by Nguyen Van Thieu [64, 65] on the same computational platform as the sensitivity tests.

Table 6 summarizes the parameter settings for the CIA and the comparison algorithms. The population size for each algorithm was set to 100, with the CIA's population size being the sum of its two subpopulations. The maximum number of evaluations was fixed at 1000 times the dimensionality of each benchmark function. Each benchmark function was evaluated 30 times using different random seeds to ensure robustness and accuracy. The average results were calculated, and statistical tests were conducted to analyze performance differences. Specifically, the Freedman test was first employed to detect significant differences among the algorithms. If differences were identified,

Table 6
Parameter setting of the model in the experiment.

Algorithms	Parameters	Values
DE	F	0.7
	C_r	0.9
PSO	ω	0.4
	C_1	2.05
	C_2	2.05
CMA-ES	σ	0.75
CL-PSO	C	1.2
	F_{max}	7
	ω_{max}, ω_{min}	0.9, 0.4
SaDE	Parameter free	
JADE	μ_f, μ_{cr}	0.5, 0.5
	σ_f, σ_{cr}	0.1, 0.1
	p	0.1
L-SHADE	μ_f, μ_{cr}	0.5, 0.5
	σ_f, σ_{cr}	0.1, 0.1
	Initial population size	$18 \times D$
WOA	a	0 to 2
	a_2	−2 to −1
PPSO	Parameter free	
MBGO	α	0.8 to 1.2
COA	Parameter free	
FATA	α	0.2
HO	Parameter free	
MSA	α	0.8 to 1.2
	p	0.5
	P_c	0.2
	A	30
	a	2
CIA	N_1	60
	N_2	40
	C	0.2

pairwise comparisons were conducted using the U-test. The resulting p-values were corrected using Holm's multiple comparison method to determine statistically significant algorithm differences.

4.2. CEC2017 benchmark suite

We evaluated the performance of the proposed CIA algorithm and compared it with other algorithms using the CEC2017 benchmark suite. Basic information about the benchmark suite is provided in Appendix A, Table 10. The experimental parameters were identical to those described in Section 4.1. The results of the experiments for 10, 30, 50, and 100 dimensions are presented in Table 7, which includes the CIA's performance relative to competing algorithms and the average rankings across the benchmark suite. For a more precise visualization, the average rankings of each algorithm are depicted using bar charts in Fig. 8.

To provide detailed insights, we have included the performance of each algorithm for all test functions in the benchmark suite in Appendix B. This includes the exploration and exploitation dynamics of the CIA during its evolution. Specific details are provided in Tables 13, 14, 15, and 16, and Figs. 13 through 32.

In the experimental results table, the symbols "+", "=", and "−" indicate the following:

- "+": The CIA algorithm significantly outperforms the comparison algorithm.
- "=": No significant performance difference is observed between the CIA and the comparison algorithm.
- "−": The CIA algorithm's performance is significantly worse than the comparison algorithm's.

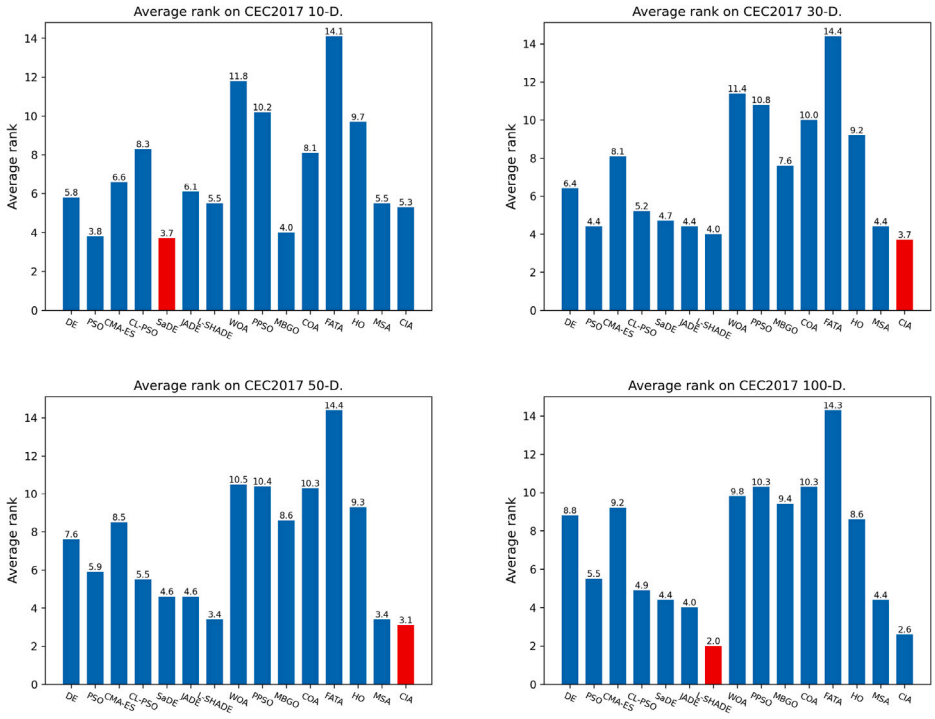

Fig. 8. The average rank on CEC2017.

Table 7
The summarize results on CEC2017.

Dim.	Item name	DE	PSO	CMA-ES	CL-PSO	SaDE	JADE	L-SHADE
10D	+ /=/−	8/10/11	4/21/4	15/7/17	17/9/3	6/13/10	4/21/4	5/21/3
	Avg. Rank	5.8	3.8	6.6	8.3	**3.7**	6.1	5.5
30D	+ /=/−	13/9/7	13/10/6	22/0/6	17/5/7	13/9/7	4/19/6	4/20/6
	Avg. Rank	6.4	4.4	8.1	5.2	4.7	4.4	4.0
50D	+ /=/−	18/9/2	16/5/8	23/0/6	19/3/7	16/5/8	4/20/5	6/15/8
	Avg. Rank	7.6	5.9	8.5	5.5	4.6	4.6	3.4
100D	+ /=/−	23/3/2	18/2/9	24/0/5	18/3/8	16/4/9	6/15/8	6/12/11
	Avg. Rank	8.8	5.5	9.2	4.9	4.4	4.0	**2.0**
Dim.	Item name	WOA	PPSO	MBGO	COA	FATA	HO	MSA	CIA
10D	+ /=/−	19/10/0	15/14/0	7/16/6	14/10/4	28/0/1	8/12/9	11/14/4	–
	Avg. Rank	11.8	10.2	4.0	8.1	14.1	9.7	5.5	5.3
30D	+ /=/−	23/5/1	24/5/0	20/8/1	24/2/3	29/0/0	17/4/8	12/8/9	–
	Avg. Rank	11.4	10.8	7.6	10.0	14.4	9.2	4.4	**3.7**
50D	+ /=/−	25/2/2	25/3/1	25/3/1	26/0/3	29/0/0	24/4/1	13/7/9	–
	Avg. Rank	10.5	10.4	8.6	10.3	14.4	9.3	3.4	**3.1**
100D	+ /=/−	25/4/0	27/2/0	29/0/0	28/0/1	29/0/0	26/1/2	17/4/8	–
	Avg. Rank	9.8	10.3	9.4	10.3	14.3	8.6	4.4	2.6

For each test function, the optimal solutions identified by all algorithms are highlighted in bold.

4.3. CEC2022 benchmark suite

Table 12 provides an overview of the CEC2022 benchmark suite. Additional details about the benchmark suite are available in Appendix A, Table 12. The performance of the proposed CIA algorithm compared with other algorithms under 10 and 20 dimensions is summarized in Fig. 9 and Table 8, respectively. Detailed test results, along with exploration and development trend graphs, are presented in Tables 17, 18, and Figs. 33, 34, 35, and 36 in Appendix C. The symbols and meanings in these tables are consistent with those used in the CEC2017 benchmark analysis.

Table 8
The summarize results on CEC2022.

Dim.	Item name	DE	PSO	CMA-ES	CL-PSO	SaDE	JADE	L-SHADE
10D	+ /=/−	5/5/2	2/8/2	6/5/1	6/5/1	4/4/4	5/3/4	3/5/4
	Avg. Rank	8.2	4.1	7.6	8.3	**3.7**	6.3	5.4
20D	+ /=/−	6/5/1	8/4/0	9/1/2	10/0/2	8/1/3	4/5/3	5/4/3
	Avg. Rank	8.4	6.4	8.3	7.9	4.9	5.8	5.8
Dim.	Item name	WOA	PPSO	MBGO	COA	FATA	HO	MSA	CIA
10D	+ /=/−	9/3/0	8/4/0	0/8/4	8/3/1	11/1/0	7/4/1	8/2/2	–
	Avg. Rank	12.8	10.7	3.8	9.3	14.6	9.8	8.6	6.8
20D	+ /=/−	9/3/0	9/3/0	4/4/4	8/3/1	11/1/0	8/3/1	9/0/3	–
	Avg. Rank	11.9	10.5	4.3	9.8	14.4	9.7	8.0	**3.8**

5. Human-powered aircraft design

5.1. Definition of HPA design

Human-powered airplanes (HPAs) are unique flying machines propelled solely by human power through a pedal mechanism similar to a bicycle (Fig. 10). These aircraft are designed to be exceptionally lightweight, featuring a large wingspan to maximize lift and enable sustained flight. Wing optimization is a critical aspect of HPA design, as it directly impacts the aircraft's ability to generate sufficient lift while minimizing the pilot's required physical effort. As depicted in Fig. 11, typical HPAs have wingspans ranging from 20 to 35 m, comparable to commercial airliners. The wings are generally constructed from foam and balsa wood, covered with heat-shrink film, and supported by a carbon fiber-reinforced plastic (CFRP) tube frame. This optimization problem focuses on the design of the wings, assuming the use of these materials to ensure the aircraft achieves the necessary lift and overall performance.

When designing the wing, the shape is primarily defined by the following variables: n is defined, resulting in $n+1$ cross-sections. Fig. 11 shows an example with $n = 4$. There are five variables in total:

- a_i: represents the choice of airfoil.

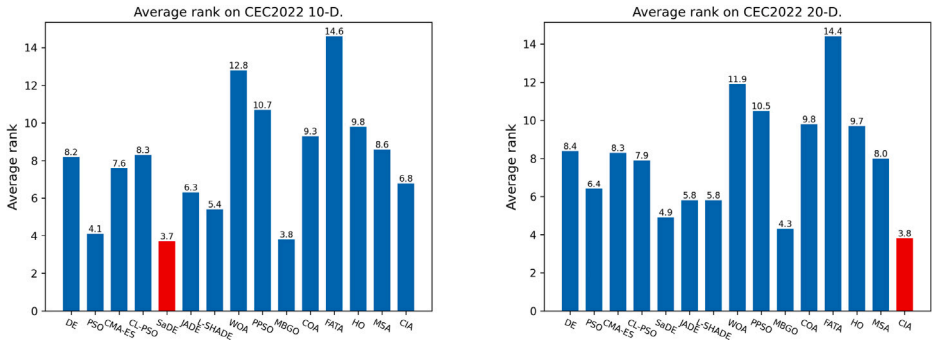

Fig. 9. The average rank on CEC2022.

Fig. 10. Human-powered aircraft.

- b_i: represents the length of each segment.
- c_i: represents the chord length.
- α_i: represents the angle of attack.
- d_i: represents the diameter of the CFRP tubes that make up the main wing.

These variables are all denoted by R. The definition of the HPA optimization problems is shown as follows:

$$\text{HPA101:}\quad f(x) = D(x)$$

$$\text{HPA102:}\quad f(x) = P(x) = \frac{D(x)V(x)}{\eta}$$

$$\text{HPA103:}\quad f(x) = -V(x)$$

$$\begin{aligned}\text{HPA131:}\quad & f(x) = D(x)\\ & g_1(x) = \frac{n_m n_s \epsilon(x)}{\epsilon_u} - 1\\ & g_2(x) = B\left(\sin\gamma(x) - \sin\gamma_u\right)/2\\ & g_3(x) = -\delta_{\text{park}}(x)\end{aligned}$$

$$\text{HPA142:}\quad f(x) = P(x) = \frac{D(x)V(x)}{\eta}$$

$$\begin{aligned} & g_1(x) = \frac{n_m n_s \epsilon(x)}{\epsilon_u} - 1\\ & g_2(x) = 1 - \left(\frac{V(x)}{V_{\min}}\right)^3\end{aligned}$$

$$\begin{aligned}\text{HPA143:}\quad & f(x) = -V(x)\\ & g_1(x) = \frac{n_m n_s \epsilon(x)}{\epsilon_u} - 1\\ & g_2(x) = B\left(\sin\gamma(x) - \sin\gamma_u\right)/2\\ & g_3(x) = -\delta_{\text{park}}(x)\\ & g_4(x) = P(x) - P_{\max}\end{aligned}$$

HPA101, HPA102, and HPA103 represent unconstrained optimization problems, while HPA131, HPA142, and HPA143 are their constrained counterparts. In the equations, $P(x)$, $D(x)$, $E(x)$, and $B(x)$ denote the power, drag, cruise speed, wing efficiency, and wingspan, respectively. The variables σ and σ_{park} represent the wing tip displacement relative to the wing root height during flight and parking conditions, respectively. $\epsilon_{\max}$ indicates the maximum strain experienced by the CFRP ducts under flight and parking conditions, while γ denotes the dihedral angle of the wing tip during flight.

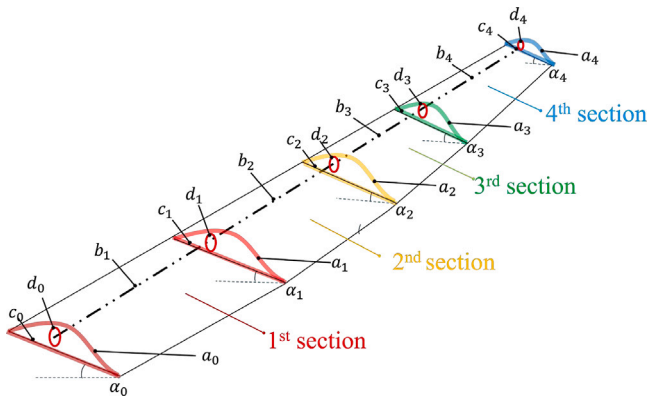

Fig. 11. The wing of Human-powered aircraft.

Table 9
The summarize results on HPA.

Dim.	Item name	DE	PSO	CMA-ES	CL-PSO	SaDE	JADE	L-SHADE	
Cons.	+ /=/−	25/2/0	19/8/0	18/9/0	27/0/0	17/10/0	8/15/4	15/11/1	
	Avg. Rank	7.1	4.4	5.7	11.8	6.2	3.1	4.8	
Uncons.	+ /=/−	25/2/0	19/8/0	18/7/2	27/0/0	18/8/1	15/10/2	16/8/3	
	Avg. Rank	6.7	4.3	5.7	12.3	6.1	2.9	4.5	
Dim.	Item name	WOA	PPSO	MBGO	COA	FATA	HO	MSA	CIA
Cons.	+ /=/−	17/9/1	11/13/3	17/9/1	18/9/0	27/0/0	21/6/0	18/9/0	–
	Avg. Rank	9.0	4.3	5.4	10.0	14.0	8.3	9.1	**1.7**
Uncons.	+ /=/−	18/9/0	16/11/0	23/4/0	18/3/6	27/0/0	24/3/0	18/9/0	–
	Avg. Rank	9.7	5.5	6.1	7.1	14.0	8.6	9.9	**1.6**

The overall system efficiency, η, is calculated as the product of the propeller efficiency (0.85) and the drive train efficiency (0.95). The load factor, n_m, is set to 1.5, while the safety factor, n_s, is 2. The reference strain variable, ϵ_u, is specified as 0.0027, and the maximum dihedral angle, γ_u is 8°. Additionally, the maximum power $P_{\max}$ is 400 [W]. The minimum cruise speed $V_{\min}$ is 7.3 [m/s].

5.2. The result of HPA

To evaluate the actual performance of the proposed CIA algorithm in the context of HPA, we used the parameter settings for all algorithms as listed in Table 6, with the only modification being the maximum number of evaluations, which was set to 1000. The results are presented in Table 9 and Fig. 12. Each function was evaluated 30 times using different random seeds to ensure robustness and accuracy. The average results were calculated, and a statistical analysis was performed to identify significant differences. Specifically, the Freedman test was first employed to detect significant differences among the algorithms. If differences were identified, pairwise comparisons were conducted using the U-test. Holm's multiple comparison method was then applied to adjust the p-values obtained from the U-test, ensuring reliable identification of statistically significant differences between algorithms.

Additionally, the performance of each algorithm on individual test functions for the HPA problem, along with the exploration and exploitation dynamics of the CIA during its evolution, is provided in Appendix C. Detailed results can be found in Tables 19 and 20.

6. Analysis and discussion

6.1. Computational complexity

For better understanding, the maximum number of fitness evaluations is defined as T, and we continue to use the previous notation in Section 2 for the rest of the symbols. Thus, the complexity of the core operations of the CIA algorithm is as follows:

- Population initialization: $O(N \times D)$.
- Selecting current best individual in the population: $O(N)$.
- Calculation for adaptive factor I: $O(N \times D)$.
- Update for two populations: $O(N \times D)$.

Therefore, the computational complexity of CIA can be calculated using Eq. (9).

$$\begin{aligned} &O(N \times D + T \times (N + N \times D + N \times D)) \\ =&O(N \times D + T \times N(1 + 2 \times D)) \\ =&O(N \times D + T \times N \times D) \\ =&O(T \times N \times D) \end{aligned} \tag{9}$$

The runtime experiments presented in Section 3.4 highlight the CIA algorithm's lower computational complexity. The CIA demonstrates superior efficiency compared to classical algorithms such as PSO, CMA-ES, and MBGO. Additionally, it significantly outperforms high-performing algorithms such as L-SHADE and JADE regarding execution speed.

6.2. Superiority of the core structure

The proposed CIA algorithm incorporates two distinct populations: the crested ibis and fish populations. Each population evolves independently through unique mechanisms while facilitating each other's evolution via information exchange. This synergistic interaction is a crucial factor contributing to the CIA algorithm's superior performance compared to competing algorithms. Additionally, the algorithm employs an adaptive factor to balance exploration and exploitation dynamically, enhancing its effectiveness. Below, we elaborate on the components responsible for the significant performance improvements achieved by this architecture.

Traditional EAs typically comprise a single population that evolves based on a uniform evolutionary strategy. Due to this inherent characteristic, where all individuals in the population follow the same approach, the algorithm is prone to becoming trapped in regions of non-optimal solutions, often leading to convergence at a local optimum. In contrast, the CIA algorithm employs two distinct populations governed by entirely different evolutionary strategies. This design prevents both populations from converging to the same region, significantly reducing the likelihood of falling into local optima. Furthermore, even if one population becomes trapped in a local optimum, the other population can assist it in escaping through information exchange. This mechanism enables a more comprehensive exploration of the solution space and minimizes the risk of stagnation.

Moreover, an adaptive factor is introduced to dynamically regulate the balance between exploration and exploitation within the populations. This mechanism enables the algorithm to adjust the balance at different stages of evolution, ensuring adaptability throughout the process. Additionally, when individuals within a population become excessively convergent or overly dispersed, the adaptive factor mitigates these extremes. Preventing over-convergence or over-dispersion maintains an optimal balance between exploration and exploitation, thereby enhancing the algorithm's overall efficiency and robustness.

6.3. Analysis of exploration and exploitation

Since balancing exploration and exploitation is the cornerstone of evolutionary algorithms, the crested ibis population in the CIA algorithm was designed to emphasize global exploration. In contrast, the fish population was tailored to prioritize local exploration. However, this design does not imply that the crested ibis population performs exclusively global exploration or that the fish population is restricted solely to local exploration.

For the crested ibis's "abandonment of fishing" strategy, we designed two scenarios for the "prey". In the first scenario, the approach

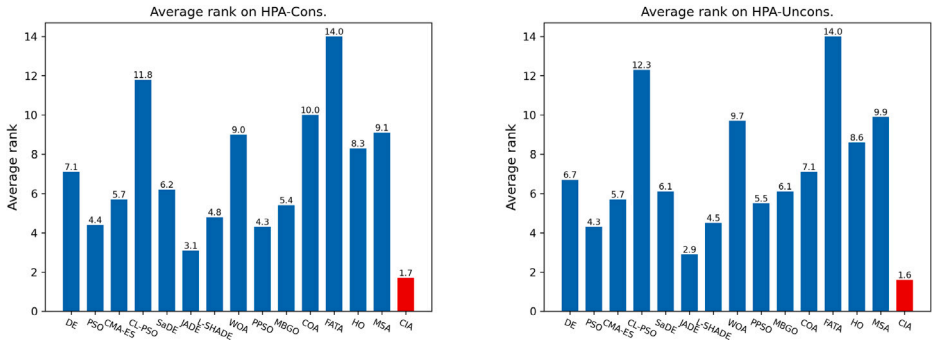

Fig. 12. The rank on HPA.

Table 10
The description of CEC2017 benchmark suite.

No.	Function name	Optimal solution	Type
f_1	Shifted and Rotated Bent Cigar Function	100	Unimodal Functions
f_3	Shifted and Rotated Zakharov Function	300	
f_4	Shifted and Rotated Rosenbrock's Function	400	Simple Multimodal Functions
f_5	Shifted and Rotated Rastrigin's Function	500	
f_6	Shifted and Rotated Expanded Scaffer's F6 Function	600	
f_7	Shifted and Rotated Lunacek BiRastrigin Function	700	
f_8	Shifted and Rotated Non-Continuous Rastrigin's Function	800	
f_9	Shifted and Rotated Levy Function	900	
f_{10}	Shifted and Rotated Schwefel's Function	1000	
f_{11}	Hybrid Function 1 (N=3)	1100	Simple Multimodal Functions
f_{12}	Hybrid Function 2 (N=3)	1200	
f_{13}	Hybrid Function 3 (N=3)	1300	
f_{14}	Hybrid Function 4 (N=4)	1400	
f_{15}	Hybrid Function 5 (N=4)	1500	
f_{16}	Hybrid Function 6 (N=4)	1600	
f_{17}	Hybrid Function 6 (N=5)	1700	
f_{18}	Hybrid Function 6 (N=5)	1800	
f_{19}	Hybrid Function 6 (N=5)	1900	
f_{20}	Hybrid Function 6 (N=6)	2000	
f_{21}	Composition Function 1 (N=3)	2100	Composition Functions
f_{22}	Composition Function 2 (N=3)	2200	
f_{23}	Composition Function 3 (N=4)	2300	
f_{24}	Composition Function 4 (N=4)	2400	
f_{25}	Composition Function 5 (N=5)	2500	
f_{26}	Composition Function 6 (N=5)	2600	
f_{27}	Composition Function 7 (N=6)	2700	
f_{28}	Composition Function 8 (N=6)	2800	
f_{29}	Composition Function 9 (N=3)	2900	
f_{30}	Composition Function 10 (N=3)	3000	

Table 11
The description of CEC2020 benchmark suite.

No.	Function name	Optimal solution	Type
f_1	Shifted and rotated Bent Cigar function	100	Unimodal
f_2	Shifted and rotated Schwefel's function	1100	Basic functions
f_3	Shifted and rotated Lunacek bi-Rastrigin function	700	
f_4	Expanded Rosen-brock's plus Griewangk's function	1900	
f_5	Hybrid function 1 (N = 3)	1700	Hybrid functions
f_6	Hybrid function 2 (N = 4)	1600	
f_7	Hybrid function 3 (N = 5)	2100	
f_8	Composition function 1 (N = 3)	2200	Composition functions
f_9	Composition function 2 (N = 4)	2400	
f_{10}	Composition function 3 (N = 5)	2500	

Table 12
The description of CEC2022 benchmark suite.

No.	Function name	Optimal solution	Type
f_1	Shifted and full rotated Zakharov function	300	Unimodal
f_2	Shifted and full rotated Rosen-brock's function	400	Basic functions
f_3	Shifted and full rotated expanded Schaffer's f_6 function	600	
f_4	Shifted and full rotated non-continuous Rastrigin's function	800	
f_5	Shifted and full rotated Levy function	900	
f_6	Hybrid function 2 (N = 3)	1800	Hybrid functions
f_7	Hybrid function 3 (N = 6)	2000	
f_8	Hybrid Function 3 (N = 5)	2200	
f_9	Composition Function 1 (N = 5)	2300	Composition functions
f_{10}	Composition function 3 (N = 4)	2400	
f_{11}	Composition function 3 (N = 5)	2600	
f_{12}	Composition function 3 (N = 6)	2700	

Table 13
CEC2017-10D.

FunNum.		DE		PSO		CMA-ES		CL-PSO		SaDE		JADE		L-SHADE		WOA		PPSO		MBGO		COA		FATA		HO		MSA		CIA
f_1	mean	1.292×10^{8}	+	3.353×10^{3}	=	2.415×10^{7}	+	1.672×10^{8}	+	1.425×10^{6}	+	6.188×10^{6}	+	1.043×10^{7}	+	3.202×10^{7}	+	3.665×10^{7}	+	1.360×10^{5}	+	1.522×10^{8}	+	5.051×10^{9}	+	1.818×10^{8}	+	2.066×10^{7}	+	$\mathbf{1.474\times10^{3}}$
	std	1.153×10^{8}		4.269×10^{3}		7.005×10^{6}		8.727×10^{7}		8.092×10^{5}		8.128×10^{6}		1.818×10^{7}		5.952×10^{7}		3.496×10^{7}		7.571×10^{4}		5.561×10^{7}		1.270×10^{9}		2.203×10^{8}		1.346×10^{7}		1.558×10^{3}
f_3	mean	8.267×10^{3}	+	1.473×10^{3}	+	8.624×10^{2}	+	8.314×10^{3}	+	6.795×10^{3}	+	5.644×10^{3}	=	6.183×10^{3}	+	4.133×10^{3}	+	2.428×10^{3}	+	2.725×10^{3}	+	9.097×10^{3}	+	1.563×10^{4}	+	3.092×10^{3}	+	4.700×10^{3}	+	$\mathbf{3.033\times10^{2}}$
	std	5.335×10^{3}		9.302×10^{2}		1.353×10^{2}		2.789×10^{3}		2.678×10^{3}		5.654×10^{3}		6.428×10^{3}		2.523×10^{3}		2.025×10^{3}		1.067×10^{3}		2.983×10^{3}		4.564×10^{3}		1.231×10^{3}		1.934×10^{3}		4.201
f_4	mean	4.389×10^{2}	+	$\mathbf{4.042\times10^{2}}$	=	4.101×10^{2}	+	4.386×10^{2}	+	4.060×10^{2}	+	4.065×10^{2}	=	4.063×10^{2}	=	4.573×10^{2}	+	4.253×10^{2}	+	4.081×10^{2}	+	4.200×10^{2}	+	7.391×10^{2}	+	3.092×10^{3}	+	4.085×10^{2}	+	4.042×10^{2}
	std	3.250×10^{1}		1.848		1.408		1.291×10^{1}		6.550×10^{-1}		2.954		2.641		5.910×10^{1}		2.583×10^{1}		5.895		4.320		1.003×10^{2}		3.610×10^{1}		8.627×10^{-1}		1.978
f_5	mean	$\mathbf{5.113\times10^{2}}$	−	5.234×10^{2}	=	5.467×10^{2}	+	5.340×10^{2}	+	5.301×10^{2}	+	5.310×10^{2}	+	5.305×10^{2}	+	5.497×10^{2}	+	5.479×10^{2}	+	5.272×10^{2}	=	5.490×10^{2}	+	5.884×10^{2}	+	4.536×10^{2}	=	5.380×10^{2}	+	5.253×10^{2}
	std	4.871		1.321		4.966		4.967		4.213		5.032		5.142		1.942×10^{1}		1.736×10^{1}		4.909		5.249		9.114		1.359×10^{1}		6.209		1.026×10^{1}
f_6	mean	6.008×10^{2}	−	6.017×10^{2}	=	6.152×10^{2}	+	6.083×10^{2}	+	6.006×10^{2}	=	6.018×10^{2}	=	6.024×10^{2}	=	6.365×10^{2}	+	6.281×10^{2}	+	6.003×10^{2}	−	6.145×10^{2}	+	6.488×10^{2}	+	$\mathbf{5.341\times10^{2}}$	−	6.039×10^{2}	+	6.023×10^{2}
	std	1.212		2.009		2.593		1.926		1.479×10^{-1}		1.298		2.067		1.158×10^{1}		8.397		9.915×10^{-2}		2.560		6.097		8.198		9.762×10^{-1}		2.889
f_7	mean	7.238×10^{2}	=	7.496×10^{2}	+	7.757×10^{2}	+	7.498×10^{2}	+	7.451×10^{2}	+	7.457×10^{2}	+	7.450×10^{2}	+	7.888×10^{2}	+	7.711×10^{2}	+	7.413×10^{2}	+	7.855×10^{2}	+	9.115×10^{2}	+	$\mathbf{6.238\times10^{2}}$	−	7.584×10^{2}	+	7.303×10^{2}
	std	7.377		2.029×10^{1}		8.364		5.863		3.564		6.191		7.647		2.814×10^{1}		1.948×10^{1}		5.922		9.125		2.816×10^{1}		1.601×10^{1}		6.400		9.551
f_8	mean	8.109×10^{2}	−	8.265×10^{2}	=	8.526×10^{2}	+	8.270×10^{2}	=	8.314×10^{2}	=	8.317×10^{2}	=	8.314×10^{2}	=	8.418×10^{2}	+	8.311×10^{2}	=	8.282×10^{2}	=	8.540×10^{2}	+	8.800×10^{2}	+	$\mathbf{7.526\times10^{2}}$	=	8.368×10^{2}	+	8.227×10^{2}
	std	4.777		1.057×10^{1}		8.213		4.447		5.293		5.714		6.939		1.268×10^{1}		1.235×10^{1}		4.914		7.428		8.938		6.665		7.264		1.122×10^{1}
f_9	mean	9.202×10^{2}	+	1.003×10^{3}	+	1.131×10^{3}	+	9.719×10^{2}	+	9.017×10^{2}	=	9.246×10^{2}	=	9.281×10^{2}	=	1.657×10^{3}	+	1.313×10^{3}	+	$\mathbf{9.004\times10^{2}}$	=	1.202×10^{3}	+	2.100×10^{3}	+	$\mathbf{8.218\times10^{2}}$	−	9.182×10^{2}	+	9.045×10^{2}
	std	2.320×10^{1}		2.462×10^{2}		6.534×10^{1}		3.632×10^{1}		5.668×10^{-1}		3.029×10^{1}		3.462×10^{1}		4.436×10^{2}		2.403×10^{2}		1.848×10^{-1}		1.063×10^{2}		3.260×10^{2}		1.068×10^{2}		1.548×10^{1}		8.277
f_{10}	mean	1.622×10^{3}	−	1.703×10^{3}	−	2.473×10^{3}	+	2.109×10^{3}	=	2.319×10^{3}	=	2.190×10^{3}	=	2.166×10^{3}	=	2.073×10^{3}	=	2.187×10^{3}	=	2.451×10^{3}	+	2.286×10^{3}	=	2.791×10^{3}	+	$\mathbf{1.066\times10^{3}}$	=	2.000×10^{3}	=	2.122×10^{3}
	std	3.622×10^{2}		2.913×10^{2}		2.278×10^{2}		1.439×10^{2}		2.028×10^{2}		2.470×10^{2}		2.224×10^{2}		3.645×10^{2}		2.898×10^{2}		1.605×10^{2}		2.329×10^{2}		1.361×10^{2}		3.068×10^{2}		1.990×10^{2}		3.369×10^{2}
f_{11}	mean	1.159×10^{3}	=	$\mathbf{1.110\times10^{3}}$	−	1.128×10^{3}	=	1.169×10^{3}	+	1.113×10^{3}	−	1.127×10^{3}	=	1.123×10^{3}	=	1.241×10^{3}	+	1.225×10^{3}	+	1.116×10^{3}	−	1.155×10^{3}	+	1.659×10^{3}	+	2.077×10^{3}	+	1.146×10^{3}	=	1.146×10^{3}
	std	8.635×10^{1}		6.139		4.045		2.443×10^{1}		2.372		1.803×10^{1}		1.818×10^{1}		1.292×10^{2}		9.880×10^{1}		8.628		1.313×10^{1}		2.318×10^{2}		6.107×10^{1}		1.169×10^{1}		4.952×10^{1}
f_{12}	mean	2.314×10^{6}	+	2.309×10^{4}	=	2.417×10^{4}	=	1.771×10^{6}	+	9.209×10^{5}	+	2.739×10^{6}	=	2.381×10^{6}	=	1.248×10^{6}	=	2.050×10^{6}	+	5.659×10^{5}	+	1.893×10^{6}	+	1.491×10^{8}	+	$\mathbf{1.203\times10^{3}}$	−	1.774×10^{6}	+	4.156×10^{4}
	std	3.611×10^{6}		1.451×10^{4}		1.271×10^{4}		1.094×10^{6}		3.878×10^{5}		3.334×10^{6}		3.665×10^{6}		1.943×10^{6}		3.102×10^{6}		4.629×10^{5}		9.242×10^{5}		5.849×10^{7}		2.521×10^{6}		1.701×10^{6}		6.052×10^{4}
f_{13}	mean	8.326×10^{3}	=	9.472×10^{3}	=	$\mathbf{1.353\times10^{3}}$	−	8.209×10^{3}	+	1.577×10^{3}	−	5.047×10^{3}	=	4.046×10^{3}	=	1.315×10^{4}	=	4.978×10^{3}	=	2.938×10^{3}	=	1.869×10^{3}	−	1.158×10^{6}	+	2.284×10^{6}	+	2.606×10^{3}	=	4.809×10^{3}
	std	6.417×10^{3}		8.594×10^{3}		9.995		3.916×10^{3}		9.854×10^{1}		4.896×10^{3}		4.195×10^{3}		1.108×10^{4}		4.480×10^{3}		1.416×10^{3}		1.769×10^{2}		1.129×10^{6}		9.001×10^{3}		7.692×10^{2}		3.549×10^{3}
f_{14}	mean	2.427×10^{3}	=	1.484×10^{3}	=	1.430×10^{3}	−	1.585×10^{3}	+	$\mathbf{1.430\times10^{3}}$	−	1.475×10^{3}	=	1.453×10^{3}	=	1.530×10^{3}	+	1.585×10^{3}	=	1.470×10^{3}	=	1.440×10^{3}	−	2.628×10^{3}	+	1.575×10^{4}	+	1.453×10^{3}	=	1.467×10^{3}
	std	2.181×10^{3}		4.359×10^{1}		2.322		2.097×10^{2}		3.377		9.217×10^{1}		3.706×10^{1}		5.056×10^{1}		4.831×10^{2}		1.169×10^{1}		4.461		1.617×10^{3}		6.178×10^{2}		9.101		2.881×10^{1}
f_{15}	mean	2.408×10^{3}	=	1.937×10^{3}	=	1.519×10^{3}	−	1.762×10^{3}	=	$\mathbf{1.519\times10^{3}}$	−	1.707×10^{3}	−	1.565×10^{3}	−	2.646×10^{3}	=	2.399×10^{3}	=	1.757×10^{3}	=	1.581×10^{3}	−	1.290×10^{4}	+	1.765×10^{3}	+	1.585×10^{3}	−	1.741×10^{3}
	std	2.705×10^{3}		4.020×10^{2}		4.680		1.821×10^{2}		5.865		4.540×10^{2}		9.320×10^{1}		4.682×10^{3}		1.736×10^{3}		9.698×10^{1}		1.700×10^{1}		7.025×10^{3}		2.476×10^{3}		2.718×10^{1}		1.581×10^{2}
f_{16}	mean	$\mathbf{1.660\times10^{3}}$	−	1.727×10^{3}	=	1.775×10^{3}	=	1.777×10^{3}	=	1.670×10^{3}	−	1.671×10^{3}	−	1.694×10^{3}	=	1.952×10^{3}	+	1.948×10^{3}	+	1.669×10^{3}	−	1.721×10^{3}	=	2.035×10^{3}	+	4.545×10^{3}	=	1.670×10^{3}	−	1.774×10^{3}
	std	7.121×10^{1}		1.079×10^{2}		4.908×10^{1}		7.271×10^{1}		2.542×10^{1}		3.767×10^{1}		4.635×10^{1}		1.624×10^{2}		1.716×10^{2}		5.065×10^{1}		1.147×10^{2}		4.579×10^{1}		1.323×10^{1}		1.334×10^{1}		3.953×10^{1}
f_{17}	mean	$\mathbf{1.731\times10^{3}}$	−	1.747×10^{3}	−	1.817×10^{3}	+	1.756×10^{3}	=	1.755×10^{3}	=	1.758×10^{3}	=	1.758×10^{3}	=	1.833×10^{3}	=	1.800×10^{3}	=	1.756×10^{3}	=	1.789×10^{3}	=	1.889×10^{3}	+	1.829×10^{3}	=	1.769×10^{3}	=	1.782×10^{3}
	std	1.370×10^{1}		3.676×10^{1}		2.212×10^{1}		1.433×10^{1}		7.117		9.223		1.371×10^{1}		7.042×10^{1}		6.153×10^{1}		1.253×10^{1}		1.675×10^{1}		4.579×10^{1}		1.323×10^{1}		1.334×10^{1}		3.953×10^{1}
f_{18}	mean	8.749×10^{3}	=	1.162×10^{4}	=	1.835×10^{3}	−	4.952×10^{3}	=	2.488×10^{3}	−	9.759×10^{3}	=	9.381×10^{3}	=	1.258×10^{4}	=	9.264×10^{3}	=	5.486×10^{3}	=	3.012×10^{3}	−	3.575×10^{6}	+	$\mathbf{1.769\times10^{3}}$	=	6.979×10^{3}	=	9.399×10^{3}
	std	6.493×10^{3}		1.075×10^{4}		3.627		1.983×10^{3}		2.959×10^{2}		1.176×10^{4}		1.775×10^{4}		9.098×10^{3}		9.324×10^{3}		3.052×10^{3}		5.714×10^{2}		4.372×10^{6}		1.028×10^{4}		4.617×10^{3}		6.469×10^{3}
f_{19}	mean	3.236×10^{3}	=	2.900×10^{3}	+	1.909×10^{3}	−	2.017×10^{3}	+	$\mathbf{1.908\times10^{3}}$	−	2.174×10^{3}	=	1.947×10^{3}	=	1.252×10^{4}	+	3.670×10^{3}	=	1.983×10^{3}	+	1.932×10^{3}	=	3.727×10^{4}	+	1.558×10^{4}	+	1.939×10^{3}	=	1.975×10^{3}
	std	2.717×10^{3}		2.053×10^{3}		1.394		8.943		1.350		9.010×10^{2}		7.509×10^{1}		1.093×10^{4}		5.259×10^{3}		3.751×10^{1}		6.607		2.967×10^{4}		2.609×10^{3}		1.587×10^{1}		1.530×10^{2}
f_{20}	mean	$\mathbf{2.015\times10^{3}}$	−	2.032×10^{3}	−	2.186×10^{3}	+	2.061×10^{3}	−	2.039×10^{3}	−	2.049×10^{3}	−	2.053×10^{3}	−	2.170×10^{3}	=	2.150×10^{3}	=	2.046×10^{3}	−	2.085×10^{3}	=	2.191×10^{3}	+	4.248×10^{3}	=	2.065×10^{3}	−	2.119×10^{3}
	std	9.511		2.400×10^{1}		4.772×10^{1}		1.348×10^{1}		6.491		1.077×10^{1}		1.174×10^{1}		7.914×10^{1}		6.878×10^{1}		1.114×10^{1}		1.830×10^{1}		3.640×10^{1}		4.181×10^{1}		1.204×10^{1}		5.669×10^{1}
f_{21}	mean	2.264×10^{3}	−	2.277×10^{3}	=	2.343×10^{3}	+	2.277×10^{3}	−	2.291×10^{3}	=	2.303×10^{3}	=	2.302×10^{3}	=	2.339×10^{3}	+	2.307×10^{3}	=	2.282×10^{3}	=	2.284×10^{3}	=	2.282×10^{3}	−	2.096×10^{3}	=	2.255×10^{3}	=	2.315×10^{3}
	std	4.342×10^{1}		5.998×10^{1}		2.703×10^{1}		3.234×10^{1}		5.528×10^{1}		4.734×10^{1}		4.802×10^{1}		3.922×10^{1}		6.416×10^{1}		4.448×10^{1}		6.439×10^{1}		2.680×10^{1}		5.733×10^{1}		5.738×10^{1}		3.800×10^{1}
f_{22}	mean	2.319×10^{3}	+	$\mathbf{2.298\times10^{3}}$	=	2.317×10^{3}	+	2.325×10^{3}	+	2.301×10^{3}	−	2.309×10^{3}	+	2.309×10^{3}	+	2.358×10^{3}	+	2.329×10^{3}	+	2.302×10^{3}	−	2.335×10^{3}	+	2.681×10^{3}	+	2.272×10^{3}	−	2.309×10^{3}	+	2.307×10^{3}
	std	1.377×10^{1}		1.970×10^{1}		1.541		1.013×10^{1}		1.523×10^{1}		4.000		5.413		4.744×10^{1}		2.409×10^{1}		1.603×10^{1}		1.662×10^{1}		1.191×10^{2}		3.089×10^{1}		1.289×10^{1}		2.389×10^{1}
f_{23}	mean	$\mathbf{2.616\times10^{3}}$	−	2.626×10^{3}	=	2.642×10^{3}	+	2.635×10^{3}	+	2.632×10^{3}	=	2.630×10^{3}	=	2.632×10^{3}	=	2.657×10^{3}	+	2.648×10^{3}	+	2.626×10^{3}	=	2.638×10^{3}	+	2.709×10^{3}	+	2.330×10^{3}	−	2.633×10^{3}	=	2.629×10^{3}
	std	6.648		1.497×10^{1}		5.243		4.501×10^{1}		4.445		5.085		6.100		2.412×10^{1}		2.322×10^{1}		4.347		5.265		1.621×10^{1}		1.609×10^{1}		5.877		1.048×10^{1}
f_{24}	mean	2.718×10^{3}	−	2.735×10^{3}	=	2.773×10^{3}	+	$\mathbf{2.685\times10^{3}}$	−	2.752×10^{3}	=	2.737×10^{3}	=	2.738×10^{3}	=	2.772×10^{3}	+	2.757×10^{3}	=	2.737×10^{3}	=	2.773×10^{3}	+	2.799×10^{3}	+	2.645×10^{3}	=	2.753×10^{3}	=	2.753×10^{3}
	std	5.786×10^{1}		6.361×10^{1}		6.801		5.591×10^{1}		4.000×10^{1}		5.371×10^{1}		5.807×10^{1}		8.596×10^{1}		8.088×10^{1}		5.379×10^{1}		1.073×10^{1}		4.292×10^{1}		6.786×10^{1}		4.509×10^{1}		4.873×10^{1}
f_{25}	mean	2.949×10^{3}	+	$\mathbf{2.909\times10^{3}}$	=	2.929×10^{3}	=	2.951×10^{3}	+	2.931×10^{3}	=	2.924×10^{3}	=	2.925×10^{3}	=	2.972×10^{3}	+	2.950×10^{3}	+	2.936×10^{3}	=	2.949×10^{3}	+	3.210×10^{3}	+	2.748×10^{3}	−	2.939×10^{3}	+	2.912×10^{3}
	std	1.848×10^{1}		6.233×10^{1}		1.147×10^{1}		6.642		2.033×10^{1}		2.230×10^{1}		1.989×10^{1}		3.917×10^{1}		3.247×10^{1}		1.517×10^{1}		1.268×10^{1}		7.308×10^{1}		2.758×10^{1}		1.790×10^{1}		6.229×10^{1}
f_{26}	mean	3.121×10^{3}	+	2.926×10^{3}	=	2.930×10^{3}	=	3.190×10^{3}	+	$\mathbf{2.911\times10^{3}}$	=	2.947×10^{3}	=	2.942×10^{3}	=	3.289×10^{3}	=	3.208×10^{3}	=	2.929×10^{3}	=	3.020×10^{3}	=	3.518×10^{3}	+	2.956×10^{3}	=	2.968×10^{3}	=	3.102×10^{3}
	std	9.656		7.363		5.569		1.428×10^{2}		1.373×10^{1}		5.395×10^{1}		5.696×10^{1}		3.332×10^{2}		3.476×10^{2}		1.633×10^{1}		1.408×10^{1}		1.311×10^{2}		1.376×10^{2}		2.455×10^{1}		3.679×10^{2}
f_{27}	mean	3.112×10^{3}	=	3.102×10^{3}	=	3.095×10^{3}	−	3.126×10^{3}	+	$\mathbf{3.092\times10^{3}}$	−	3.092×10^{3}	−	3.093×10^{3}	−	3.181×10^{3}	+	3.130×10^{3}	+	3.095×10^{3}	−	3.094×10^{3}	−	3.173×10^{3}	+	3.049×10^{3}	−	3.093×10^{3}	−	3.109×10^{3}
	std	1.742×10^{1}		1.505×10^{1}		1.062		7.194		9.505×10^{-1}		2.136		2.733		3.742×10^{1}		3.215×10^{1}		2.364		1.009		1.787×10^{1}		2.600×10^{1}		1.625		2.097×10^{1}
f_{28}	mean	3.291×10^{3}	=	3.261×10^{3}	=	3.289×10^{3}	=	3.391×10^{3}	=	3.267×10^{3}	=	3.292×10^{3}	=	3.272×10^{3}	=	3.416×10^{3}	=	3.391×10^{3}	=	3.263×10^{3}	=	3.233×10^{3}	=	3.484×10^{3}	+	3.126×10^{3}	=	$\mathbf{3.217\times10^{3}}$	=	3.332×10^{3}
	std	1.211×10^{2}		1.274×10^{2}		1.113×10^{2}		9.040×10^{1}		1.082×10^{2}		1.066×10^{2}		1.020×10^{2}		1.567×10^{2}		1.737×10^{2}		9.788×10^{1}		5.090×10^{1}		1.163×10^{2}		4.620×10^{1}		1.124×10^{2}
f_{29}	mean	$\mathbf{3.175\times10^{3}}$	−	3.205×10^{3}	=	3.227×10^{3}	=	3.239×10^{3}	=	3.228×10^{3}	=	3.208×10^{3}	=	3.214×10^{3}	=	3.396×10^{3}	+	3.326×10^{3}	+	3.208×10^{3}	=	3.230×10^{3}	=	3.361×10^{3}	+	3.312×10^{3}	=	3.220×10^{3}	=	3.250×10^{3}
	std	2.578×10^{1}		4.477×10^{1}		2.977×10^{1}		2.942×10^{1}		2.599×10^{1}		2.143×10^{1}		2.772×10^{1}		1.153×10^{2}		8.782×10^{1}		2.172×10^{1}		2.712×10^{1}		5.769×10^{1}		4.172×10^{1}		3.226×10^{1}		5.837×10^{1}
f_{30}	mean	6.233×10^{5}	=	6.863×10^{5}	=	$\mathbf{1.770\times10^{4}}$	−	6.935×10^{5}	=	2.219×10^{5}	=	2.588×10^{5}	=	1.305×10^{5}	=	9.884×10^{5}	=	1.977×10^{6}	=	6.660×10^{5}	=	1.442×10^{5}	=	8.794×10^{6}	+	3.251×10^{3}	−	2.461×10^{5}	=	5.509×10^{5}
	std	9.212×10^{5}		8.950×10^{5}		1.959×10^{4}		5.033×10^{5}		2.601×10^{5}		3.088×10^{5}		1.598×10^{5}		1.952×10^{6}		2.529×10^{6}		7.716×10^{5}		2.062×10^{5}		4.439×10^{6}		2.141×10^{6}		2.382×10^{5}		6.727×10^{5}
+ /=/−		8/10/11		4/21/4		15/7/17		17/9/3		6/13/10		4/21/4		5/21/3		19/10/0		15/14/0		7/16/6		14/10/4		28/0/1		8/12/9		11/14/4		–	
Avg. Rank		5.8		3.8		6.6		8.3		**3.7**		6.1		5.5		11.8		10.2		4.0		8.1		14.1		9.7		5.5		5.3	

Table 14
CEC2017-30D.

FunNum.		DE	PSO	CMA-ES	CL-PSO	SaDE	JADE	L-SHADE	WOA	PPSO	MBGO	COA	FATA	HO	MSA	CIA
f_1	mean	5.247×10^{9} +	1.240×10^{4} +	3.073×10^{9} +	1.334×10^{7} +	9.008×10^{3} +	5.393×10^{6} =	6.621×10^{6} =	5.006×10^{9} +	1.440×10^{9} +	3.023×10^{8} +	5.602×10^{9} +	6.336×10^{10} +	4.335×10^{9} +	5.548×10^{7} +	**2.259×10^{3}**
	std	2.839×10^{9}	2.656×10^{4}	3.966×10^{8}	4.329×10^{6}	5.716×10^{3}	1.003×10^{7}	1.670×10^{7}	3.311×10^{9}	6.935×10^{8}	1.927×10^{8}	1.079×10^{9}	6.597×10^{9}	1.533×10^{9}	2.160×10^{7}	2.021×10^{3}
f_3	mean	6.258×10^{4} +	7.343×10^{4} +	9.717×10^{3} +	9.636×10^{4} +	1.094×10^{5} +	5.852×10^{3} +	6.593×10^{4} +	1.385×10^{5} +	7.485×10^{4} +	6.541×10^{4} +	1.522×10^{5} +	1.415×10^{5} +	5.245×10^{4} +	7.867×10^{4} +	**9.870×10^{3}**
	std	1.178×10^{4}	1.916×10^{4}	1.762×10^{4}	1.359×10^{4}	1.287×10^{4}	5.052×10^{4}	6.320×10^{4}	7.504×10^{4}	1.966×10^{4}	9.416×10^{3}	2.040×10^{4}	1.627×10^{4}	6.498×10^{3}	1.312×10^{4}	4.595×10^{3}
f_4	mean	1.171×10^{3} +	5.191×10^{2} =	7.247×10^{2} +	5.509×10^{2} +	5.017×10^{2} =	5.007×10^{2} =	4.981×10^{2} =	1.101×10^{3} +	7.725×10^{2} +	6.136×10^{2} +	8.039×10^{2} +	1.365×10^{4} +	5.245×10^{4} +	5.189×10^{2} =	5.017×10^{2}
	std	4.716×10^{2}	3.446×10^{1}	5.633×10^{1}	1.280×10^{1}	1.356×10^{1}	2.374×10^{1}	1.836×10^{1}	3.254×10^{2}	1.276×10^{2}	5.318×10^{1}	7.430×10^{1}	3.060×10^{3}	3.621×10^{2}	1.405×10^{1}	2.063×10^{1}
f_5	mean	**5.771×10^{2}** −	6.421×10^{2} =	7.692×10^{2} +	6.571×10^{2} +	6.830×10^{2} +	6.334×10^{2} =	6.442×10^{2} =	8.327×10^{2} +	7.585×10^{2} +	7.087×10^{2} +	7.708×10^{2} +	9.753×10^{2} +	1.326×10^{3} +	6.641×10^{2} +	6.172×10^{2}
	std	1.529×10^{1}	3.674×10^{1}	1.724×10^{1}	1.014×10^{1}	1.289×10^{1}	2.050×10^{1}	2.374×10^{1}	5.761×10^{1}	3.718×10^{1}	1.386×10^{1}	1.435×10^{1}	2.302×10^{1}	4.074×10^{1}	2.981×10^{1}	4.003×10^{1}
f_6	mean	6.063×10^{2} −	6.249×10^{2} +	6.392×10^{2} +	6.076×10^{2} −	6.002×10^{2} −	6.024×10^{2} −	6.022×10^{2} −	6.715×10^{2} +	6.645×10^{2} +	6.061×10^{2} −	6.295×10^{2} +	6.952×10^{2} +	7.199×10^{2} +	6.060×10^{2} −	6.130×10^{2}
	std	2.512	6.826	4.347	1.452	4.622×10^{-2}	2.588	2.596	1.109×10^{1}	6.075	2.086	3.000	5.546	7.449	1.201	6.835
f_7	mean	9.001×10^{2} +	1.058×10^{3} +	1.163×10^{3} +	8.933×10^{2} +	9.124×10^{2} +	8.752×10^{2} +	8.881×10^{2} +	1.291×10^{3} +	1.319×10^{3} +	9.597×10^{2} +	1.163×10^{3} +	2.394×10^{3} +	**6.561×10^{2}** −	9.430×10^{2} +	8.246×10^{2}
	std	5.075×10^{1}	9.112×10^{1}	3.166×10^{1}	1.769×10^{1}	1.264×10^{1}	2.409×10^{1}	2.681×10^{1}	9.337×10^{1}	1.186×10^{2}	1.307×10^{1}	3.665×10^{1}	1.201×10^{2}	5.249×10^{1}	2.674×10^{1}	4.859×10^{1}
f_8	mean	**8.761×10^{2}** =	9.387×10^{2} +	1.066×10^{3} +	9.573×10^{2} +	9.859×10^{2} +	9.317×10^{2} +	9.456×10^{2} +	1.054×10^{3} +	9.993×10^{2} +	1.006×10^{3} +	1.064×10^{3} +	1.232×10^{3} +	1.036×10^{3} +	9.610×10^{2} +	8.883×10^{2}
	std	1.883×10^{1}	3.735×10^{1}	1.569×10^{1}	1.070×10^{1}	1.169×10^{1}	2.287×10^{1}	2.584×10^{1}	7.008×10^{1}	3.666×10^{1}	1.209×10^{1}	1.294×10^{1}	1.889×10^{1}	2.395×10^{1}	2.575×10^{1}	2.150×10^{1}
f_9	mean	1.704×10^{3} =	4.509×10^{3} +	5.127×10^{3} +	3.471×10^{3} +	**9.013×10^{2}** −	1.748×10^{3} =	1.595×10^{3} =	9.243×10^{3} +	8.324×10^{3} +	1.400×10^{3} =	5.502×10^{3} +	1.880×10^{4} +	9.526×10^{2} −	1.160×10^{3} −	1.450×10^{3}
	std	5.459×10^{2}	1.978×10^{3}	7.678×10^{2}	7.271×10^{2}	8.645×10^{-1}	1.119×10^{3}	9.379×10^{2}	3.611×10^{3}	2.918×10^{3}	2.123×10^{2}	8.234×10^{2}	1.703×10^{3}	7.960×10^{2}	9.687×10^{1}	2.954×10^{2}
f_{10}	mean	5.002×10^{3} −	**4.896×10^{3}** =	8.635×10^{3} +	6.301×10^{3} −	8.194×10^{3} +	6.859×10^{3} =	6.453×10^{3} =	6.579×10^{3} =	6.694×10^{3} =	8.611×10^{3} +	8.082×10^{3} +	8.861×10^{3} +	5.189×10^{3} −	6.613×10^{3} =	6.826×10^{3}
	std	1.629×10^{3}	1.050×10^{3}	2.743×10^{2}	2.741×10^{2}	3.418×10^{2}	8.050×10^{2}	3.545×10^{2}	1.063×10^{3}	7.375×10^{2}	2.725×10^{2}	4.861×10^{2}	3.039×10^{2}	5.546×10^{2}	7.289×10^{2}	8.303×10^{2}
f_{11}	mean	2.978×10^{3} +	**1.217×10^{3}** =	1.360×10^{3} +	1.513×10^{3} +	1.258×10^{3} =	1.675×10^{3} =	1.350×10^{3} =	3.715×10^{3} +	1.993×10^{3} +	2.573×10^{3} +	1.883×10^{3} +	1.022×10^{4} +	5.705×10^{3} +	1.443×10^{3} +	1.246×10^{3}
	std	1.255×10^{3}	4.696×10^{1}	1.527×10^{1}	9.604×10^{1}	1.974×10^{1}	5.820×10^{2}	2.200×10^{2}	1.547×10^{3}	5.312×10^{2}	5.694×10^{2}	1.204×10^{2}	2.008×10^{3}	2.303×10^{2}	6.114×10^{1}	3.900×10^{1}
f_{12}	mean	3.153×10^{8} +	7.514×10^{5} −	2.259×10^{7} +	4.172×10^{6} =	1.765×10^{6} −	1.220×10^{7} =	1.322×10^{7} =	7.673×10^{7} +	1.039×10^{8} +	9.906×10^{6} +	1.461×10^{8} +	8.706×10^{9} +	**1.833×10^{3}** −	7.916×10^{6} +	4.271×10^{6}
	std	2.891×10^{8}	1.117×10^{6}	4.324×10^{6}	1.551×10^{6}	1.445×10^{6}	1.889×10^{7}	2.188×10^{7}	9.259×10^{7}	1.205×10^{8}	7.452×10^{6}	4.234×10^{7}	1.677×10^{9}	3.857×10^{8}	4.726×10^{6}	3.203×10^{6}
f_{13}	mean	1.032×10^{8} +	2.124×10^{4} −	7.374×10^{3} −	7.272×10^{4} =	2.694×10^{5} +	4.792×10^{6} =	1.370×10^{6} =	7.257×10^{6} =	5.324×10^{6} +	9.318×10^{4} =	1.458×10^{6} +	4.345×10^{9} +	4.916×10^{8} +	3.227×10^{5} +	6.219×10^{4}
	std	1.569×10^{8}	2.080×10^{4}	1.214×10^{3}	3.353×10^{4}	1.442×10^{5}	7.245×10^{6}	2.893×10^{6}	2.150×10^{7}	2.021×10^{7}	7.160×10^{4}	8.986×10^{5}	1.300×10^{9}	2.343×10^{6}	4.101×10^{5}	3.049×10^{4}
f_{14}	mean	8.434×10^{5} +	2.905×10^{4} +	**1.508×10^{3}** −	1.177×10^{5} +	9.089×10^{3} +	1.513×10^{5} =	2.205×10^{4} =	5.091×10^{5} +	1.130×10^{5} +	9.925×10^{4} +	1.824×10^{4} +	1.677×10^{6} +	7.146×10^{5} +	5.422×10^{3} =	3.508×10^{3}
	std	1.126×10^{6}	2.826×10^{4}	8.214	8.436×10^{4}	4.474×10^{3}	2.121×10^{5}	4.922×10^{4}	8.489×10^{5}	1.321×10^{5}	7.276×10^{4}	9.769×10^{3}	8.786×10^{5}	7.200×10^{5}	2.698×10^{3}	1.825×10^{3}
f_{15}	mean	1.268×10^{7} +	1.457×10^{4} −	**1.843×10^{3}** −	2.388×10^{4} −	3.560×10^{4} =	1.348×10^{6} =	1.369×10^{5} =	3.726×10^{4} =	1.415×10^{5} =	1.236×10^{5} +	1.953×10^{4} −	4.832×10^{8} +	4.898×10^{5} =	4.541×10^{4} =	4.526×10^{4}
	std	3.805×10^{7}	1.215×10^{4}	3.888×10^{1}	1.710×10^{4}	1.623×10^{4}	2.794×10^{6}	2.886×10^{5}	3.110×10^{4}	4.337×10^{5}	9.762×10^{4}	7.874×10^{3}	1.898×10^{8}	4.048×10^{4}	3.497×10^{4}	2.653×10^{4}
f_{16}	mean	**2.382×10^{3}** −	2.650×10^{3} =	3.527×10^{3} +	2.657×10^{3} −	3.025×10^{3} =	2.847×10^{3} −	2.772×10^{3} =	3.583×10^{3} +	3.599×10^{3} +	3.385×10^{3} +	3.485×10^{3} +	5.017×10^{3} +	4.524×10^{4} +	2.619×10^{3} −	2.857×10^{3}
	std	2.745×10^{2}	2.995×10^{2}	1.780×10^{2}	1.459×10^{2}	1.995×10^{2}	1.529×10^{2}	2.286×10^{2}	5.626×10^{2}	4.659×10^{2}	2.254×10^{2}	1.812×10^{2}	2.941×10^{2}	3.582×10^{2}	2.834×10^{2}	2.092×10^{2}
f_{17}	mean	**1.976×10^{3}** −	2.120×10^{3} =	2.578×10^{3} +	2.028×10^{3} −	2.189×10^{3} =	2.106×10^{3} =	2.111×10^{3} =	2.884×10^{3} +	2.657×10^{3} +	2.257×10^{3} =	2.509×10^{3} +	3.351×10^{3} +	3.254×10^{3} =	2.045×10^{3} −	2.202×10^{3}
	std	1.334×10^{2}	2.008×10^{2}	1.298×10^{2}	9.674×10^{1}	7.622×10^{1}	1.379×10^{2}	1.176×10^{2}	3.692×10^{2}	3.390×10^{2}	1.387×10^{2}	1.288×10^{2}	2.094×10^{2}	1.795×10^{2}	1.266×10^{2}	1.888×10^{2}
f_{18}	mean	1.724×10^{6} +	7.882×10^{5} +	4.008×10^{3} −	4.654×10^{5} +	1.157×10^{6} +	1.123×10^{6} =	5.916×10^{5} =	9.231×10^{5} +	1.443×10^{6} +	1.338×10^{6} +	1.371×10^{6} +	2.412×10^{7} +	**2.326×10^{3}** −	3.613×10^{5} +	1.265×10^{5}
	std	2.816×10^{6}	1.502×10^{6}	8.521×10^{2}	2.703×10^{5}	6.026×10^{5}	1.282×10^{6}	8.204×10^{5}	1.865×10^{6}	1.915×10^{6}	1.109×10^{6}	5.007×10^{5}	9.964×10^{6}	1.390×10^{6}	2.549×10^{5}	6.999×10^{4}
f_{19}	mean	3.194×10^{6} =	1.729×10^{4} −	**2.138×10^{3}** −	1.985×10^{4} −	3.855×10^{4} −	6.181×10^{5} =	1.458×10^{5} =	1.201×10^{5} −	7.101×10^{5} =	1.881×10^{5} =	1.482×10^{5} =	6.891×10^{8} +	1.247×10^{6} +	8.610×10^{4} −	1.463×10^{5}
	std	7.224×10^{6}	1.565×10^{4}	4.566×10^{1}	9.820×10^{3}	2.105×10^{4}	8.243×10^{5}	2.310×10^{5}	2.238×10^{5}	1.254×10^{6}	2.108×10^{5}	6.459×10^{4}	2.566×10^{8}	4.105×10^{6}	8.225×10^{4}	7.952×10^{4}
f_{20}	mean	**2.228×10^{3}** −	2.474×10^{3} =	2.931×10^{3} +	2.421×10^{3} −	2.550×10^{3} =	2.500×10^{3} =	2.480×10^{3} =	2.967×10^{3} +	2.775×10^{3} +	2.751×10^{3} +	2.788×10^{3} +	2.977×10^{3} +	5.358×10^{6} =	2.347×10^{3} −	2.602×10^{3}
	std	9.245×10^{1}	1.864×10^{2}	1.167×10^{2}	1.056×10^{2}	1.342×10^{2}	1.088×10^{2}	1.075×10^{2}	2.335×10^{2}	2.048×10^{2}	1.103×10^{2}	1.054×10^{2}	1.071×10^{2}	1.279×10^{2}	1.373×10^{2}	1.781×10^{2}
f_{21}	mean	**2.383×10^{3}** =	2.425×10^{3} +	2.555×10^{3} +	2.455×10^{3} +	2.473×10^{3} +	2.432×10^{3} +	2.444×10^{3} +	2.583×10^{3} +	2.566×10^{3} +	2.503×10^{3} +	2.551×10^{3} +	2.739×10^{3} +	2.586×10^{3} +	2.460×10^{3} +	2.400×10^{3}
	std	2.088×10^{1}	2.786×10^{1}	1.624×10^{1}	1.514×10^{1}	1.296×10^{1}	2.427×10^{1}	2.048×10^{1}	5.595×10^{1}	4.387×10^{1}	1.447×10^{1}	1.467×10^{1}	1.592×10^{1}	4.723×10^{1}	3.104×10^{1}	2.420×10^{1}
f_{22}	mean	3.342×10^{3} =	4.550×10^{3} =	9.789×10^{3} +	3.341×10^{3} =	3.095×10^{3} =	4.015×10^{3} =	3.759×10^{3} =	7.112×10^{3} =	7.426×10^{3} =	2.460×10^{3} =	8.206×10^{3} +	8.881×10^{3} +	2.503×10^{3} =	**2.339×10^{3}** =	5.193×10^{3}
	std	6.139×10^{2}	2.055×10^{3}	1.070×10^{3}	4.586×10^{2}	1.911×10^{3}	2.064×10^{3}	2.051×10^{3}	1.951×10^{3}	1.857×10^{3}	8.111×10^{1}	2.451×10^{3}	9.383×10^{2}	1.395×10^{3}	8.759	2.999×10^{3}
f_{23}	mean	**2.777×10^{3}** =	2.833×10^{3} +	2.910×10^{3} +	2.827×10^{3} +	2.828×10^{3} +	2.784×10^{3} =	2.794×10^{3} =	3.181×10^{3} +	3.148×10^{3} +	2.870×10^{3} +	2.901×10^{3} +	3.328×10^{3} +	4.154×10^{3} +	2.820×10^{3} +	2.787×10^{3}
	std	3.805×10^{1}	4.862×10^{1}	1.512×10^{1}	1.577×10^{1}	1.153×10^{1}	1.930×10^{1}	2.076×10^{1}	9.076×10^{1}	1.162×10^{2}	2.141×10^{1}	1.189×10^{1}	6.254×10^{1}	5.957×10^{1}	2.289×10^{1}	3.605×10^{1}
f_{24}	mean	**2.951×10^{3}** =	3.028×10^{3} +	3.074×10^{3} +	3.025×10^{3} +	3.002×10^{3} +	2.956×10^{3} =	2.969×10^{3} =	3.316×10^{3} +	3.271×10^{3} +	3.047×10^{3} +	3.058×10^{3} +	3.562×10^{3} +	2.990×10^{3} +	2.965×10^{3} =	2.962×10^{3}
	std	3.945×10^{1}	6.674×10^{1}	1.682×10^{1}	1.427×10^{1}	1.151×10^{1}	2.315×10^{1}	2.247×10^{1}	1.218×10^{2}	1.064×10^{2}	2.114×10^{1}	9.910	5.515×10^{1}	9.582×10^{1}	3.397×10^{1}	4.619×10^{1}
f_{25}	mean	3.204×10^{3} +	2.919×10^{3} =	3.138×10^{3} +	2.970×10^{3} +	**2.888×10^{3}** −	2.896×10^{3} −	2.891×10^{3} −	3.149×10^{3} +	3.152×10^{3} +	3.007×10^{3} +	3.217×10^{3} +	8.103×10^{3} +	3.195×10^{3} +	2.919×10^{3} =	2.914×10^{3}
	std	1.435×10^{2}	2.965×10^{1}	5.932×10^{1}	1.388×10^{1}	6.544×10^{-1}	1.509×10^{1}	7.465	9.569×10^{1}	1.280×10^{2}	4.278×10^{1}	5.905×10^{1}	7.934×10^{2}	8.173×10^{1}	1.282×10^{1}	1.812×10^{1}
f_{26}	mean	5.074×10^{3} −	5.625×10^{3} +	6.335×10^{3} +	5.793×10^{3} +	5.342×10^{3} +	4.930×10^{3} =	4.947×10^{3} =	8.290×10^{3} +	8.079×10^{3} +	5.975×10^{3} +	6.314×10^{3} +	1.063×10^{4} +	**3.178×10^{3}** −	5.329×10^{3} =	5.023×10^{3}
	std	4.663×10^{2}	8.432×10^{2}	1.343×10^{2}	1.901×10^{2}	1.438×10^{2}	2.840×10^{2}	4.728×10^{2}	7.897×10^{2}	1.169×10^{3}	2.645×10^{2}	1.715×10^{2}	6.201×10^{2}	1.086×10^{3}	3.382×10^{2}	5.340×10^{2}
f_{27}	mean	3.272×10^{3} =	3.316×10^{3} +	3.241×10^{3} −	3.290×10^{3} +	3.220×10^{3} −	**3.218×10^{3}** −	3.220×10^{3} −	3.630×10^{3} +	3.438×10^{3} +	3.269×10^{3} =	3.229×10^{3} −	4.005×10^{3} +	6.921×10^{3} +	3.225×10^{3} −	3.266×10^{3}
	std	2.349×10^{1}	4.990×10^{1}	1.174×10^{1}	7.690	6.939	1.286×10^{1}	1.304×10^{1}	2.072×10^{2}	1.215×10^{2}	2.313×10^{1}	4.343	1.108×10^{2}	1.390×10^{2}	7.726	2.413×10^{1}
f_{28}	mean	3.903×10^{3} +	3.264×10^{3} =	3.448×10^{3} +	3.390×10^{3} +	**3.227×10^{3}** =	3.254×10^{3} =	3.244×10^{3} =	3.707×10^{3} +	3.509×10^{3} +	3.451×10^{3} +	3.602×10^{3} +	7.265×10^{3} +	3.458×10^{3} +	3.293×10^{3} +	3.239×10^{3}
	std	3.135×10^{2}	4.262×10^{1}	3.829×10^{1}	2.370×10^{1}	1.832×10^{1}	6.178×10^{1}	4.953×10^{1}	2.306×10^{2}	1.874×10^{2}	9.484×10^{1}	1.456×10^{2}	5.672×10^{2}	1.750×10^{2}	2.518×10^{1}	1.994×10^{1}
f_{29}	mean	3.746×10^{3} −	3.948×10^{3} −	4.451×10^{3} +	4.046×10^{3} =	4.047×10^{3} =	3.808×10^{3} −	3.887×10^{3} −	5.534×10^{3} +	5.065×10^{3} +	4.305×10^{3} =	4.265×10^{3} =	6.094×10^{3} +	**3.707×10^{3}** −	3.876×10^{3} −	4.213×10^{3}
	std	1.650×10^{2}	2.066×10^{2}	1.438×10^{2}	1.295×10^{2}	1.363×10^{2}	1.514×10^{2}	1.362×10^{2}	7.709×10^{2}	4.888×10^{2}	1.907×10^{2}	2.109×10^{2}	3.458×10^{2}	4.025×10^{2}	1.537×10^{2}	2.512×10^{2}
f_{30}	mean	1.303×10^{7} +	8.604×10^{4} −	8.332×10^{4} −	1.145×10^{6} =	1.995×10^{5} −	3.125×10^{5} −	1.879×10^{5} −	2.336×10^{6} =	3.342×10^{6} =	1.301×10^{6} =	2.991×10^{5} −	5.435×10^{8} +	**4.741×10^{3}** −	4.566×10^{5} −	1.393×10^{6}
	std	4.208×10^{7}	1.938×10^{5}	3.471×10^{4}	4.879×10^{5}	1.508×10^{5}	3.464×10^{5}	2.404×10^{5}	4.875×10^{6}	3.718×10^{6}	9.800×10^{5}	1.195×10^{5}	1.463×10^{8}	1.069×10^{7}	2.950×10^{5}	9.463×10^{5}
+ /=/−		13/9/7	13/10/6	22/0/6	17/5/7	13/9/7	4/19/6	4/20/6	23/5/1	24/5/0	20/8/1	24/2/3	29/0/0	17/4/8	12/8/9	–
Avg.	Rank	6.4	4.4	8.1	5.2	4.7	4.4	4.0	11.4	10.8	7.6	10.0	14.4	9.2	4.4	**3.7**

facilitates local exploration when the "prey" is the optimal individual within the crested ibis population. In the second scenario, when the "prey" is the optimal individual in the fish population, the strategy primarily focuses on local exploration but also enhances the diversity of the crested ibis population through information exchange with the fish population.

In the "start fishing" strategy, the diversity of the crested ibis population is significantly increased by exchanging dimensions between crested ibis individuals and fish individuals. Additionally, introducing an adaptive factor allows the crested ibis population to automatically adjust the step size of exploration based on the degree of evolution and the population's diversity during the exploration process.

For the fish population's evolutionary strategy, the optimal individual within the population is directly used as a substrate to accelerate local exploration. This approach facilitates rapid convergence of the fish population toward the neighborhood of the current optimal individual. However, optimal individuals from the crested ibis population are introduced as a diversification mechanism to prevent over-convergence and a complete loss of global exploration capability. This strategy differs from the introduction of random individuals, as it prevents the population from converging to a tiny region while preserving a certain degree of global exploration capability.

In summary, while the crested ibis population primarily focuses on global exploration and the fish population on local exploration, the CIA algorithm retains a degree of local exploration capability within the crested ibis population and global exploration capability within the fish population. Combined with the information exchange between the two populations, this design enables the CIA algorithm to balance exploration and exploitation dynamically.

6.4. Analysis of experimental results

Firstly, the CIA algorithm demonstrates strong performance in lower-dimensional optimization problems. Its average ranking is 5.3 for 10D on the CEC2017 benchmark and 6.8 and 3.8 for 10D and 20D on the CEC2022 benchmark, respectively. These results indicate that the CIA's performance on low-dimensional, less challenging optimization problems is comparable to mainstream classical algorithms. Furthermore, it exhibits strong competitiveness against similar DE algorithms and their improved variants, which are well-regarded for their convergence capabilities.

The CIA algorithm exhibits even more potent performance as the dimensionality increases to 30D and 50D. Its average rankings on the CEC2017 benchmark are 3.7 for 30D and 3.1 for 50D, placing it ahead of all compared algorithms. Notably, when competing against high-performance algorithms such as JADE and L-SHADE, which are renowned for their competitiveness, the CIA slightly outperforms them, demonstrating its robustness in higher-dimensional optimization problems.

As the dimensionality increases to 100D, the difficulty and complexity of optimization problems rise significantly. Classical algorithms

Table 15
CEC2017-50D.

FunNum.		DE	PSO	CMA-ES	CL-PSO	SaDE	JADE	L-SHADE	WOA	PPSO	MBGO	COA	FATA	HO	MSA	CIA
f_1	mean	1.996×10^{10} +	5.946×10^{7} +	1.109×10^{10} +	6.024×10^{7} +	3.889×10^{3} =	1.224×10^{7} =	1.739×10^{5} =	2.416×10^{10} +	5.946×10^{9} +	5.095×10^{9} +	1.724×10^{10} +	1.552×10^{11} +	1.496×10^{10} +	9.936×10^{7} +	**3.307×10^{3}**
	std	7.682×10^{9}	1.354×10^{8}	2.871×10^{9}	2.278×10^{7}	4.131×10^{3}	3.144×10^{7}	6.126×10^{5}	7.688×10^{9}	2.720×10^{9}	1.404×10^{9}	2.197×10^{9}	1.150×10^{10}	2.708×10^{9}	3.369×10^{7}	3.357×10^{3}
f_3	mean	2.107×10^{5} +	1.837×10^{5} +	2.686×10^{5} +	1.913×10^{5} +	2.165×10^{5} +	1.408×10^{5} =	1.025×10^{5} =	1.341×10^{5} +	1.617×10^{5} +	1.615×10^{5} +	3.151×10^{5} +	2.882×10^{5} +	1.267×10^{5} +	2.679×10^{5} +	**4.155×10^{4}**
	std	3.810×10^{4}	3.599×10^{4}	3.409×10^{4}	1.537×10^{4}	2.632×10^{4}	9.285×10^{4}	1.079×10^{5}	2.883×10^{4}	2.721×10^{4}	2.096×10^{4}	3.824×10^{4}	2.177×10^{4}	1.119×10^{4}	6.043×10^{4}	8.661×10^{3}
f_4	mean	3.030×10^{3} +	6.526×10^{2} =	1.691×10^{3} +	6.946×10^{2} +	**5.507×10^{2}** −	5.705×10^{2} =	5.571×10^{2} −	3.505×10^{3} +	1.523×10^{3} +	1.312×10^{3} +	2.145×10^{3} +	4.269×10^{4} +	1.267×10^{5} +	6.475×10^{2} +	6.080×10^{2}
	std	9.302×10^{2}	1.210×10^{2}	3.245×10^{2}	3.198×10^{1}	4.145×10^{1}	7.934×10^{1}	5.439×10^{1}	1.438×10^{3}	3.453×10^{2}	2.323×10^{2}	3.043×10^{2}	5.583×10^{3}	6.532×10^{2}	2.311×10^{1}	2.986×10^{1}
f_5	mean	7.119×10^{2} =	8.072×10^{2} +	1.017×10^{3} +	8.234×10^{2} +	8.560×10^{2} +	7.475×10^{2} +	7.691×10^{2} +	1.033×10^{3} +	9.589×10^{2} +	9.364×10^{2} +	1.009×10^{3} +	1.403×10^{3} +	3.116×10^{3} +	7.606×10^{2} +	**7.028×10^{2}**
	std	3.309×10^{1}	6.204×10^{1}	2.679×10^{1}	1.918×10^{1}	1.394×10^{1}	4.940×10^{1}	4.553×10^{1}	7.935×10^{1}	6.628×10^{1}	2.148×10^{1}	1.660×10^{1}	3.688×10^{1}	3.545×10^{1}	5.080×10^{1}	5.535×10^{1}
f_6	mean	6.150×10^{2} =	6.381×10^{2} +	6.555×10^{2} +	6.069×10^{2} −	**6.001×10^{2}** −	6.035×10^{2} −	6.017×10^{2} −	6.813×10^{2} +	6.753×10^{2} +	6.197×10^{2} =	6.352×10^{2} +	7.150×10^{2} +	8.521×10^{2} +	6.069×10^{2} −	6.210×10^{2}
	std	3.955	6.977	6.042	1.380	1.030×10^{-2}	3.764	2.312	8.251	6.477	3.826	2.188	4.532	6.249	1.208	1.024×10^{1}
f_7	mean	1.264×10^{3} +	1.597×10^{3} +	1.486×10^{3} +	1.069×10^{3} +	1.116×10^{3} +	1.018×10^{3} +	1.038×10^{3} +	1.815×10^{3} +	1.977×10^{3} +	1.258×10^{3} +	1.667×10^{3} +	4.366×10^{3} +	**6.644×10^{2}** −	1.149×10^{3} +	9.373×10^{2}
	std	1.657×10^{2}	2.025×10^{2}	4.009×10^{1}	1.890×10^{1}	1.970×10^{1}	5.406×10^{1}	4.187×10^{1}	9.314×10^{1}	1.696×10^{2}	3.009×10^{1}	5.159×10^{1}	1.888×10^{2}	9.300×10^{1}	2.820×10^{1}	1.604×10^{1}
f_8	mean	1.007×10^{3} =	1.089×10^{3} +	1.315×10^{3} +	1.121×10^{3} +	1.157×10^{3} +	1.050×10^{3} +	1.071×10^{3} +	1.317×10^{3} +	1.268×10^{3} +	1.242×10^{3} +	1.316×10^{3} +	1.690×10^{3} +	1.416×10^{3} +	1.079×10^{3} +	**9.975×10^{2}**
	std	3.837×10^{1}	4.527×10^{1}	2.234×10^{1}	1.870×10^{1}	1.539×10^{1}	5.348×10^{1}	3.666×10^{1}	8.919×10^{1}	6.989×10^{1}	2.774×10^{1}	2.804×10^{1}	4.415×10^{1}	4.029×10^{1}	5.241×10^{1}	5.093×10^{1}
f_9	mean	3.761×10^{3} =	1.284×10^{4} +	1.301×10^{4} +	1.392×10^{4} +	**9.029×10^{2}** −	4.347×10^{3} =	3.613×10^{3} =	2.263×10^{4} +	2.513×10^{4} +	5.807×10^{3} +	1.343×10^{4} +	5.749×10^{4} +	1.144×10^{3} −	1.792×10^{3} −	4.152×10^{3}
	std	1.202×10^{3}	6.683×10^{3}	2.688×10^{3}	2.219×10^{3}	5.763	4.168×10^{3}	3.560×10^{3}	6.566×10^{3}	4.965×10^{3}	2.082×10^{3}	2.248×10^{3}	5.399×10^{3}	2.114×10^{3}	3.589×10^{2}	2.960×10^{3}
f_{10}	mean	8.320×10^{3} =	**7.918×10^{3}** −	1.511×10^{4} +	1.108×10^{4} =	1.437×10^{4} +	1.137×10^{4} =	1.051×10^{4} =	1.132×10^{4} =	1.161×10^{4} =	1.521×10^{4} +	1.476×10^{4} +	1.514×10^{4} +	1.632×10^{4} =	9.285×10^{3} =	1.021×10^{4}
	std	2.632×10^{3}	1.684×10^{3}	3.437×10^{2}	4.030×10^{2}	3.669×10^{2}	1.499×10^{3}	4.672×10^{2}	1.631×10^{3}	1.228×10^{3}	3.178×10^{2}	6.249×10^{2}	4.585×10^{2}	7.773×10^{2}	1.534×10^{3}	1.956×10^{3}
f_{11}	mean	9.682×10^{3} +	1.880×10^{3} =	1.632×10^{3} +	2.656×10^{3} +	1.465×10^{3} +	4.400×10^{3} =	2.333×10^{3} =	3.400×10^{3} +	3.707×10^{3} +	1.150×10^{4} +	4.616×10^{3} +	3.027×10^{4} +	8.932×10^{3} +	1.901×10^{3} +	**1.409×10^{3}**
	std	3.665×10^{3}	9.793×10^{2}	5.658×10^{1}	3.399×10^{2}	4.735×10^{1}	3.503×10^{3}	1.798×10^{3}	1.584×10^{3}	9.108×10^{2}	2.747×10^{3}	6.691×10^{2}	4.444×10^{3}	1.350×10^{3}	1.678×10^{2}	6.838×10^{1}
f_{12}	mean	3.782×10^{9} +	6.266×10^{6} −	2.518×10^{8} +	2.023×10^{7} +	4.383×10^{6} −	4.968×10^{7} =	2.415×10^{7} +	1.174×10^{9} +	1.189×10^{9} +	2.453×10^{8} +	1.703×10^{9} +	5.443×10^{10} +	**6.063×10^{3}** −	3.489×10^{7} +	1.897×10^{7}
	std	2.252×10^{9}	4.476×10^{6}	4.979×10^{7}	5.214×10^{6}	2.426×10^{6}	8.293×10^{7}	4.165×10^{7}	1.052×10^{9}	1.295×10^{9}	1.307×10^{8}	3.772×10^{8}	9.427×10^{9}	9.228×10^{8}	1.215×10^{7}	1.105×10^{7}
f_{13}	mean	1.281×10^{9} +	**1.130×10^{4}** −	2.070×10^{5} +	2.165×10^{5} +	6.728×10^{4} =	6.160×10^{6} =	1.991×10^{6} =	7.586×10^{6} +	1.763×10^{8} +	1.949×10^{6} +	3.110×10^{7} +	2.281×10^{10} +	1.981×10^{9} +	1.144×10^{5} +	5.377×10^{4}
	std	1.325×10^{9}	8.227×10^{3}	5.749×10^{4}	2.556×10^{5}	6.765×10^{4}	9.831×10^{6}	4.184×10^{6}	1.649×10^{7}	5.396×10^{8}	1.151×10^{6}	2.803×10^{7}	3.496×10^{9}	1.204×10^{8}	7.498×10^{4}	2.476×10^{4}
f_{14}	mean	3.202×10^{6} +	1.937×10^{5} +	**1.618×10^{3}** −	1.489×10^{6} +	1.800×10^{5} +	9.682×10^{5} =	5.500×10^{4} +	7.531×10^{5} +	1.321×10^{6} +	1.370×10^{6} +	2.380×10^{5} +	1.752×10^{7} +	1.038×10^{8} +	6.788×10^{4} =	3.139×10^{4}
	std	3.210×10^{6}	2.207×10^{5}	1.528×10^{1}	7.945×10^{5}	5.822×10^{4}	1.148×10^{6}	1.254×10^{5}	7.557×10^{5}	1.610×10^{6}	9.996×10^{5}	1.129×10^{5}	6.641×10^{6}	1.956×10^{6}	5.579×10^{4}	2.245×10^{4}
f_{15}	mean	1.743×10^{8} +	1.109×10^{4} −	**3.480×10^{3}** −	1.833×10^{4} −	5.192×10^{4} +	2.293×10^{6} =	8.477×10^{4} =	1.803×10^{5} +	1.019×10^{6} +	1.226×10^{5} +	1.613×10^{5} +	6.133×10^{9} +	1.733×10^{6} =	4.856×10^{4} =	2.777×10^{4}
	std	3.148×10^{8}	9.178×10^{3}	3.393×10^{2}	1.575×10^{4}	3.573×10^{4}	3.393×10^{6}	1.985×10^{5}	5.306×10^{5}	1.385×10^{6}	9.047×10^{4}	1.896×10^{5}	1.541×10^{9}	2.388×10^{6}	3.770×10^{4}	1.178×10^{4}
f_{16}	mean	3.354×10^{3} =	3.486×10^{3} =	5.382×10^{3} +	3.504×10^{3} =	4.504×10^{3} +	3.725×10^{3} =	3.740×10^{3} =	4.997×10^{3} +	5.195×10^{3} +	4.924×10^{3} +	5.261×10^{3} +	8.068×10^{3} +	4.903×10^{5} +	**3.232×10^{3}** =	3.560×10^{3}
	std	2.438×10^{2}	4.236×10^{2}	2.587×10^{2}	2.018×10^{2}	2.849×10^{2}	4.157×10^{2}	4.761×10^{2}	1.102×10^{3}	1.203×10^{3}	2.809×10^{2}	2.348×10^{2}	5.555×10^{2}	5.064×10^{2}	3.870×10^{2}	4.089×10^{2}
f_{17}	mean	3.034×10^{3} =	3.078×10^{3} =	4.231×10^{3} +	2.936×10^{3} −	3.567×10^{3} =	3.179×10^{3} =	3.173×10^{3} =	4.423×10^{3} +	4.204×10^{3} +	3.922×10^{3} +	4.190×10^{3} +	1.031×10^{4} +	4.641×10^{3} +	**2.887×10^{3}** −	3.324×10^{3}
	std	3.432×10^{2}	3.395×10^{2}	1.972×10^{2}	1.541×10^{2}	1.735×10^{2}	2.934×10^{2}	2.191×10^{2}	7.313×10^{2}	5.685×10^{2}	1.776×10^{2}	2.229×10^{2}	3.165×10^{3}	3.318×10^{2}	2.759×10^{2}	3.747×10^{2}
f_{18}	mean	8.339×10^{6} +	3.602×10^{6} +	6.051×10^{4} −	3.360×10^{6} +	2.851×10^{6} +	4.230×10^{6} =	1.552×10^{6} =	3.571×10^{6} +	1.075×10^{7} +	1.129×10^{7} +	3.816×10^{6} +	9.688×10^{7} +	**3.511×10^{3}** −	1.431×10^{6} +	3.295×10^{5}
	std	5.671×10^{6}	4.330×10^{6}	1.765×10^{4}	1.414×10^{6}	7.862×10^{5}	4.984×10^{6}	2.278×10^{6}	3.319×10^{6}	2.421×10^{7}	5.925×10^{6}	1.469×10^{6}	3.088×10^{7}	2.217×10^{6}	1.189×10^{6}	2.436×10^{5}
f_{19}	mean	8.716×10^{7} +	1.331×10^{4} −	**5.428×10^{3}** −	1.914×10^{4} −	3.345×10^{4} −	4.823×10^{5} −	9.914×10^{4} −	8.780×10^{4} −	1.168×10^{6} =	8.749×10^{4} −	3.510×10^{5} −	2.684×10^{9} +	3.448×10^{6} +	2.961×10^{4} −	7.370×10^{5}
	std	1.063×10^{8}	1.166×10^{4}	1.201×10^{3}	2.945×10^{3}	1.514×10^{4}	8.311×10^{5}	2.118×10^{5}	9.167×10^{4}	1.310×10^{6}	5.751×10^{4}	5.014×10^{5}	4.664×10^{8}	1.320×10^{7}	2.424×10^{4}	4.863×10^{5}
f_{20}	mean	**2.637×10^{3}** −	3.122×10^{3} =	4.187×10^{3} +	3.060×10^{3} =	3.775×10^{3} +	3.389×10^{3} =	3.311×10^{3} =	3.807×10^{3} +	3.629×10^{3} +	4.042×10^{3} +	4.071×10^{3} +	4.298×10^{3} +	1.222×10^{7} +	2.880×10^{3} −	3.236×10^{3}
	std	2.011×10^{2}	3.423×10^{2}	1.618×10^{2}	1.761×10^{2}	1.682×10^{2}	2.074×10^{2}	2.331×10^{2}	4.382×10^{2}	3.848×10^{2}	1.612×10^{2}	1.641×10^{2}	1.597×10^{2}	2.245×10^{2}	2.710×10^{2}	2.979×10^{2}
f_{21}	mean	2.505×10^{3} =	2.590×10^{3} +	2.825×10^{3} +	2.610×10^{3} +	2.655×10^{3} +	2.550×10^{3} +	2.564×10^{3} +	2.917×10^{3} +	2.829×10^{3} +	2.735×10^{3} +	2.807×10^{3} +	3.207×10^{3} +	3.225×10^{3} +	2.549×10^{3} +	**2.470×10^{3}**
	std	4.328×10^{1}	6.378×10^{1}	2.378×10^{1}	1.287×10^{1}	1.531×10^{1}	6.198×10^{1}	3.875×10^{1}	1.093×10^{2}	8.435×10^{1}	2.733×10^{1}	1.937×10^{1}	4.868×10^{1}	5.833×10^{1}	6.412×10^{1}	4.457×10^{1}
f_{22}	mean	9.645×10^{3} −	9.349×10^{3} −	1.655×10^{4} +	1.098×10^{4} −	1.574×10^{4} +	1.353×10^{4} =	1.218×10^{4} −	1.288×10^{4} =	1.327×10^{4} =	1.526×10^{4} +	1.619×10^{4} +	1.686×10^{4} +	**2.738×10^{3}** −	9.279×10^{3} −	1.381×10^{4}
	std	2.876×10^{3}	1.261×10^{3}	4.476×10^{2}	2.840×10^{3}	4.738×10^{2}	1.876×10^{3}	1.115×10^{3}	1.879×10^{3}	1.138×10^{3}	2.948×10^{3}	3.593×10^{2}	4.249×10^{2}	9.703×10^{2}	3.675×10^{3}	1.812×10^{3}
f_{23}	mean	3.055×10^{3} +	3.183×10^{3} +	3.320×10^{3} +	3.094×10^{3} +	3.075×10^{3} +	2.987×10^{3} =	3.005×10^{3} =	3.930×10^{3} +	3.810×10^{3} +	3.232×10^{3} +	3.248×10^{3} +	4.165×10^{3} +	1.160×10^{4} +	2.989×10^{3} =	**2.970×10^{3}**
	std	7.265×10^{1}	8.804×10^{1}	4.873×10^{1}	2.895×10^{1}	2.530×10^{1}	5.556×10^{1}	3.880×10^{1}	1.505×10^{2}	1.752×10^{2}	3.633×10^{1}	2.159×10^{1}	1.010×10^{2}	1.382×10^{2}	4.945×10^{1}	8.375×10^{1}
f_{24}	mean	3.279×10^{3} +	3.372×10^{3} +	3.478×10^{3} +	3.325×10^{3} +	3.249×10^{3} +	3.154×10^{3} =	3.168×10^{3} =	4.044×10^{3} +	3.979×10^{3} +	3.396×10^{3} +	3.371×10^{3} +	4.501×10^{3} +	3.495×10^{3} +	**3.116×10^{3}** =	3.153×10^{3}
	std	8.637×10^{1}	9.642×10^{1}	4.289×10^{1}	2.658×10^{1}	1.733×10^{1}	6.048×10^{1}	4.610×10^{1}	2.099×10^{2}	2.169×10^{2}	3.098×10^{1}	1.277×10^{1}	9.365×10^{1}	1.394×10^{2}	5.718×10^{1}	8.055×10^{1}
f_{25}	mean	5.487×10^{3} +	3.136×10^{3} +	3.971×10^{3} +	3.228×10^{3} +	**3.047×10^{3}** −	3.060×10^{3} =	3.052×10^{3} −	4.799×10^{3} +	3.852×10^{3} +	3.793×10^{3} +	4.623×10^{3} +	2.790×10^{4} +	3.762×10^{3} +	3.136×10^{3} +	3.079×10^{3}
	std	1.186×10^{3}	4.938×10^{1}	2.513×10^{2}	2.062×10^{1}	2.046×10^{1}	6.181×10^{1}	3.544×10^{1}	4.823×10^{2}	3.312×10^{2}	3.062×10^{2}	2.635×10^{2}	3.031×10^{3}	3.963×10^{2}	2.976×10^{1}	2.191×10^{1}
f_{26}	mean	7.405×10^{3} +	8.483×10^{3} +	9.295×10^{3} +	7.543×10^{3} +	7.079×10^{3} +	6.127×10^{3} =	6.361×10^{3} =	1.490×10^{4} +	1.350×10^{4} +	8.961×10^{3} +	9.110×10^{3} +	1.943×10^{4} +	**4.824×10^{3}** −	6.217×10^{3} =	6.214×10^{3}
	std	8.369×10^{2}	1.474×10^{3}	3.551×10^{2}	2.940×10^{2}	1.742×10^{2}	6.601×10^{2}	4.767×10^{2}	1.619×10^{3}	1.579×10^{3}	5.198×10^{2}	2.224×10^{2}	1.118×10^{3}	2.032×10^{3}	4.107×10^{2}	7.513×10^{2}
f_{27}	mean	3.805×10^{3} +	3.915×10^{3} +	3.463×10^{3} −	3.717×10^{3} +	**3.291×10^{3}** −	3.373×10^{3} −	3.368×10^{3} −	5.257×10^{3} +	4.339×10^{3} +	3.714×10^{3} =	3.426×10^{3} −	5.891×10^{3} +	4.156×10^{3} +	3.392×10^{3} −	3.651×10^{3}
	std	1.337×10^{2}	2.254×10^{2}	6.644×10^{1}	3.373×10^{1}	4.301×10^{1}	8.992×10^{1}	8.043×10^{1}	7.196×10^{2}	2.915×10^{2}	9.969×10^{1}	8.309×10^{1}	2.924×10^{2}	2.752×10^{2}	4.093×10^{1}	1.466×10^{2}
f_{28}	mean	6.027×10^{3} +	3.521×10^{3} +	3.952×10^{3} +	3.865×10^{3} +	**3.327×10^{3}** =	3.421×10^{3} =	3.332×10^{3} =	5.555×10^{3} +	4.542×10^{3} +	4.752×10^{3} +	7.764×10^{3} +	1.461×10^{4} +	4.274×10^{3} +	3.422×10^{3} +	3.337×10^{3}
	std	6.986×10^{2}	1.477×10^{2}	2.256×10^{2}	7.520×10^{1}	1.936×10^{1}	2.270×10^{2}	6.779×10^{1}	7.192×10^{2}	4.897×10^{2}	4.035×10^{2}	9.290×10^{2}	8.058×10^{2}	4.615×10^{2}	3.920×10^{1}	2.315×10^{1}
f_{29}	mean	4.910×10^{3} =	4.588×10^{3} −	5.981×10^{3} +	4.674×10^{3} −	4.838×10^{3} =	**4.159×10^{3}** −	4.366×10^{3} −	8.600×10^{3} +	7.903×10^{3} +	5.730×10^{3} +	5.854×10^{3} +	1.635×10^{4} +	5.189×10^{3} +	4.311×10^{3} −	5.069×10^{3}
	std	6.832×10^{2}	3.149×10^{2}	1.969×10^{2}	2.252×10^{2}	2.319×10^{2}	3.304×10^{2}	2.793×10^{2}	1.485×10^{3}	1.204×10^{3}	3.404×10^{2}	2.392×10^{2}	3.122×10^{3}	1.178×10^{3}	3.215×10^{2}	5.697×10^{2}
f_{30}	mean	1.269×10^{8} +	1.489×10^{7} −	9.597×10^{6} −	1.186×10^{7} −	6.179×10^{6} −	5.704×10^{6} −	6.650×10^{6} −	3.200×10^{7} −	3.723×10^{7} −	4.696×10^{7} =	2.995×10^{7} −	4.206×10^{9} +	**7.968×10^{3}** −	9.190×10^{6} −	4.859×10^{7}
	std	1.151×10^{8}	8.588×10^{6}	1.812×10^{6}	2.778×10^{6}	2.861×10^{6}	6.444×10^{6}	8.738×10^{6}	1.917×10^{7}	3.147×10^{7}	1.699×10^{7}	1.581×10^{7}	8.596×10^{8}	1.085×10^{8}	4.667×10^{6}	8.765×10^{6}
+ /=/−		18/9/2	16/5/8	23/0/6	19/3/7	16/5/8	4/20/5	6/15/8	25/2/2	25/3/1	25/3/1	26/0/3	29/0/0	24/4/1	13/7/9	–
Avg. Rank		7.6	5.9	8.5	5.5	4.6	4.6	3.4	10.5	10.4	8.6	10.3	14.4	9.3	3.4	**3.1**

such as DE, PSO, and CMA-ES exhibit diminished performance at this level. In contrast, the CIA algorithm performs admirably, achieving an average ranking of 2.6 on the 100D benchmark of CEC2017. Although the CIA does not achieve the highest average ranking, it still outperforms high-performance algorithms such as JADE, L-SHADE, and SaDE, demonstrating strong competitiveness in tackling high-dimensional optimization problems.

In real-world engineering optimization problems, such as the HPA test, the performance advantage of the CIA algorithm becomes even more pronounced. When tackling engineering challenges, particularly those involving constraints, the CIA consistently outperforms high-performing algorithms such as JADE, L-SHADE, and SaDE. Under unconstrained conditions, the CIA achieves an average ranking 1.7, maintaining a clear lead over all comparative algorithms. Under constrained conditions, its performance is even more remarkable, with an average ranking of 1.6, placing it definitively ahead of the other algorithms in more than two-thirds of the optimizations.

Through this series of experiments, the CIA algorithm demonstrates outstanding performance on the CEC benchmark suite and exceptional performance on real-world optimization problems. These results further underscore the proposed CIA algorithm's high performance and robustness.

6.5. Potential analyses and open issues

Although a series of experiments have validated the high performance and robustness of the CIA algorithm, areas warrant further improvement. We outline the following points for potential enhancement to support future research on the CIA.

1. In the current implementation of the CIA algorithm, the population sizes of the crested ibis and fish populations are manually fixed. A promising direction for future research is the development of adaptive methods to dynamically adjust these population sizes, which could further enhance the algorithm's performance.

2. Compared to simple optimization problems in low dimensions, the CIA algorithm performs comparatively lower than in high-dimensional scenarios, primarily due to its slower convergence speed. Investigating methods to enhance convergence speed in such cases represents a valuable direction for future research.

3. Another promising research direction involves modifying or expanding the updating strategies for the crested ibis and fish populations to enhance the algorithm's performance further.

4. Given that the CIA algorithm inherently employs a dual-population mechanism, a promising research direction would be to leverage this feature more effectively to adapt the CIA for solving multi-objective optimization problems.

Naturally, there are additional research directions beyond those mentioned above. For instance, adapting the CIA algorithm to tackle

Table 16
CEC2017-100D.

FunNum.		DE	PSO	CMA-ES	CL-PSO	SaDE	JADE	L-SHADE	WOA	PPSO	MBGO	COA	FATA	HO	MSA	CIA
$f1$	mean	9.836×10^{10} +	2.675×10^{9} +	8.873×10^{10} +	4.365×10^{8} +	8.715×10^{3} =	5.979×10^{5} =	8.349×10^{3} =	7.483×10^{10} +	1.954×10^{10} +	5.018×10^{10} +	6.761×10^{10} +	4.366×10^{11} +	5.102×10^{10} +	2.373×10^{9} +	**5.224×10^{3}**
	std	2.307×10^{10}	3.273×10^{9}	1.280×10^{10}	1.428×10^{8}	8.814×10^{3}	3.036×10^{6}	9.109×10^{3}	1.460×10^{10}	4.477×10^{9}	7.212×10^{9}	4.827×10^{9}	2.252×10^{10}	5.616×10^{9}	5.491×10^{8}	4.652×10^{3}
$f3$	mean	6.019×10^{5} +	4.976×10^{5} +	6.560×10^{5} +	4.634×10^{5} +	5.643×10^{5} +	3.881×10^{5} =	3.253×10^{5} =	3.790×10^{5} +	4.457×10^{5} +	3.584×10^{5} +	7.879×10^{5} +	6.729×10^{5} +	2.834×10^{5} +	6.700×10^{5} +	**1.820×10^{5}**
	std	6.523×10^{4}	5.648×10^{4}	5.861×10^{4}	3.468×10^{4}	4.513×10^{4}	2.181×10^{5}	2.947×10^{5}	9.514×10^{4}	8.603×10^{4}	2.355×10^{4}	6.228×10^{4}	6.037×10^{4}	1.705×10^{4}	1.409×10^{5}	3.157×10^{4}
$f4$	mean	1.339×10^{4} +	1.263×10^{3} +	8.970×10^{3} +	9.796×10^{2} +	6.974×10^{2} =	7.089×10^{2} =	**6.595×10^{2}** −	9.857×10^{3} +	3.920×10^{3} +	7.330×10^{3} +	8.679×10^{3} +	1.460×10^{5} +	7.722×10^{3} +	1.162×10^{3} +	7.177×10^{2}
	std	4.482×10^{3}	3.060×10^{2}	2.264×10^{3}	3.253×10^{1}	4.262×10^{1}	1.174×10^{2}	4.648×10^{1}	3.220×10^{3}	1.144×10^{3}	1.154×10^{3}	1.037×10^{3}	1.180×10^{4}	1.383×10^{3}	8.350×10^{1}	3.657×10^{1}
$f5$	mean	1.261×10^{3} +	1.242×10^{3} +	1.698×10^{3} +	1.344×10^{3} +	1.330×10^{3} +	1.116×10^{3} +	1.114×10^{3} +	1.635×10^{3} +	1.637×10^{3} +	1.637×10^{3} +	1.673×10^{3} +	2.610×10^{3} +	1.442×10^{3} +	1.238×10^{3} +	**9.215×10^{2}**
	std	1.088×10^{2}	9.950×10^{1}	3.952×10^{1}	2.623×10^{1}	3.139×10^{1}	1.558×10^{2}	9.794×10^{1}	1.669×10^{2}	7.451×10^{1}	5.065×10^{1}	3.355×10^{1}	5.300×10^{1}	4.162×10^{1}	9.520×10^{1}	8.719×10^{1}
$f6$	mean	6.316×10^{2} =	6.478×10^{2} +	6.720×10^{2} +	6.094×10^{2} −	**6.000×10^{2}** −	6.053×10^{2} −	6.026×10^{2} −	6.898×10^{2} +	6.836×10^{2} +	6.499×10^{2} +	6.439×10^{2} +	7.344×10^{2} +	6.713×10^{2} +	6.201×10^{2} −	6.302×10^{2}
	std	5.562	5.702	6.442	1.092	6.111×10^{-3}	4.952	4.416	8.103	5.043	4.190	2.419	3.598	3.985	2.673	1.136×10^{1}
$f7$	mean	2.961×10^{3} +	3.541×10^{3} +	2.743×10^{3} +	1.651×10^{3} +	1.661×10^{3} +	1.455×10^{3} +	1.468×10^{3} +	3.454×10^{3} +	4.010×10^{3} +	2.364×10^{3} +	3.231×10^{3} +	1.037×10^{4} +	2.671×10^{3} +	1.806×10^{3} +	**1.334×10^{3}**
	std	4.426×10^{2}	3.707×10^{2}	1.401×10^{2}	2.905×10^{1}	2.688×10^{1}	1.282×10^{2}	9.395×10^{1}	1.355×10^{2}	2.633×10^{2}	8.534×10^{1}	1.230×10^{2}	4.450×10^{2}	1.456×10^{2}	5.339×10^{1}	1.019×10^{2}
$f8$	mean	1.532×10^{3} +	1.572×10^{3} +	2.020×10^{3} +	1.638×10^{3} +	1.626×10^{3} +	1.403×10^{3} +	1.409×10^{3} +	2.026×10^{3} +	2.057×10^{3} +	1.968×10^{3} +	1.980×10^{3} +	2.995×10^{3} +	1.869×10^{3} +	1.495×10^{3} +	**1.226×10^{3}**
	std	9.204×10^{1}	1.242×10^{2}	3.883×10^{1}	2.467×10^{1}	2.165×10^{1}	1.684×10^{2}	9.362×10^{1}	9.754×10^{1}	1.190×10^{2}	5.224×10^{1}	3.433×10^{1}	7.094×10^{1}	6.365×10^{1}	1.091×10^{2}	9.066×10^{1}
$f9$	mean	1.717×10^{4} =	4.767×10^{4} +	4.557×10^{4} +	4.443×10^{4} +	**1.051×10^{3}** −	1.606×10^{4} =	1.355×10^{4} =	4.194×10^{4} +	5.987×10^{4} +	3.532×10^{4} +	4.061×10^{4} +	1.532×10^{5} +	2.980×10^{4} +	1.503×10^{4} =	1.401×10^{4}
	std	4.230×10^{3}	2.810×10^{4}	6.321×10^{3}	3.940×10^{3}	2.459×10^{2}	1.972×10^{4}	1.499×10^{4}	1.074×10^{4}	1.170×10^{4}	3.915×10^{3}	4.556×10^{3}	7.863×10^{3}	1.924×10^{3}	7.078×10^{3}	1.009×10^{4}
$f10$	mean	2.288×10^{4} =	**1.782×10^{4}** −	3.235×10^{4} +	2.644×10^{4} +	3.103×10^{4} +	2.630×10^{4} +	2.236×10^{4} =	2.491×10^{4} =	2.700×10^{4} +	3.232×10^{4} +	3.171×10^{4} +	3.225×10^{4} +	1.934×10^{4} −	1.913×10^{4} −	2.283×10^{4}
	std	5.629×10^{3}	4.402×10^{3}	5.774×10^{2}	4.535×10^{2}	4.120×10^{2}	3.087×10^{3}	6.745×10^{2}	3.130×10^{3}	2.309×10^{3}	7.933×10^{2}	5.144×10^{2}	5.017×10^{2}	1.096×10^{3}	1.574×10^{3}	3.571×10^{3}
$f11$	mean	8.112×10^{4} +	7.980×10^{4} +	1.180×10^{4} +	3.821×10^{4} +	7.127×10^{4} +	4.513×10^{4} =	3.405×10^{4} =	5.707×10^{4} +	6.118×10^{4} +	1.501×10^{5} +	1.678×10^{5} +	2.868×10^{5} +	7.660×10^{4} +	8.169×10^{4} +	**4.634×10^{3}**
	std	2.266×10^{4}	2.206×10^{4}	3.940×10^{3}	5.123×10^{3}	1.231×10^{4}	4.194×10^{4}	3.401×10^{4}	1.950×10^{4}	2.418×10^{4}	2.458×10^{4}	2.595×10^{4}	2.586×10^{4}	1.089×10^{4}	1.666×10^{4}	6.635×10^{2}
$f12$	mean	2.199×10^{10} +	2.013×10^{8} −	3.810×10^{9} +	1.269×10^{8} −	1.008×10^{7} −	1.222×10^{8} −	**6.291×10^{6}** −	1.213×10^{10} +	3.567×10^{9} +	5.856×10^{9} +	1.326×10^{10} +	2.027×10^{11} +	1.350×10^{10} +	3.764×10^{8} +	2.087×10^{8}
	std	8.979×10^{9}	6.325×10^{8}	1.572×10^{9}	2.118×10^{7}	7.421×10^{6}	2.132×10^{8}	7.926×10^{6}	9.168×10^{9}	2.421×10^{9}	2.051×10^{9}	1.715×10^{9}	1.127×10^{10}	4.050×10^{9}	1.022×10^{8}	4.379×10^{7}
$f13$	mean	4.132×10^{9} +	1.428×10^{4} −	9.522×10^{6} +	8.447×10^{3} −	**6.721×10^{3}** −	6.321×10^{5} =	1.552×10^{4} −	4.214×10^{8} +	1.301×10^{8} +	6.412×10^{7} +	4.470×10^{8} +	4.434×10^{10} +	2.647×10^{8} +	2.543×10^{5} +	3.819×10^{4}
	std	2.428×10^{9}	1.074×10^{4}	2.164×10^{6}	9.308×10^{2}	4.768×10^{3}	1.326×10^{6}	2.045×10^{4}	8.541×10^{8}	2.281×10^{8}	5.202×10^{7}	9.167×10^{7}	4.634×10^{9}	1.649×10^{8}	9.401×10^{4}	1.144×10^{4}
$f14$	mean	1.530×10^{7} +	2.129×10^{6} +	**2.640×10^{3}** −	5.034×10^{6} +	2.981×10^{6} +	7.362×10^{6} =	3.939×10^{5} −	1.997×10^{6} +	5.927×10^{6} +	1.468×10^{7} +	5.130×10^{6} +	1.105×10^{8} +	4.296×10^{6} +	2.937×10^{6} +	4.170×10^{5}
	std	1.282×10^{7}	1.145×10^{6}	1.656×10^{2}	1.198×10^{6}	1.353×10^{6}	8.174×10^{6}	1.081×10^{6}	9.142×10^{5}	2.449×10^{6}	5.481×10^{6}	1.004×10^{6}	2.197×10^{7}	2.271×10^{6}	1.257×10^{6}	1.493×10^{5}
$f15$	mean	1.191×10^{9} +	7.080×10^{3} −	4.528×10^{4} +	**2.471×10^{3}** −	6.094×10^{3} −	5.697×10^{5} =	8.005×10^{4} +	1.369×10^{6} +	1.380×10^{7} +	1.933×10^{6} +	1.571×10^{7} +	1.794×10^{10} +	4.890×10^{7} +	4.503×10^{4} +	2.950×10^{4}
	std	9.472×10^{8}	7.000×10^{3}	8.572×10^{3}	3.187×10^{2}	3.933×10^{3}	1.077×10^{6}	1.831×10^{5}	3.210×10^{6}	4.644×10^{7}	1.022×10^{6}	1.411×10^{7}	2.483×10^{9}	1.080×10^{8}	1.803×10^{4}	7.649×10^{3}
$f16$	mean	6.765×10^{3} =	**5.912×10^{3}** =	1.107×10^{4} +	7.139×10^{3} +	9.118×10^{3} +	6.791×10^{3} =	6.940×10^{3} +	1.095×10^{4} +	1.107×10^{4} +	1.062×10^{4} +	1.093×10^{4} +	2.097×10^{4} +	1.115×10^{4} +	5.949×10^{3} =	6.137×10^{3}
	std	7.573×10^{2}	6.839×10^{2}	3.792×10^{2}	3.316×10^{2}	3.931×10^{2}	9.315×10^{2}	9.046×10^{2}	3.992×10^{3}	2.346×10^{3}	5.904×10^{2}	3.390×10^{2}	1.322×10^{3}	1.093×10^{3}	8.021×10^{2}	7.503×10^{2}
$f17$	mean	5.840×10^{3} +	5.281×10^{3} =	7.833×10^{3} +	**4.904×10^{3}** =	6.517×10^{3} +	5.608×10^{3} +	5.299×10^{3} =	9.977×10^{3} +	1.159×10^{4} +	7.554×10^{3} +	8.177×10^{3} +	1.097×10^{6} +	7.256×10^{3} +	5.348×10^{3} =	4.955×10^{3}
	std	1.513×10^{3}	4.562×10^{2}	2.999×10^{2}	2.483×10^{2}	2.127×10^{2}	6.000×10^{2}	6.290×10^{2}	2.713×10^{3}	5.248×10^{3}	2.945×10^{2}	2.798×10^{2}	4.855×10^{5}	8.418×10^{2}	4.106×10^{2}	5.009×10^{2}
$f18$	mean	1.452×10^{7} +	7.914×10^{6} +	**5.123×10^{5}** −	4.781×10^{6} +	4.337×10^{6} +	8.941×10^{6} =	1.660×10^{6} =	2.607×10^{6} +	7.947×10^{6} +	1.937×10^{7} +	1.332×10^{7} +	1.951×10^{8} +	6.715×10^{6} +	3.777×10^{6} +	7.637×10^{5}
	std	9.277×10^{6}	6.175×10^{6}	1.429×10^{5}	1.442×10^{6}	1.323×10^{6}	1.012×10^{7}	3.194×10^{6}	1.102×10^{6}	3.288×10^{6}	9.072×10^{6}	3.097×10^{6}	4.604×10^{7}	2.145×10^{6}	1.605×10^{6}	3.337×10^{5}
$f19$	mean	1.015×10^{9} +	1.137×10^{4} −	4.284×10^{5} −	**2.777×10^{3}** −	4.012×10^{3} −	1.066×10^{6} −	3.359×10^{4} −	2.023×10^{7} =	9.504×10^{7} +	6.795×10^{6} +	1.919×10^{7} +	1.847×10^{10} +	4.602×10^{7} +	3.264×10^{4} −	2.433×10^{6}
	std	8.911×10^{8}	9.729×10^{3}	1.493×10^{5}	4.564×10^{2}	1.753×10^{3}	2.005×10^{6}	1.071×10^{5}	5.077×10^{7}	2.080×10^{8}	3.757×10^{6}	1.919×10^{7}	2.160×10^{9}	3.139×10^{7}	2.101×10^{4}	1.300×10^{6}
$f20$	mean	**4.492×10^{3}** −	5.279×10^{3} −	7.582×10^{3} +	5.374×10^{3} −	6.964×10^{3} +	6.250×10^{3} =	5.854×10^{3} =	6.514×10^{3} =	6.603×10^{3} =	7.649×10^{3} +	7.551×10^{3} +	7.930×10^{3} +	5.575×10^{3} =	4.894×10^{3} −	6.040×10^{3}
	std	3.482×10^{2}	5.981×10^{2}	2.926×10^{2}	2.951×10^{2}	2.785×10^{2}	4.460×10^{2}	3.099×10^{2}	6.917×10^{2}	5.061×10^{2}	2.248×10^{2}	2.706×10^{2}	2.358×10^{2}	4.676×10^{2}	4.375×10^{2}	7.677×10^{2}
$f21$	mean	3.089×10^{3} +	3.155×10^{3} +	3.631×10^{3} +	3.132×10^{3} +	3.158×10^{3} +	2.932×10^{3} +	2.931×10^{3} +	4.039×10^{3} +	3.972×10^{3} +	3.521×10^{3} +	3.519×10^{3} +	4.697×10^{3} +	3.844×10^{3} +	3.014×10^{3} +	**2.722×10^{3}**
	std	7.246×10^{1}	1.101×10^{2}	5.987×10^{1}	3.391×10^{1}	1.839×10^{1}	1.694×10^{2}	9.794×10^{1}	1.720×10^{2}	1.502×10^{2}	3.443×10^{1}	3.799×10^{1}	7.335×10^{1}	1.915×10^{2}	1.007×10^{2}	9.980×10^{1}
$f22$	mean	2.187×10^{4} −	**2.057×10^{4}** −	3.425×10^{4} +	2.939×10^{4} =	3.310×10^{4} +	2.941×10^{4} =	2.461×10^{4} −	2.894×10^{4} =	2.961×10^{4} =	3.482×10^{4} +	3.370×10^{4} +	3.485×10^{4} +	2.349×10^{4} −	2.276×10^{4} −	2.768×10^{4}
	std	4.213×10^{3}	4.470×10^{3}	5.231×10^{2}	6.302×10^{2}	7.012×10^{2}	3.000×10^{3}	5.314×10^{2}	3.463×10^{3}	1.972×10^{3}	6.701×10^{2}	4.627×10^{2}	4.791×10^{2}	1.184×10^{3}	4.200×10^{3}	4.181×10^{3}
$f23$	mean	3.658×10^{3} +	3.796×10^{3} +	4.377×10^{3} +	3.525×10^{3} +	3.616×10^{3} +	**3.275×10^{3}** −	3.332×10^{3} =	5.458×10^{3} +	5.273×10^{3} +	4.148×10^{3} +	3.951×10^{3} +	6.243×10^{3} +	4.880×10^{3} +	3.407×10^{3} =	3.387×10^{3}
	std	1.693×10^{2}	1.374×10^{2}	9.156×10^{1}	3.143×10^{1}	2.081×10^{1}	9.485×10^{1}	6.950×10^{1}	2.853×10^{2}	3.205×10^{2}	8.952×10^{1}	2.755×10^{1}	1.507×10^{2}	3.458×10^{2}	7.761×10^{1}	1.343×10^{2}
$f24$	mean	4.667×10^{3} +	4.920×10^{3} +	5.091×10^{3} +	4.086×10^{3} +	4.085×10^{3} +	3.839×10^{3} =	3.842×10^{3} =	7.492×10^{3} +	7.261×10^{3} +	4.909×10^{3} +	4.510×10^{3} +	9.923×10^{3} +	6.827×10^{3} +	3.973×10^{3} +	**3.789×10^{3}**
	std	2.474×10^{2}	2.633×10^{2}	1.710×10^{2}	3.725×10^{1}	3.916×10^{1}	1.759×10^{2}	1.237×10^{2}	5.957×10^{2}	7.890×10^{2}	9.411×10^{1}	3.423×10^{1}	3.531×10^{2}	4.973×10^{2}	9.600×10^{1}	9.597×10^{1}
$f25$	mean	1.296×10^{4} +	3.737×10^{3} +	8.683×10^{3} +	3.831×10^{3} +	3.359×10^{3} −	3.383×10^{3} −	**3.301×10^{3}** −	8.076×10^{3} +	5.440×10^{3} +	7.435×10^{3} +	1.139×10^{4} +	7.503×10^{4} +	7.137×10^{3} +	3.954×10^{3} +	3.405×10^{3}
	std	2.619×10^{3}	2.038×10^{2}	6.708×10^{2}	4.994×10^{1}	6.032×10^{1}	1.688×10^{2}	7.106×10^{1}	1.272×10^{3}	4.813×10^{2}	7.304×10^{2}	9.436×10^{2}	3.946×10^{3}	5.601×10^{2}	1.123×10^{2}	4.960×10^{1}
$f26$	mean	1.935×10^{4} +	2.152×10^{4} +	2.233×10^{4} +	1.528×10^{4} +	1.389×10^{4} +	1.118×10^{4} =	1.174×10^{4} =	3.577×10^{4} +	3.337×10^{4} +	2.186×10^{4} +	1.894×10^{4} +	5.777×10^{4} +	3.072×10^{4} +	1.271×10^{4} +	**1.052×10^{4}**
	std	2.633×10^{3}	2.793×10^{3}	1.295×10^{3}	4.614×10^{2}	3.883×10^{2}	1.991×10^{3}	2.883×10^{3}	3.711×10^{3}	3.394×10^{3}	1.184×10^{3}	4.478×10^{2}	2.461×10^{3}	2.413×10^{3}	7.640×10^{2}	1.022×10^{3}
$f27$	mean	4.494×10^{3} +	4.476×10^{3} +	3.838×10^{3} −	3.822×10^{3} −	**3.427×10^{3}** −	3.532×10^{3} −	3.507×10^{3} −	7.197×10^{3} +	5.727×10^{3} +	4.287×10^{3} +	3.620×10^{3} −	1.065×10^{4} +	5.228×10^{3} +	3.617×10^{3} −	3.934×10^{3}
	std	3.248×10^{2}	2.411×10^{2}	1.016×10^{2}	3.138×10^{1}	3.292×10^{1}	9.032×10^{1}	9.449×10^{1}	1.244×10^{3}	9.097×10^{2}	1.687×10^{2}	7.007×10^{1}	6.757×10^{2}	4.743×10^{2}	5.594×10^{1}	1.205×10^{2}
$f28$	mean	1.633×10^{4} +	5.060×10^{3} +	1.016×10^{4} +	4.977×10^{3} +	3.479×10^{3} =	3.776×10^{3} =	**3.454×10^{3}** =	9.708×10^{3} +	7.166×10^{3} +	1.187×10^{4} +	1.670×10^{4} +	4.726×10^{4} +	9.405×10^{3} +	4.369×10^{3} +	3.488×10^{3}
	std	1.932×10^{3}	1.239×10^{3}	1.860×10^{3}	1.507×10^{2}	3.470×10^{1}	8.268×10^{2}	5.712×10^{1}	1.599×10^{3}	8.730×10^{2}	1.444×10^{3}	5.376×10^{2}	2.570×10^{3}	1.052×10^{3}	2.620×10^{2}	3.667×10^{1}
$f29$	mean	9.999×10^{3} +	7.720×10^{3} −	1.095×10^{4} +	8.883×10^{3} =	8.549×10^{3} =	**6.902×10^{3}** −	7.018×10^{3} −	1.921×10^{4} +	1.745×10^{4} +	1.116×10^{4} +	1.108×10^{4} +	2.997×10^{5} +	1.501×10^{4} +	7.473×10^{3} −	8.568×10^{3}
	std	1.504×10^{3}	8.085×10^{2}	2.844×10^{2}	3.707×10^{2}	4.794×10^{2}	5.486×10^{2}	4.071×10^{2}	6.353×10^{3}	4.590×10^{3}	6.951×10^{2}	6.435×10^{2}	1.004×10^{5}	1.872×10^{3}	7.210×10^{2}	6.163×10^{2}
$f30$	mean	2.477×10^{9} +	2.671×10^{6} −	8.748×10^{6} −	5.787×10^{6} −	**2.367×10^{4}** −	2.542×10^{5} −	1.093×10^{5} −	1.160×10^{8} +	3.578×10^{8} +	1.561×10^{8} +	8.184×10^{7} +	2.938×10^{10} +	1.091×10^{9} +	2.543×10^{6} −	3.057×10^{7}
	std	1.611×10^{9}	2.598×10^{6}	2.322×10^{6}	2.248×10^{6}	1.020×10^{4}	3.639×10^{5}	1.882×10^{5}	1.399×10^{8}	7.577×10^{8}	6.590×10^{7}	2.638×10^{7}	3.841×10^{9}	3.911×10^{8}	1.379×10^{6}	1.401×10^{7}
+ /=/−		23/3/2	18/2/9	24/0/5	18/3/8	16/4/9	6/15/8	6/12/11	25/4/0	27/2/0	29/0/0	28/0/1	29/0/0	26/1/2	17/4/8	–
Avg. Rank		8.8	5.5	9.2	4.9	4.4	4.0	**2.0**	9.8	10.3	9.4	10.3	14.3	8.6	4.4	2.6

Table 17
CEC2022-10D.

FunNum.		DE	PSO	CMA-ES	CL-PSO	SaDE	JADE	L-SHADE	WOA	PPSO	MBGO	COA	FATA	HO	MSA	CIA
$f1$	mean	2.232×10^{3} +	3.000×10^{2} =	4.235×10^{2} +	1.647×10^{3} +	3.669×10^{2} +	1.063×10^{3} +	1.099×10^{3} +	3.481×10^{3} +	4.276×10^{2} +	**3.000×10^{2}** −	1.787×10^{3} +	8.018×10^{3} +	5.182×10^{2} +	1.489×10^{3} +	3.000×10^{2}
	std	1.120×10^{3}	3.821×10^{-5}	3.547×10^{1}	6.176×10^{2}	2.551×10^{1}	1.160×10^{3}	1.184×10^{3}	2.039×10^{3}	1.724×10^{2}	0.000	5.604×10^{2}	1.766×10^{3}	1.577×10^{2}	5.128×10^{2}	1.449×10^{-4}
$f2$	mean	4.475×10^{2} +	**4.074×10^{2}** =	4.165×10^{2} +	4.437×10^{2} +	4.085×10^{2} +	4.091×10^{2} +	4.086×10^{2} =	4.606×10^{2} +	4.457×10^{2} +	4.075×10^{2} =	4.149×10^{2} +	7.700×10^{2} +	4.554×10^{2} +	4.234×10^{2} +	4.077×10^{2}
	std	2.281×10^{1}	1.427×10^{1}	3.113	1.257×10^{1}	1.581	3.608	5.559	7.425×10^{1}	3.165×10^{1}	1.364×10^{1}	3.530	1.285×10^{2}	3.558×10^{1}	1.008×10^{1}	1.722×10^{1}
$f3$	mean	6.000×10^{2} +	6.000×10^{2} +	6.000×10^{2} +	6.000×10^{2} +	6.000×10^{2} +	6.000×10^{2} +	6.000×10^{2} +	6.000×10^{2} +	6.000×10^{2} +	6.000×10^{2} =	6.000×10^{2} =	6.002×10^{2} +	6.000×10^{2} +	6.000×10^{2} =	**6.000×10^{2}**
	std	6.293×10^{-3}	3.116×10^{-12}	6.805×10^{-4}	2.767×10^{-4}	1.494×10^{-7}	4.193×10^{-5}	7.087×10^{-5}	6.162×10^{-3}	1.268×10^{-3}	0.000	0.000	7.017×10^{-2}	7.671×10^{-3}	0.000	0.000
$f4$	mean	8.003×10^{2} =	**8.002×10^{2}** =	8.010×10^{2} +	8.005×10^{2} +	8.008×10^{2} +	8.007×10^{2} +	8.007×10^{2} +	8.004×10^{2} =	8.003×10^{2} =	8.003×10^{2} =	8.010×10^{2} +	8.010×10^{2} +	8.004×10^{2} =	8.009×10^{2} +	8.003×10^{2}
	std	9.391×10^{-2}	9.912×10^{-2}	2.193×10^{-1}	1.576×10^{-1}	1.965×10^{-1}	2.398×10^{-1}	2.380×10^{-1}	1.830×10^{-1}	1.543×10^{-1}	4.714×10^{-1}	0.000	1.934×10^{-1}	1.164×10^{-1}	3.000×10^{-1}	1.936×10^{-1}
$f5$	mean	9.004×10^{2} =	9.006×10^{2} =	9.003×10^{2} =	9.004×10^{2} =	**9.000×10^{2}** −	9.001×10^{2} −	9.001×10^{2} −	9.021×10^{2} +	9.012×10^{2} =	9.000×10^{2} −	9.009×10^{2} =	9.030×10^{2} +	9.007×10^{2} =	9.000×10^{2} −	9.007×10^{2}
	std	2.462×10^{-1}	5.420×10^{-1}	6.862×10^{-2}	1.141×10^{-1}	7.159×10^{-4}	1.514×10^{-1}	1.802×10^{-1}	1.447	7.946×10^{-1}	1.795×10^{-1}	3.399×10^{-1}	7.501×10^{-1}	5.837×10^{-1}	1.795×10^{-1}	5.827×10^{-1}
$f6$	mean	2.727×10^{4} =	1.536×10^{4} −	**4.166×10^{3}** −	3.300×10^{4} =	2.774×10^{4} =	3.781×10^{4} =	2.721×10^{4} =	3.466×10^{4} =	3.401×10^{4} =	2.243×10^{4} =	5.989×10^{4} +	2.846×10^{7} +	3.142×10^{4} =	1.363×10^{5} +	3.500×10^{4}
	std	1.412×10^{4}	8.900×10^{3}	5.843×10^{2}	9.553×10^{3}	8.955×10^{3}	4.364×10^{4}	2.882×10^{4}	1.688×10^{4}	1.994×10^{4}	1.257×10^{4}	1.886×10^{4}	1.778×10^{7}	1.749×10^{4}	6.985×10^{4}	1.497×10^{4}
$f7$	mean	2.033×10^{3} −	**2.033×10^{3}** =	2.080×10^{3} +	2.040×10^{3} =	2.040×10^{3} =	2.040×10^{3} =	2.039×10^{3} =	2.126×10^{3} +	2.098×10^{3} +	2.034×10^{3} =	2.074×10^{3} +	2.349×10^{3} +	2.054×10^{3} +	2.089×10^{3} +	2.047×10^{3}
	std	1.883×10^{1}	7.290	1.276×10^{1}	5.983	3.669	7.679	8.098	7.071×10^{1}	8.039×10^{1}	5.588	1.667×10^{1}	1.241×10^{2}	2.346×10^{1}	1.938×10^{1}	3.551×10^{1}
$f8$	mean	2.480×10^{3} =	2.231×10^{3} =	**2.228×10^{3}** =	2.257×10^{3} +	2.229×10^{3} =	2.238×10^{3} =	2.233×10^{3} =	3.551×10^{3} +	2.467×10^{3} +	2.230×10^{3} −	2.237×10^{3} −	3.613×10^{4} +	2.488×10^{3} +	2.266×10^{3} +	2.242×10^{3}
	std	4.946×10^{2}	9.577	2.445	2.009×10^{1}	3.240	1.629×10^{1}	1.036×10^{1}	1.190×10^{3}	2.686×10^{2}	1.795	5.260	1.502×10^{5}	2.040×10^{2}	2.614×10^{1}	7.826×10^{1}
$f9$	mean	2.519×10^{3} +	2.425×10^{3} =	2.574×10^{3} =	2.457×10^{3} =	**2.364×10^{3}** =	2.525×10^{3} +	2.474×10^{3} =	2.645×10^{3} +	2.653×10^{3} +	2.464×10^{3} =	2.635×10^{3} +	2.917×10^{3} +	2.578×10^{3} +	2.560×10^{3} +	2.431×10^{3}
	std	1.579×10^{2}	1.740×10^{2}	1.339×10^{2}	8.919×10^{1}	1.188×10^{2}	1.638×10^{2}	1.646×10^{2}	1.716×10^{2}	1.371×10^{2}	1.441×10^{2}	6.854×10^{1}	4.898×10^{1}	2.002×10^{2}	1.320×10^{2}	1.731×10^{2}
$f10$	mean	2.613×10^{3} =	2.632×10^{3} =	2.606×10^{3} =	2.618×10^{3} =	**2.600×10^{3}** −	2.603×10^{3} −	2.604×10^{3} −	2.747×10^{3} =	2.745×10^{3} =	2.618×10^{3} =	2.610×10^{3} =	2.664×10^{3} =	2.639×10^{3} =	2.610×10^{3} =	2.651×10^{3}
	std	8.904	6.096×10^{1}	8.392×10^{-1}	6.990	1.044	4.441	5.214	2.193×10^{2}	2.541×10^{2}	4.214×10^{1}	0.000	1.475×10^{1}	5.796×10^{1}	0.000	4.947×10^{1}
$f11$	mean	2.627×10^{3} −	2.602×10^{3} −	2.762×10^{3} +	2.614×10^{3} −	**2.600×10^{3}** −	2.603×10^{3} −	2.604×10^{3} −	2.702×10^{3} +	2.644×10^{3} +	2.620×10^{3} −	2.650×10^{3} +	2.758×10^{3} +	2.613×10^{3} −	2.618×10^{3} −	2.633×10^{3}
	std	4.028×10^{1}	5.702	3.161×10^{2}	4.202	8.142×10^{-2}	5.484	6.620	2.619×10^{2}	1.556×10^{2}	7.483×10^{1}	1.542×10^{2}	5.495×10^{1}	1.789×10^{1}	7.341	1.582×10^{2}
$f12$	mean	2.875×10^{3} +	2.876×10^{3} +	2.867×10^{3} =	2.888×10^{3} +	2.866×10^{3} −	**2.866×10^{3}** −	2.866×10^{3} −	2.918×10^{3} +	2.880×10^{3} +	2.868×10^{3} =	2.870×10^{3} +	2.974×10^{3} +	2.883×10^{3} +	2.870×10^{3} +	2.868×10^{3}
	std	6.093	1.689×10^{1}	6.181×10^{-1}	4.787	5.734×10^{-1}	7.909×10^{-1}	9.331×10^{-1}	5.932×10^{1}	1.617×10^{1}	2.319×10^{1}	0.000	2.993×10^{1}	1.906×10^{1}	0.000	1.551
+ /=/−		5/5/2	2/8/2	6/5/1	6/5/1	4/4/4	5/3/4	3/5/4	9/3/0	8/4/0	0/8/4	8/3/1	11/1/0	7/4/1	8/2/2	–
Avg. Rank		8.2	4.1	7.6	8.3	**3.7**	6.3	5.4	12.8	10.7	3.8	9.3	14.6	9.8	8.6	6.8

Table 18
CEC2022–20D.

FunNum.		DE	PSO	CMA-ES	CL-PSO	SaDE	JADE	L-SHADE	WOA	PPSO	MBGO	COA	FATA	HO	MSA	CIA
$f1$	mean	8.695×10^{3} +	3.211×10^{2} +	2.404×10^{3} +	4.296×10^{3} +	4.134×10^{2} +	3.121×10^{3} =	2.109×10^{3} +	6.299×10^{3} +	2.431×10^{3} +	3.004×10^{2} =	1.071×10^{4} +	3.256×10^{4} +	5.336×10^{3} +	6.039×10^{3} +	**3.000×10^{2}**
	std	4.632×10^{3}	1.081×10^{2}	4.815×10^{2}	7.646×10^{2}	4.040×10^{1}	4.010×10^{3}	2.875×10^{3}	4.090×10^{3}	1.718×10^{3}	4.899×10^{-1}	1.897×10^{3}	4.475×10^{3}	1.829×10^{3}	1.978×10^{3}	2.834×10^{-4}
$f2$	mean	5.659×10^{2} +	4.653×10^{2} +	5.099×10^{2} +	5.267×10^{2} +	**4.490×10^{2}** =	4.511×10^{2} =	4.529×10^{2} =	6.200×10^{2} +	5.554×10^{2} +	4.653×10^{2} +	5.406×10^{2} +	2.702×10^{3} +	5.905×10^{2} +	4.854×10^{2} +	4.528×10^{2}
	std	4.854×10^{1}	1.754×10^{1}	1.293×10^{1}	1.643×10^{1}	7.576×10^{-1}	4.500	6.440	6.772×10^{1}	4.663×10^{1}	2.703×10^{1}	2.446×10^{1}	4.448×10^{2}	6.945×10^{1}	1.891×10^{1}	1.331×10^{1}
$f3$	mean	6.001×10^{2} +	6.000×10^{2} +	6.000×10^{2} +	6.000×10^{2} +	6.000×10^{2} +	6.000×10^{2} +	6.000×10^{2} +	6.001×10^{2} +	6.000×10^{2} +	6.000×10^{2} +	6.000×10^{2} +	6.012×10^{2} +	6.000×10^{2} +	6.000×10^{2} +	**6.000×10^{2}**
	std	6.056×10^{-2}	1.718×10^{-10}	4.836×10^{-3}	1.473×10^{-4}	9.551×10^{-9}	5.183×10^{-5}	1.551×10^{-4}	5.067×10^{-2}	1.618×10^{-2}	0.000	0.000	2.065×10^{-1}	0.000	0.000	1.581×10^{-13}
$f4$	mean	8.010×10^{2} =	**8.009×10^{2}** =	8.040×10^{2} +	8.017×10^{2} +	8.031×10^{2} +	8.023×10^{2} +	8.022×10^{2} +	8.014×10^{2} +	8.012×10^{2} =	8.016×10^{2} +	8.035×10^{2} +	8.038×10^{2} +	8.013×10^{2} =	8.023×10^{2} +	8.010×10^{2}
	std	3.383×10^{-1}	5.754×10^{-1}	3.850×10^{-1}	2.377×10^{-1}	3.993×10^{-1}	4.447×10^{-1}	4.945×10^{-1}	4.908×10^{-1}	4.198×10^{-1}	5.588×10^{-1}	5.000×10^{-1}	3.671×10^{-1}	6.992×10^{-1}	5.735×10^{-1}	4.521×10^{-1}
$f5$	mean	9.017×10^{2} −	9.039×10^{2} =	9.031×10^{2} −	9.018×10^{2} −	**9.000×10^{2}** −	9.007×10^{2} −	9.006×10^{2} −	9.065×10^{2} =	9.037×10^{2} =	9.008×10^{2} −	9.038×10^{2} =	9.138×10^{2} +	9.022×10^{2} −	9.009×10^{2} −	9.059×10^{2}
	std	9.638×10^{-1}	2.001	5.039×10^{-1}	4.282×10^{-1}	1.872×10^{-4}	9.800×10^{-1}	1.026	3.354	1.781	6.368×10^{-1}	9.452×10^{-1}	2.801	8.718×10^{-1}	3.590×10^{-1}	4.073
$f6$	mean	3.208×10^{5} =	7.297×10^{4} +	5.331×10^{6} +	2.260×10^{6} +	8.552×10^{6} +	5.761×10^{6} =	1.070×10^{7} =	7.548×10^{4} =	1.858×10^{5} +	5.239×10^{4} =	2.123×10^{7} +	1.622×10^{9} +	5.810×10^{4} =	2.176×10^{7} +	**5.046×10^{4}**
	std	8.815×10^{5}	2.210×10^{4}	2.308×10^{6}	1.170×10^{6}	3.546×10^{6}	7.582×10^{6}	1.534×10^{7}	3.067×10^{4}	2.543×10^{5}	1.518×10^{4}	1.155×10^{7}	6.017×10^{8}	2.646×10^{4}	1.568×10^{7}	2.030×10^{4}
$f7$	mean	2.080×10^{3} =	2.132×10^{3} +	2.322×10^{3} +	2.143×10^{3} +	2.122×10^{3} +	2.110×10^{3} +	2.123×10^{3} +	3.338×10^{3} +	2.810×10^{3} +	2.054×10^{3} =	2.380×10^{3} +	4.394×10^{3} +	2.389×10^{3} +	2.238×10^{3} +	**2.052×10^{3}**
	std	6.254×10^{1}	1.035×10^{2}	5.504×10^{1}	3.051×10^{1}	3.373×10^{1}	6.928×10^{1}	8.275×10^{1}	6.484×10^{2}	4.201×10^{2}	1.451×10^{1}	1.085×10^{2}	6.095×10^{2}	2.274×10^{2}	1.045×10^{2}	2.298×10^{1}
$f8$	mean	4.205×10^{3} +	3.331×10^{3} +	2.434×10^{3} +	2.815×10^{3} +	4.454×10^{3} +	3.137×10^{3} =	3.225×10^{3} =	4.887×10^{3} +	5.656×10^{3} +	2.652×10^{3} +	5.221×10^{3} +	8.176×10^{8} +	4.977×10^{3} +	5.557×10^{3} +	**2.293×10^{3}**
	std	1.434×10^{3}	9.285×10^{2}	7.040×10^{1}	2.641×10^{2}	1.222×10^{3}	1.157×10^{3}	1.569×10^{3}	1.667×10^{3}	2.469×10^{3}	3.232×10^{2}	1.548×10^{3}	9.986×10^{8}	1.191×10^{3}	1.352×10^{3}	7.170×10^{1}
$f9$	mean	2.788×10^{3} +	2.705×10^{3} +	2.650×10^{3} +	2.738×10^{3} +	2.640×10^{3} +	2.648×10^{3} =	2.646×10^{3} =	2.853×10^{3} +	2.909×10^{3} +	**2.569×10^{3}** −	2.645×10^{3} =	4.320×10^{3} +	2.873×10^{3} +	2.684×10^{3} +	2.622×10^{3}
	std	6.470×10^{1}	9.737×10^{1}	4.017	1.240×10^{1}	1.722	1.419×10^{1}	1.207×10^{1}	1.161×10^{2}	3.108×10^{2}	2.363×10^{1}	5.000	3.220×10^{2}	1.157×10^{2}	1.958×10^{1}	8.618×10^{1}
$f10$	mean	3.132×10^{3} =	3.410×10^{3} =	3.298×10^{3} =	2.854×10^{3} −	2.801×10^{3} −	2.820×10^{3} −	2.938×10^{3} −	4.607×10^{3} =	4.303×10^{3} =	3.083×10^{3} −	3.574×10^{3} =	3.326×10^{3} =	3.733×10^{3} =	**2.783×10^{3}** −	3.796×10^{3}
	std	6.302×10^{2}	7.966×10^{2}	1.223×10^{3}	1.089×10^{2}	8.334×10^{1}	7.591×10^{1}	5.734×10^{2}	1.043×10^{3}	1.183×10^{3}	7.082×10^{2}	1.449×10^{3}	2.663×10^{2}	9.552×10^{2}	6.992	1.193×10^{3}
$f11$	mean	2.691×10^{3} +	2.602×10^{3} +	2.620×10^{3} +	2.621×10^{3} +	2.600×10^{3} +	2.603×10^{3} +	2.603×10^{3} +	3.045×10^{3} +	2.698×10^{3} +	2.731×10^{3} =	2.784×10^{3} +	4.995×10^{3} +	2.909×10^{3} +	2.671×10^{3} +	**2.600×10^{3}**
	std	7.235×10^{1}	4.901	1.987	6.318	4.548×10^{-2}	6.787	7.039	6.044×10^{2}	2.983×10^{2}	1.481×10^{2}	3.815×10^{2}	8.501×10^{2}	5.612×10^{2}	2.450×10^{2}	1.439×10^{-4}
$f12$	mean	2.990×10^{3} =	3.014×10^{3} =	2.958×10^{3} −	3.045×10^{3} +	2.940×10^{3} −	2.943×10^{3} −	2.942×10^{3} −	3.239×10^{3} +	3.078×10^{3} +	**2.900×10^{3}** −	2.945×10^{3} −	3.495×10^{3} +	3.101×10^{3} +	2.956×10^{3} −	3.007×10^{3}
	std	2.551×10^{1}	2.809×10^{1}	1.011×10^{1}	1.664×10^{1}	2.420	5.108	4.810	1.541×10^{2}	1.075×10^{2}	0.000	6.700	7.033×10^{1}	1.307×10^{2}	7.951	5.048×10^{1}
+ /=/−		6/5/1	8/4/0	9/1/2	10/0/2	8/1/3	4/5/3	5/4/3	9/3/0	9/3/0	4/4/4	8/3/1	11/1/0	8/3/1	9/0/3	–
Avg. Rank		8.42	6.42	8.25	7.92	4.92	5.75	5.83	11.92	10.50	4.33	9.83	14.42	9.67	8.00	**3.83**

Table 19
HPA-Cons.

Pro.	Level	Type		DE	PSO	CMA-ES	CL-PSO	SaDE	JADE	L-SHADE	WOA	PPSO	MBGO	COA	FATA	HO	MSA	CIA
HPA131	3	0		2.532×10^{1} +	2.474×10^{1} =	2.506×10^{1} +	2.608×10^{1} +	2.512×10^{1} +	2.500×10^{1} +	2.516×10^{1} +	2.550×10^{1} +	2.500×10^{1} +	2.524×10^{1} +	2.600×10^{1} +	2.746×10^{1} +	2.565×10^{1} +	2.617×10^{1} +	**2.456×10^{1}**
			std	3.593×10^{-1}	3.434×10^{-1}	5.385×10^{-1}	3.356×10^{-1}	3.414×10^{-1}	2.655×10^{-1}	3.563×10^{-1}	6.479×10^{-1}	5.903×10^{-1}	3.402×10^{-1}	6.487×10^{-1}	7.580×10^{-1}	5.110×10^{-1}	6.579×10^{-1}	4.163×10^{-1}
		1		2.521×10^{1} +	2.431×10^{1} +	2.491×10^{1} +	2.598×10^{1} +	2.470×10^{1} +	2.447×10^{1} −	2.465×10^{1} −	2.552×10^{1} +	2.492×10^{1} +	2.479×10^{1} +	2.628×10^{1} +	2.757×10^{1} +	2.543×10^{1} +	2.580×10^{1} +	**2.384×10^{1}**
			std	5.488×10^{-1}	4.219×10^{-1}	3.659×10^{-1}	5.238×10^{-1}	4.077×10^{-1}	4.153×10^{-1}	3.621×10^{-1}	8.619×10^{-1}	6.864×10^{-1}	4.531×10^{-1}	5.264×10^{-1}	1.235	5.662×10^{-1}	6.935×10^{-1}	5.217×10^{-1}
		2		7.279×10^{7} +	1.069×10^{8} +	**4.804×10^{6}** =	3.242×10^{8} +	1.770×10^{7} =	8.881×10^{6} =	1.382×10^{7} =	4.435×10^{7} =	1.691×10^{7} =	1.733×10^{8} +	1.308×10^{7} =	7.280×10^{8} +	1.006×10^{8} +	1.211×10^{7} =	2.550×10^{7}
			std	5.385×10^{7}	4.056×10^{7}	1.601×10^{7}	1.025×10^{8}	2.680×10^{7}	1.446×10^{7}	1.653×10^{7}	5.047×10^{7}	3.103×10^{7}	4.430×10^{7}	1.577×10^{7}	1.944×10^{8}	7.598×10^{7}	1.855×10^{7}	2.768×10^{7}
	4	0		2.375×10^{1} +	2.338×10^{1} +	2.413×10^{1} +	2.461×10^{1} +	2.380×10^{1} +	2.372×10^{1} +	2.367×10^{1} +	2.385×10^{1} +	2.359×10^{1} +	2.372×10^{1} −	2.481×10^{1} +	2.575×10^{1} +	2.410×10^{1} +	2.464×10^{1} +	**2.308×10^{1}**
			std	2.995×10^{-1}	2.683×10^{-1}	4.284×10^{-1}	4.288×10^{-1}	3.686×10^{-1}	2.042×10^{-1}	2.437×10^{-1}	4.940×10^{-1}	4.062×10^{-1}	2.409×10^{-1}	4.656×10^{-1}	5.369×10^{-1}	3.179×10^{-1}	4.774×10^{-1}	1.568×10^{-1}
		1		2.423×10^{1} +	2.387×10^{1} +	2.466×10^{1} +	2.526×10^{1} +	2.434×10^{1} +	2.417×10^{1} −	2.418×10^{1} +	2.453×10^{1} +	2.409×10^{1} −	2.378×10^{1} +	2.561×10^{1} +	2.683×10^{1} +	2.458×10^{1} +	2.544×10^{1} +	**2.322×10^{1}**
			std	3.054×10^{-1}	3.861×10^{-1}	5.169×10^{-1}	4.807×10^{-1}	4.146×10^{-1}	3.184×10^{-1}	3.889×10^{-1}	6.522×10^{-1}	5.347×10^{-1}	3.716×10^{-1}	6.538×10^{-1}	9.214×10^{-1}	6.325×10^{-1}	4.321×10^{-1}	2.955×10^{-1}
		2		5.509×10^{7} +	5.773×10^{7} +	**4.192×10^{6}** =	2.578×10^{8} +	2.144×10^{7} =	1.018×10^{7} =	2.715×10^{7} =	3.031×10^{7} =	2.314×10^{7} =	1.495×10^{8} +	2.917×10^{7} =	4.996×10^{8} +	8.188×10^{7} +	3.985×10^{7} =	2.562×10^{7}
			std	3.178×10^{7}	4.014×10^{7}	9.188×10^{6}	9.366×10^{7}	2.104×10^{7}	1.410×10^{7}	2.879×10^{7}	3.718×10^{7}	3.003×10^{7}	5.098×10^{7}	3.103×10^{7}	2.300×10^{8}	5.234×10^{7}	3.625×10^{7}	3.311×10^{7}
	5	0		2.433×10^{1} +	2.384×10^{1} +	2.450×10^{1} +	2.501×10^{1} +	2.439×10^{1} +	2.411×10^{1} +	2.419×10^{1} +	2.453×10^{1} +	2.400×10^{1} +	2.419×10^{1} +	2.558×10^{1} +	2.656×10^{1} +	2.468×10^{1} +	2.548×10^{1} +	**2.350×10^{1}**
			std	3.341×10^{-1}	3.489×10^{-1}	3.755×10^{-1}	3.649×10^{-1}	3.067×10^{-1}	2.290×10^{-1}	3.129×10^{-1}	4.508×10^{-1}	4.704×10^{-1}	2.909×10^{-1}	5.354×10^{-1}	7.904×10^{-1}	3.673×10^{-1}	4.601×10^{-1}	3.218×10^{-1}
		1		2.495×10^{1} +	2.440×10^{1} +	2.540×10^{1} +	2.598×10^{1} +	2.497×10^{1} +	2.482×10^{1} +	2.475×10^{1} +	2.547×10^{1} +	2.469×10^{1} +	2.446×10^{1} +	2.649×10^{1} +	2.734×10^{1} +	2.546×10^{1} +	2.631×10^{1} +	**2.373×10^{1}**
			std	4.996×10^{-1}	4.646×10^{-1}	4.176×10^{-1}	5.630×10^{-1}	3.612×10^{-1}	3.114×10^{-1}	3.438×10^{-1}	7.099×10^{-1}	5.752×10^{-1}	3.353×10^{-1}	4.411×10^{-1}	9.821×10^{-1}	5.806×10^{-1}	7.099×10^{-1}	3.434×10^{-1}
		2		1.101×10^{8} +	1.337×10^{8} +	**2.768×10^{7}** =	4.657×10^{8} +	7.157×10^{7} =	3.675×10^{7} =	6.747×10^{7} =	7.782×10^{7} =	2.822×10^{7} =	2.704×10^{8} +	5.522×10^{7} =	7.950×10^{8} +	1.663×10^{8} +	9.265×10^{7} =	5.712×10^{7}
			std	3.758×10^{7}	4.582×10^{7}	2.245×10^{7}	1.697×10^{8}	3.590×10^{7}	2.923×10^{7}	3.457×10^{7}	9.404×10^{7}	4.144×10^{7}	8.421×10^{7}	4.498×10^{7}	2.529×10^{8}	1.071×10^{8}	6.303×10^{7}	4.267×10^{7}
HPA142	3	0		2.365×10^{2} +	2.303×10^{2} +	2.421×10^{2} +	2.426×10^{2} +	2.371×10^{2} +	2.361×10^{2} −	2.355×10^{2} +	2.514×10^{2} +	2.309×10^{2} −	2.309×10^{2} +	2.510×10^{2} +	2.615×10^{2} +	2.380×10^{2} +	2.492×10^{2} +	**2.276×10^{2}**
			std	4.579	2.858	5.079	5.201	4.264	2.812	3.645	1.139×10^{1}	4.934	2.682	8.691	8.971	6.020	6.235	2.167
		1		2.401×10^{2} +	2.350×10^{2} +	2.490×10^{2} +	2.513×10^{2} +	2.428×10^{2} +	2.395×10^{2} −	2.402×10^{2} +	2.563×10^{2} +	2.381×10^{2} −	2.334×10^{2} =	2.604×10^{2} +	2.824×10^{2} +	2.415×10^{2} +	2.580×10^{2} +	**2.298×10^{2}**
			std	5.413	5.654	5.412	7.115	5.208	5.268	3.293	1.004×10^{1}	7.195	5.135	7.756	1.759×10^{1}	6.952	9.680	5.102
		2		1.955×10^{8} +	2.157×10^{8} +	**4.502×10^{7}** =	8.467×10^{8} +	7.509×10^{7} =	5.782×10^{7} =	8.906×10^{7} =	6.582×10^{7} =	9.602×10^{7} =	1.974×10^{8} +	8.429×10^{7} =	1.417×10^{9} +	2.029×10^{8} +	5.965×10^{7} =	7.678×10^{7}
			std	9.694×10^{7}	6.275×10^{7}	2.767×10^{7}	2.385×10^{8}	3.925×10^{7}	2.944×10^{7}	3.823×10^{7}	1.088×10^{8}	1.375×10^{8}	1.373×10^{8}	5.621×10^{7}	4.641×10^{8}	1.337×10^{8}	6.054×10^{7}	7.845×10^{7}
	4	0		2.314×10^{2} +	2.266×10^{2} +	2.392×10^{2} +	2.375×10^{2} +	2.335×10^{2} +	2.317×10^{2} +	2.310×10^{2} +	2.446×10^{2} +	2.267×10^{2} +	2.250×10^{2} +	2.448×10^{2} +	2.596×10^{2} +	2.329×10^{2} +	2.476×10^{2} +	**2.226×10^{2}**
			std	2.979	3.113	4.551	5.355	3.730	2.300	3.262	1.079×10^{1}	3.152	1.975	7.079	1.585×10^{1}	3.815	7.880	2.282
		1		2.413×10^{2} +	2.363×10^{2} =	2.517×10^{2} +	2.543×10^{2} +	2.474×10^{2} +	2.434×10^{2} +	2.421×10^{2} +	2.617×10^{2} +	2.391×10^{2} +	2.340×10^{2} =	2.622×10^{2} +	2.804×10^{2} +	2.462×10^{2} +	2.618×10^{2} +	**2.328×10^{2}**
			std	5.579	5.950	5.345	6.404	5.631	5.283	5.066	1.051×10^{1}	6.669	5.320	1.091×10^{1}	1.476×10^{1}	7.036	8.366	5.385
		2		3.561×10^{8} +	4.458×10^{8} +	1.183×10^{8} =	1.216×10^{9} +	2.088×10^{8} =	1.264×10^{8} =	2.144×10^{8} =	1.860×10^{8} =	**1.162×10^{8}** =	3.726×10^{8} +	1.630×10^{8} =	2.066×10^{9} +	2.789×10^{8} =	1.530×10^{8} =	1.734×10^{8}
			std	1.252×10^{8}	1.793×10^{8}	6.747×10^{7}	3.667×10^{8}	9.279×10^{7}	7.701×10^{7}	7.548×10^{7}	1.726×10^{8}	1.031×10^{8}	8.740×10^{7}	1.085×10^{8}	4.666×10^{8}	1.581×10^{8}	1.100×10^{8}	1.383×10^{8}
	5	0		2.329×10^{2} +	2.271×10^{2} +	2.398×10^{2} +	2.413×10^{2} +	2.332×10^{2} +	2.319×10^{2} +	2.310×10^{2} +	2.503×10^{2} +	2.283×10^{2} +	2.241×10^{2} =	2.509×10^{2} +	2.596×10^{2} +	2.321×10^{2} +	2.458×10^{2} +	**2.224×10^{2}**
			std	4.721	3.790	4.504	3.881	3.513	3.238	2.618	1.133×10^{1}	3.672	2.109	7.781	1.045×10^{1}	4.239	6.708	2.217
		1		2.442×10^{2} +	2.389×10^{2} +	2.550×10^{2} +	2.569×10^{2} +	2.494×10^{2} +	2.446×10^{2} +	2.445×10^{2} +	2.619×10^{2} +	2.416×10^{2} +	2.357×10^{2} +	2.648×10^{2} +	2.846×10^{2} +	2.446×10^{2} +	2.644×10^{2} +	**2.311×10^{2}**
			std	4.027	4.102	5.430	5.977	6.272	5.235	5.865	1.287×10^{1}	5.954	3.388	9.735	1.268×10^{1}	5.859	1.053×10^{1}	4.560
		2		5.074×10^{8} +	6.098×10^{8} +	**1.841×10^{8}** =	1.537×10^{9} +	3.500×10^{8} +	1.956×10^{8} =	2.870×10^{8} =	2.206×10^{8} =	2.183×10^{8} =	4.095×10^{8} +	3.086×10^{8} =	2.436×10^{9} +	3.538×10^{8} =	2.282×10^{8} =	2.410×10^{8}
			std	1.581×10^{8}	1.824×10^{8}	6.018×10^{7}	4.358×10^{8}	8.884×10^{7}	8.221×10^{7}	1.015×10^{8}	1.626×10^{8}	1.771×10^{8}	1.003×10^{8}	1.952×10^{8}	5.507×10^{8}	1.339×10^{8}	1.502×10^{8}	1.327×10^{8}
HPA143	3	0		-1.098×10^{1} +	**-1.123×10^{1}** =	-1.079×10^{1} +	-1.044×10^{1} +	-1.098×10^{1} +	-1.108×10^{1} =	-1.101×10^{1} +	-1.040×10^{1} −	-1.096×10^{1} =	-1.113×10^{1} =	-1.045×10^{1} +	-9.634 +	-1.092×10^{1} +	-1.048×10^{1} +	-1.119×10^{1}
			std	1.783×10^{-1}	2.153×10^{-1}	2.662×10^{-1}	3.091×10^{-1}	2.149×10^{-1}	2.151×10^{-1}	1.610×10^{-1}	5.184×10^{-1}	3.689×10^{-1}	1.569×10^{-1}	3.673×10^{-1}	4.464×10^{-1}	1.654×10^{-1}	3.674×10^{-1}	1.825×10^{-1}
		1		-1.082×10^{1} =	-1.103×10^{1} =	-1.074×10^{1} +	-1.028×10^{1} +	-1.084×10^{1} =	-1.098×10^{1} =	-1.089×10^{1} =	-1.064×10^{1} +	-1.078×10^{1} =	-1.091×10^{1} =	-1.036×10^{1} +	-9.324 +	-1.073×10^{1} +	-1.024×10^{1} +	**-1.106×10^{1}**
			std	2.421×10^{-1}	2.962×10^{-1}	2.286×10^{-1}	3.834×10^{-1}	2.207×10^{-1}	1.921×10^{-1}	1.793×10^{-1}	3.681×10^{-1}	3.804×10^{-1}	2.190×10^{-1}	4.045×10^{-1}	4.952×10^{-1}	2.873×10^{-1}	3.271×10^{-1}	2.865×10^{-1}
		2		2.323×10^{8} +	1.987×10^{8} +	**4.672×10^{7}** =	8.348×10^{8} +	8.594×10^{7} =	5.397×10^{7} =	8.864×10^{7} =	1.177×10^{8} =	1.068×10^{8} =	4.705×10^{8} +	1.069×10^{8} =	1.413×10^{9} +	3.258×10^{8} +	9.180×10^{7} =	9.338×10^{7}
			std	1.244×10^{8}	7.504×10^{7}	3.357×10^{7}	2.594×10^{8}	3.974×10^{7}	2.343×10^{7}	4.794×10^{7}	1.229×10^{8}	1.367×10^{8}	1.754×10^{8}	5.016×10^{7}	4.641×10^{8}	2.184×10^{8}	4.758×10^{7}	9.344×10^{7}
	4	0		-1.111×10^{1} +	**-1.139×10^{1}** =	-1.101×10^{1} +	-1.044×10^{1} +	-1.109×10^{1} +	-1.120×10^{1} =	-1.124×10^{1} =	-1.082×10^{1} +	-1.106×10^{1} =	-1.131×10^{1} =	-1.059×10^{1} +	-9.794 +	-1.127×10^{1} =	-1.072×10^{1} +	-1.131×10^{1}
			std	1.708×10^{-1}	2.223×10^{-1}	2.831×10^{-1}	2.645×10^{-1}	1.895×10^{-1}	1.914×10^{-1}	1.921×10^{-1}	5.209×10^{-1}	3.631×10^{-1}	1.879×10^{-1}	3.085×10^{-1}	3.269×10^{-1}	2.082×10^{-1}	3.810×10^{-1}	3.262×10^{-1}
		1		-1.071×10^{1} +	**-1.097×10^{1}** =	-1.067×10^{1} +	-1.014×10^{1} +	-1.077×10^{1} =	-1.082×10^{1} =	-1.074×10^{1} +	-1.049×10^{1} +	-1.059×10^{1} +	-1.081×10^{1} =	-1.017×10^{1} +	-9.101 +	-1.072×10^{1} =	-1.020×10^{1} +	-1.095×10^{1}
			std	2.608×10^{-1}	2.867×10^{-1}	2.402×10^{-1}	3.175×10^{-1}	2.426×10^{-1}	2.233×10^{-1}	2.524×10^{-1}	3.526×10^{-1}	3.401×10^{-1}	2.500×10^{-1}	3.507×10^{-1}	4.846×10^{-1}	2.786×10^{-1}	3.862×10^{-1}	1.723×10^{-1}
		2		3.561×10^{8} +	4.665×10^{8} +	**1.237×10^{8}** =	1.194×10^{9} +	1.773×10^{8} =	1.367×10^{8} =	2.004×10^{8} =	3.600×10^{8} =	1.468×10^{8} =	7.725×10^{8} +	1.816×10^{8} =	2.064×10^{9} +	5.146×10^{8} +	1.613×10^{8} =	1.807×10^{8}
			std	1.410×10^{8}	1.732×10^{8}	6.129×10^{7}	3.041×10^{8}	6.604×10^{7}	5.093×10^{7}	6.964×10^{7}	2.586×10^{8}	1.801×10^{8}	2.324×10^{8}	1.108×10^{8}	4.657×10^{8}	3.024×10^{8}	1.146×10^{8}	1.633×10^{8}
	5	0		-1.099×10^{1} +	-1.113×10^{1} =	-1.075×10^{1} +	-1.030×10^{1} +	-1.095×10^{1} +	-1.110×10^{1} =	-1.102×10^{1} +	-1.064×10^{1} +	-1.099×10^{1} =	**-1.130×10^{1}** =	-1.028×10^{1} +	-9.364 +	-1.118×10^{1} =	-1.054×10^{1} +	-1.123×10^{1}
			std	2.557×10^{-1}	2.669×10^{-1}	2.121×10^{-1}	3.290×10^{-1}	2.491×10^{-1}	1.923×10^{-1}	1.878×10^{-1}	3.835×10^{-1}	3.200×10^{-1}	1.798×10^{-1}	2.416×10^{-1}	4.772×10^{-1}	2.446×10^{-1}	4.527×10^{-1}	2.182×10^{-1}
		1		-1.066×10^{1} =	**-1.089×10^{1}** =	-1.054×10^{1} +	-9.967 +	-1.063×10^{1} +	-1.078×10^{1} =	-1.066×10^{1} +	-1.049×10^{1} +	-1.049×10^{1} +	-1.074×10^{1} =	-1.010×10^{1} +	-9.233 +	-1.068×10^{1} =	-1.011×10^{1} +	-1.088×10^{1}
			std	2.952×10^{-1}	2.346×10^{-1}	2.440×10^{-1}	3.692×10^{-1}	2.233×10^{-1}	2.238×10^{-1}	2.060×10^{-1}	4.758×10^{-1}	3.186×10^{-1}	2.904×10^{-1}	3.078×10^{-1}	6.040×10^{-1}	2.273×10^{-1}	3.518×10^{-1}	2.149×10^{-1}
		2		6.091×10^{8} +	6.407×10^{8} +	**1.830×10^{8}** =	1.557×10^{9} +	3.157×10^{8} =	2.096×10^{8} =	2.953×10^{8} =	4.172×10^{8} =	3.115×10^{8} =	1.103×10^{9} +	3.047×10^{8} =	2.431×10^{9} +	7.070×10^{8} +	2.791×10^{8} =	2.361×10^{8}
			std	1.646×10^{8}	1.754×10^{8}	8.911×10^{7}	2.840×10^{8}	1.098×10^{8}	7.393×10^{7}	1.103×10^{8}	3.743×10^{8}	2.284×10^{8}	2.637×10^{8}	1.761×10^{8}	5.508×10^{8}	3.557×10^{8}	1.656×10^{8}	1.425×10^{8}
+ /=/−				25/2/0	19/8/0	18/9/0	27/0/0	17/10/0	8/15/4	15/11/1	17/9/1	11/13/3	17/9/1	18/9/0	27/0/0	21/6/0	18/9/0	–
Avg. Rank				7.1	4.4	5.7	11.8	6.2	3.1	4.8	9.0	4.3	5.4	10.0	14.0	8.3	9.1	**1.7**

Table 20
HPA-Uncons.

Pro.	Level	Type		DE		PSO		CMA-ES		CL-PSO		SaDE		JADE		L-SHADE		WOA		PPSO		MBGO		COA		FATA		HO		MSA		CIA
HPA131	4	0	mean	2.486×10^{1}	+	2.454×10^{1}	+	2.467×10^{1}	+	2.563×10^{1}	+	2.472×10^{1}	+	2.459×10^{1}	+	2.463×10^{1}	+	2.523×10^{1}	+	2.480×10^{1}	+	2.500×10^{1}	+	2.514×10^{1}	+	2.685×10^{1}	+	2.529×10^{1}	+	2.550×10^{1}	+	**2.419×10^{1}**
			std	3.808×10^{-1}		3.195×10^{-1}		3.279×10^{-1}		4.199×10^{-1}		2.865×10^{-1}		2.839×10^{-1}		2.641×10^{-1}		4.716×10^{-1}		4.492×10^{-1}		2.394×10^{-1}		4.184×10^{-1}		4.989×10^{-1}		3.968×10^{-1}		5.280×10^{-1}		2.025×10^{-1}
		1	mean	2.438×10^{1}	+	2.411×10^{1}	+	2.439×10^{1}	+	2.560×10^{1}	+	2.428×10^{1}	+	2.397×10^{1}	+	2.418×10^{1}	+	2.503×10^{1}	+	2.459×10^{1}	+	2.447×10^{1}	+	2.482×10^{1}	+	2.689×10^{1}	+	2.505×10^{1}	+	2.531×10^{1}	+	**2.346×10^{1}**
			std	4.291×10^{-1}		4.382×10^{-1}		3.363×10^{-1}		5.611×10^{-1}		3.202×10^{-1}		3.139×10^{-1}		3.809×10^{-1}		6.412×10^{-1}		4.890×10^{-1}		4.297×10^{-1}		4.332×10^{-1}		7.273×10^{-1}		5.420×10^{-1}		6.427×10^{-1}		3.262×10^{-1}
		2	mean	3.540×10^{1}	=	4.047×10^{1}	+	**3.021×10^{1}**	−	5.567×10^{1}	+	3.148×10^{1}	−	3.031×10^{1}	−	3.154×10^{1}	−	3.513×10^{1}	=	3.280×10^{1}	=	4.663×10^{1}	+	3.034×10^{1}	−	8.195×10^{1}	+	3.929×10^{1}	=	3.343×10^{1}	=	3.519×10^{1}
			std	4.799		2.573		1.713		7.063		1.384		1.366		1.838		7.219		4.681		5.684		1.517		1.299×10^{1}		5.889		3.847		3.420
		0	mean	2.346×10^{1}	+	2.318×10^{1}	+	2.374×10^{1}	+	2.424×10^{1}	+	2.348×10^{1}	+	2.333×10^{1}	+	2.345×10^{1}	+	2.374×10^{1}	+	2.340×10^{1}	+	2.359×10^{1}	+	2.409×10^{1}	+	2.531×10^{1}	+	2.389×10^{1}	+	2.437×10^{1}	+	**2.282×10^{1}**
			std	2.693×10^{-1}		2.059×10^{-1}		2.872×10^{-1}		3.432×10^{-1}		2.458×10^{-1}		1.614×10^{-1}		1.996×10^{-1}		3.689×10^{-1}		3.628×10^{-1}		2.489×10^{-1}		3.241×10^{-1}		5.317×10^{-1}		3.072×10^{-1}		4.108×10^{-1}		1.080×10^{-1}
		1	mean	2.378×10^{1}	+	2.359×10^{1}	+	2.431×10^{1}	+	2.481×10^{1}	+	2.410×10^{1}	+	2.363×10^{1}	+	2.371×10^{1}	+	2.420×10^{1}	+	2.371×10^{1}	+	2.353×10^{1}	+	2.497×10^{1}	+	2.628×10^{1}	+	2.417×10^{1}	+	2.509×10^{1}	+	**2.285×10^{1}**
			std	3.694×10^{-1}		3.506×10^{-1}		4.475×10^{-1}		4.308×10^{-1}		3.530×10^{-1}		2.906×10^{-1}		3.046×10^{-1}		4.504×10^{-1}		4.136×10^{-1}		3.121×10^{-1}		4.324×10^{-1}		6.521×10^{-1}		4.785×10^{-1}		5.482×10^{-1}		2.679×10^{-1}
		2	mean	3.515×10^{1}	+	3.427×10^{1}	=	**3.036×10^{1}**	=	4.974×10^{1}	+	3.316×10^{1}	=	3.202×10^{1}	=	3.305×10^{1}	=	3.493×10^{1}	=	3.535×10^{1}	=	4.167×10^{1}	+	3.051×10^{1}	=	6.638×10^{1}	+	3.869×10^{1}	+	3.369×10^{1}	=	3.191×10^{1}
			std	2.710		2.552		1.670		6.105		2.815		2.003		2.386		4.135		4.866		3.714		1.405		1.409×10^{1}		6.792		4.386		2.652
		0	mean	2.393×10^{1}	+	2.361×10^{1}	+	2.396×10^{1}	+	2.475×10^{1}	+	2.411×10^{1}	+	2.378×10^{1}	+	2.387×10^{1}	+	2.430×10^{1}	+	2.385×10^{1}	+	2.402×10^{1}	+	2.457×10^{1}	+	2.590×10^{1}	+	2.446×10^{1}	+	2.491×10^{1}	+	**2.307×10^{1}**
			std	3.164×10^{-1}		3.319×10^{-1}		2.690×10^{-1}		4.383×10^{-1}		2.972×10^{-1}		2.063×10^{-1}		1.806×10^{-1}		4.727×10^{-1}		2.997×10^{-1}		2.353×10^{-1}		3.709×10^{-1}		5.209×10^{-1}		2.894×10^{-1}		5.329×10^{-1}		1.950×10^{-1}
		1	mean	2.453×10^{1}	+	2.429×10^{1}	+	2.489×10^{1}	+	2.558×10^{1}	+	2.467×10^{1}	+	2.431×10^{1}	+	2.422×10^{1}	+	2.496×10^{1}	+	2.430×10^{1}	+	2.416×10^{1}	+	2.537×10^{1}	+	2.684×10^{1}	+	2.490×10^{1}	+	2.580×10^{1}	+	**2.331×10^{1}**
			std	4.240×10^{-1}		3.694×10^{-1}		3.427×10^{-1}		4.585×10^{-1}		3.573×10^{-1}		3.302×10^{-1}		3.209×10^{-1}		5.821×10^{-1}		4.703×10^{-1}		3.460×10^{-1}		5.514×10^{-1}		5.575×10^{-1}		4.642×10^{-1}		3.966×10^{-1}		3.155×10^{-1}
		2	mean	4.202×10^{1}	=	4.053×10^{1}	=	**3.217×10^{1}**	−	6.280×10^{1}	+	3.747×10^{1}	=	3.591×10^{1}	=	3.909×10^{1}	=	4.063×10^{1}	=	3.716×10^{1}	=	5.084×10^{1}	+	3.351×10^{1}	=	8.648×10^{1}	+	4.606×10^{1}	+	4.050×10^{1}	=	3.685×10^{1}
			std	5.951		3.611		2.417		7.425		3.655		2.797		3.329		9.053		6.113		5.658		3.250		1.657×10^{1}		7.940		6.399		4.872
		0	mean	2.276×10^{2}	+	2.259×10^{2}	+	2.304×10^{2}	+	2.347×10^{2}	+	2.294×10^{2}	+	2.278×10^{2}	+	2.272×10^{2}	+	2.375×10^{2}	+	2.266×10^{2}	+	2.276×10^{2}	+	2.341×10^{2}	+	2.466×10^{2}	+	2.304×10^{2}	+	2.368×10^{2}	+	**2.229×10^{2}**
			std	2.245		1.356		2.096		4.095		1.949		1.635		1.574		6.397		2.984		1.627		2.928		7.426		2.777		4.201		9.424×10^{-1}
		1	mean	2.302×10^{2}	+	2.270×10^{2}	+	2.339×10^{2}	+	2.435×10^{2}	+	2.317×10^{2}	+	2.295×10^{2}	+	2.292×10^{2}	+	2.385×10^{2}	+	2.308×10^{2}	+	2.267×10^{2}	+	2.399×10^{2}	+	2.581×10^{2}	+	2.329×10^{2}	+	2.411×10^{2}	+	**2.222×10^{2}**
			std	3.084		3.113		2.117		5.007		2.281		2.401		2.486		7.087		5.094		1.806		4.101		9.475		4.912		5.281		4.890
		2	mean	4.719×10^{2}	+	4.633×10^{2}	+	3.378×10^{2}	=	9.321×10^{2}	+	3.838×10^{2}	=	3.467×10^{2}	=	3.818×10^{2}	=	4.097×10^{2}	=	3.986×10^{2}	=	5.391×10^{2}	+	**3.322×10^{2}**	=	1.338×10^{3}	+	5.744×10^{2}	+	3.895×10^{2}	=	3.864×10^{2}
			std	8.889		4.637		3.073		1.789		4.344		3.365		4.535		8.035		1.117		1.034		3.256		3.510		1.365		5.178		8.083	
		0	mean	2.235×10^{2}	+	2.211×10^{2}	+	2.271×10^{2}	+	2.312×10^{2}	+	2.250×10^{2}	+	2.229×10^{2}	+	2.235×10^{2}	+	2.339×10^{2}	+	2.225×10^{2}	+	2.221×10^{2}	+	2.304×10^{2}	+	2.414×10^{2}	+	2.250×10^{2}	+	2.325×10^{2}	+	**2.179×10^{2}**
			std	2.796		2.001		1.913		3.858		2.302		1.438		1.637		4.709		2.564		1.945		2.992		6.374		3.052		3.690		9.583×10^{-1}
		1	mean	2.323×10^{2}	+	2.301×10^{2}	+	2.357×10^{2}	+	2.430×10^{2}	+	2.342×10^{2}	+	2.313×10^{2}	+	2.312×10^{2}	+	2.451×10^{2}	+	2.301×10^{2}	+	2.281×10^{2}	+	2.430×10^{2}	+	2.643×10^{2}	+	2.369×10^{2}	+	2.432×10^{2}	+	**2.223×10^{2}**
			std	3.268		2.542		4.154		5.340		3.164		2.397		3.792		8.356		5.378		3.203		4.935		1.169×10^{1}		4.887		5.325		4.194
		2	mean	6.279×10^{2}	+	6.357×10^{2}	+	4.148×10^{2}	=	1.201×10^{3}	+	4.876×10^{2}	+	4.289×10^{2}	=	4.949×10^{2}	=	5.339×10^{2}	=	5.166×10^{2}	=	7.927×10^{2}	+	**3.728×10^{2}**	−	1.752×10^{3}	+	6.553×10^{2}	+	4.592×10^{2}	=	4.490×10^{2}
			std	1.446×10^{2}		8.612×10^{1}		5.211×10^{1}		2.990×10^{2}		6.374×10^{1}		4.744×10^{1}		6.411×10^{1}		1.906×10^{2}		1.929×10^{2}		1.935×10^{2}		5.116×10^{1}		3.019×10^{2}		1.905×10^{2}		9.400×10^{1}		7.449×10^{1}
		0	mean	2.232×10^{2}	+	2.210×10^{2}	+	2.275×10^{2}	+	2.298×10^{2}	+	2.252×10^{2}	=	2.239×10^{2}	+	2.234×10^{2}	+	2.341×10^{2}	+	2.219×10^{2}	+	2.213×10^{2}	+	2.323×10^{2}	+	2.452×10^{2}	+	2.268×10^{2}	+	2.332×10^{2}	+	**2.173×10^{2}**
			std	1.811		1.785		2.697		2.748		2.464		1.858		1.844		6.288		3.032		1.843		3.849		6.439		3.196		4.120		1.264
		1	mean	2.330×10^{2}	+	2.308×10^{2}	+	2.401×10^{2}	+	2.454×10^{2}	+	2.359×10^{2}	+	2.327×10^{2}	+	2.335×10^{2}	+	2.409×10^{2}	+	2.351×10^{2}	+	2.297×10^{2}	+	2.466×10^{2}	+	2.682×10^{2}	+	2.384×10^{2}	+	2.480×10^{2}	+	**2.222×10^{2}**
			std	3.298		3.926		3.447		5.375		3.550		2.149		3.865		5.861		5.174		3.991		5.561		7.308		5.592		7.154		3.277
		2	mean	7.514×10^{2}	+	7.887×10^{2}	+	5.000×10^{2}	=	1.383×10^{3}	+	5.751×10^{2}	=	5.187×10^{2}	=	5.836×10^{2}	=	7.544×10^{2}	=	5.662×10^{2}	=	1.001×10^{3}	+	**4.240×10^{2}**	−	1.976×10^{3}	+	8.686×10^{2}	+	5.966×10^{2}	=	5.829×10^{2}
			std	1.438×10^{2}		1.832×10^{2}		5.999×10^{1}		1.815×10^{2}		8.271×10^{1}		7.014×10^{1}		8.943×10^{1}		2.729×10^{2}		2.534×10^{2}		1.631×10^{2}		7.286×10^{1}		3.077×10^{2}		2.384×10^{2}		1.190×10^{2}		1.383×10^{2}
		0	mean	-1.124×10^{1}	+	-1.131×10^{1}	=	-1.100×10^{1}	+	-1.059×10^{1}	+	-1.115×10^{1}	+	-1.126×10^{1}	+	-1.122×10^{1}	+	-1.053×10^{1}	+	-1.095×10^{1}	+	-1.122×10^{1}	+	-1.095×10^{1}	+	-9.839	+	-1.114×10^{1}	+	-1.070×10^{1}	+	**-1.142×10^{1}**
			std	2.014×10^{-1}		1.881×10^{-1}		2.038×10^{-1}		3.131×10^{-1}		1.599×10^{-1}		1.726×10^{-1}		1.594×10^{-1}		3.548×10^{-1}		3.343×10^{-1}		1.319×10^{-1}		2.213×10^{-1}		4.692×10^{-1}		1.708×10^{-1}		3.050×10^{-1}		1.384×10^{-1}
		1	mean	-1.108×10^{1}	+	**-1.134×10^{1}**	=	-1.094×10^{1}	+	-1.033×10^{1}	+	-1.100×10^{1}	+	-1.111×10^{1}	=	-1.111×10^{1}	=	-1.071×10^{1}	+	-1.098×10^{1}	=	-1.110×10^{1}	=	-1.083×10^{1}	+	-9.556	+	-1.098×10^{1}	+	-1.058×10^{1}	+	-1.131×10^{1}
			std	1.701×10^{-1}		1.889×10^{-1}		2.470×10^{-1}		2.971×10^{-1}		2.296×10^{-1}		1.666×10^{-1}		1.890×10^{-1}		3.498×10^{-1}		4.307×10^{-1}		1.581×10^{-1}		2.283×10^{-1}		4.228×10^{-1}		2.336×10^{-1}		3.104×10^{-1}		2.379×10^{-1}
		2	mean	8.574	+	1.255×10^{1}	+	−3.439	=	5.322×10^{1}	+	-8.505×10^{-1}	=	−4.443	−	-4.140×10^{-1}	=	8.297×10^{-2}	=	-9.527×10^{-1}	=	8.922	+	**−4.757**	−	9.845×10^{1}	+	1.236×10^{1}	+	−1.721	=	5.249×10^{-1}
			std	7.293		9.601		3.961		1.963×10^{1}		4.070		1.841		3.657		1.260×10^{1}		1.120×10^{1}		1.107×10^{1}		4.245		3.516×10^{1}		1.458×10^{1}		4.969		6.444
HPA143	4	0	mean	-1.128×10^{1}	+	-1.146×10^{1}	=	-1.112×10^{1}	+	-1.063×10^{1}	+	-1.124×10^{1}	+	-1.137×10^{1}	+	-1.131×10^{1}	+	-1.091×10^{1}	+	-1.118×10^{1}	+	-1.147×10^{1}	=	-1.099×10^{1}	+	-9.855	+	-1.144×10^{1}	=	-1.084×10^{1}	+	**-1.161×10^{1}**
			std	2.173		2.118		2.442		3.422		2.050		1.548		1.571		4.051		3.107		1.931		2.367		2.817		1.869		3.212		1.988
		1	mean	-1.092×10^{1}	+	-1.107×10^{1}	=	-1.078×10^{1}	+	-1.027×10^{1}	+	-1.093×10^{1}	+	-1.105×10^{1}	=	-1.103×10^{1}	=	-1.059×10^{1}	+	-1.076×10^{1}	=	-1.100×10^{1}	+	-1.068×10^{1}	+	-9.309	+	-1.097×10^{1}	+	-1.044×10^{1}	+	**-1.119×10^{1}**
			std	2.674×10^{-1}		2.633×10^{-1}		2.273×10^{-1}		3.131×10^{-1}		2.406×10^{-1}		1.675×10^{-1}		1.194×10^{-1}		4.197×10^{-1}		3.817×10^{-1}		1.560×10^{-1}		2.208×10^{-1}		3.401×10^{-1}		1.993×10^{-1}		2.672×10^{-1}		2.462×10^{-1}
		2	mean	2.541×10^{1}	+	2.748×10^{1}	+	4.874	=	8.874×10^{1}	+	9.782	=	4.821	=	1.162×10^{1}	=	3.900	=	9.275	+	3.289×10^{1}	+	**−2.515**	−	1.406×10^{2}	+	2.369×10^{1}	+	8.859	=	7.696
			std	1.138×10^{1}		1.155×10^{1}		6.356		2.441×10^{1}		7.764		4.521		6.864		1.437×10^{1}		1.529×10^{1}		1.706×10^{1}		4.131		3.116×10^{1}		1.787×10^{1}		9.893		8.905
		0	mean	-1.118×10^{1}	+	-1.135×10^{1}	=	-1.096×10^{1}	+	-1.042×10^{1}	+	-1.112×10^{1}	+	-1.128×10^{1}	+	-1.125×10^{1}	+	-1.067×10^{1}	+	-1.112×10^{1}	+	-1.141×10^{1}	=	-1.074×10^{1}	+	-9.535	+	-1.134×10^{1}	=	-1.073×10^{1}	+	**-1.147×10^{1}**
			std	2.342×10^{-1}		2.186×10^{-1}		2.196×10^{-1}		3.560×10^{-1}		2.075×10^{-1}		1.530×10^{-1}		1.752×10^{-1}		3.375×10^{-1}		2.636×10^{-1}		1.446×10^{-1}		2.667×10^{-1}		4.492×10^{-1}		1.484×10^{-1}		3.073×10^{-1}		1.895×10^{-1}
		1	mean	-1.085×10^{1}	+	-1.100×10^{1}	=	-1.070×10^{1}	+	-1.012×10^{1}	+	-1.085×10^{1}	+	-1.094×10^{1}	=	-1.088×10^{1}	+	-1.049×10^{1}	+	-1.080×10^{1}	=	-1.094×10^{1}	=	-1.056×10^{1}	+	-9.446	+	-1.089×10^{1}	+	-1.026×10^{1}	+	**-1.112×10^{1}**
			std	2.026×10^{-1}		2.592×10^{-1}		2.387×10^{-1}		3.195×10^{-1}		2.307×10^{-1}		1.551×10^{-1}		1.448×10^{-1}		3.658×10^{-1}		4.756×10^{-1}		1.606×10^{-1}		2.702×10^{-1}		4.965×10^{-1}		1.568×10^{-1}		2.765×10^{-1}		2.541×10^{-1}
		2	mean	4.080×10^{1}	+	4.601×10^{1}	+	8.907	=	1.111×10^{2}	+	1.964×10^{1}	=	1.237×10^{1}	=	2.134×10^{1}	=	2.655×10^{1}	=	1.399×10^{1}	=	5.111×10^{1}	+	**5.702**	−	1.625×10^{2}	+	3.905×10^{1}	+	1.373×10^{1}	=	1.759×10^{1}
			std	1.574×10^{1}		1.369×10^{1}		5.813		2.631×10^{1}		7.289		8.426		6.739		2.500×10^{1}		2.216×10^{1}		1.700×10^{1}		7.486		3.210×10^{1}		2.148×10^{1}		1.029×10^{1}		1.260×10^{1}
+ /=/−				25/2/0		19/8/0		18/7/2		27/0/0		18/8/1		15/10/2		16/8/3		18/9/0		16/11/0		23/4/0		18/3/6		27/0/0		24/3/0		18/9/0		–	
Avg. Rank				6.7		4.3		5.7		12.3		6.1		2.9		4.5		9.7		5.5		6.1		7.1		14.0		8.6		9.9		**1.6**	

large-scale and multimodal optimization problems represents another promising avenue for further research exploration.

7. Conclusion

We propose the Crested Ibis Algorithm, a novel meta-heuristic inspired by the crested ibis' behavior. The CIA employs two distinct populations and achieves a balance between exploration and exploitation through information exchange between these populations and the integration of adaptive factors. Extensive evaluations of benchmark suites and real-world engineering problems demonstrate the CIA's exceptional performance and robustness. Future research will focus on enhancing the CIA's adaptability, particularly for multimodal optimization problems.

CRediT authorship contribution statement

Yuefeng Xu: Writing – review & editing, Writing – original draft, Methodology, Investigation, Funding acquisition, Conceptualization. **Rui Zhong:** Writing – review & editing, Methodology, Investigation, Formal analysis. **Chao Zhang:** Writing – review & editing, Project administration. **Jun Yu:** Writing – review & editing, Methodology, Investigation, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by a JST SPRING grant (No. JPMJSP2119).

Appendix A. Summary of CEC benchmark suites

See Tables 10–12.

Appendix B. The result on CEC2017

See Tables 13–16 and Figs. 13–32.

Appendix C. The result on CEC2022

See Tables 17 and 18 and Figs. 33–36.

Appendix D. HPA

See Tables 19 and 20.

Data availability

I have shared the data link in my manuscript.

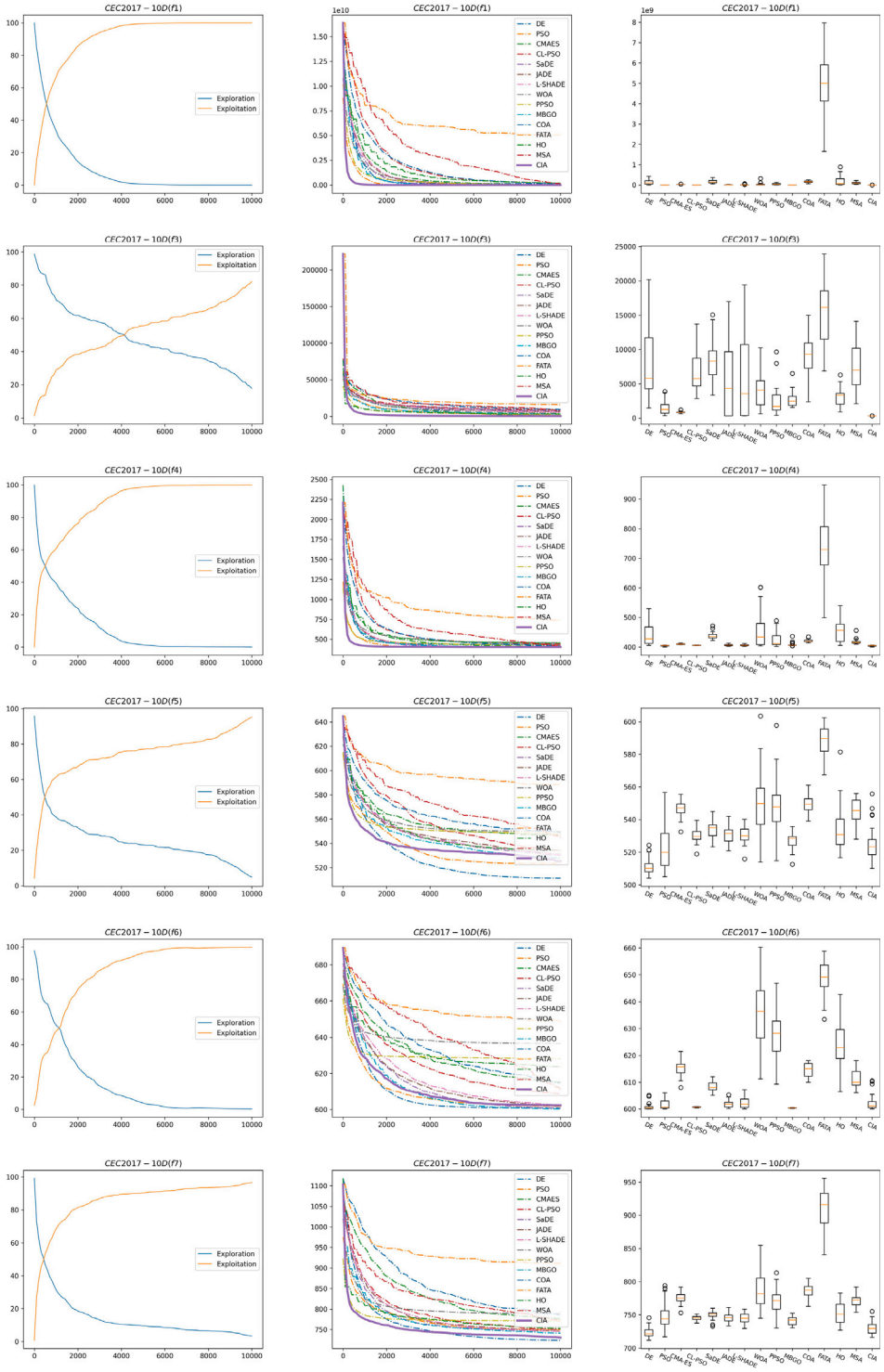

Fig. 13. The results of CEC2017-10D(f_1-f_7).

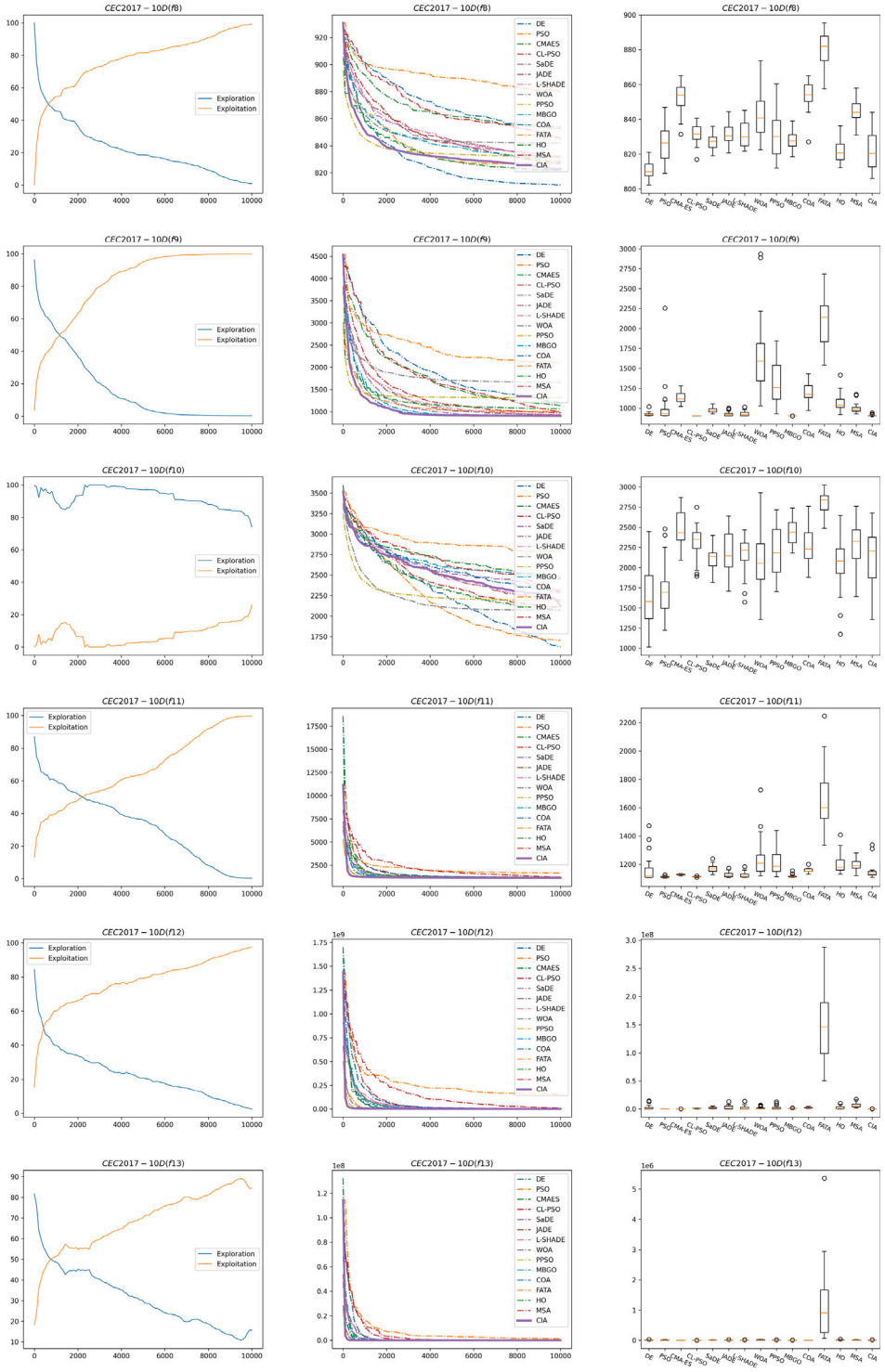

Fig. 14. The results of CEC2017-10D(f_8-f_{13}).

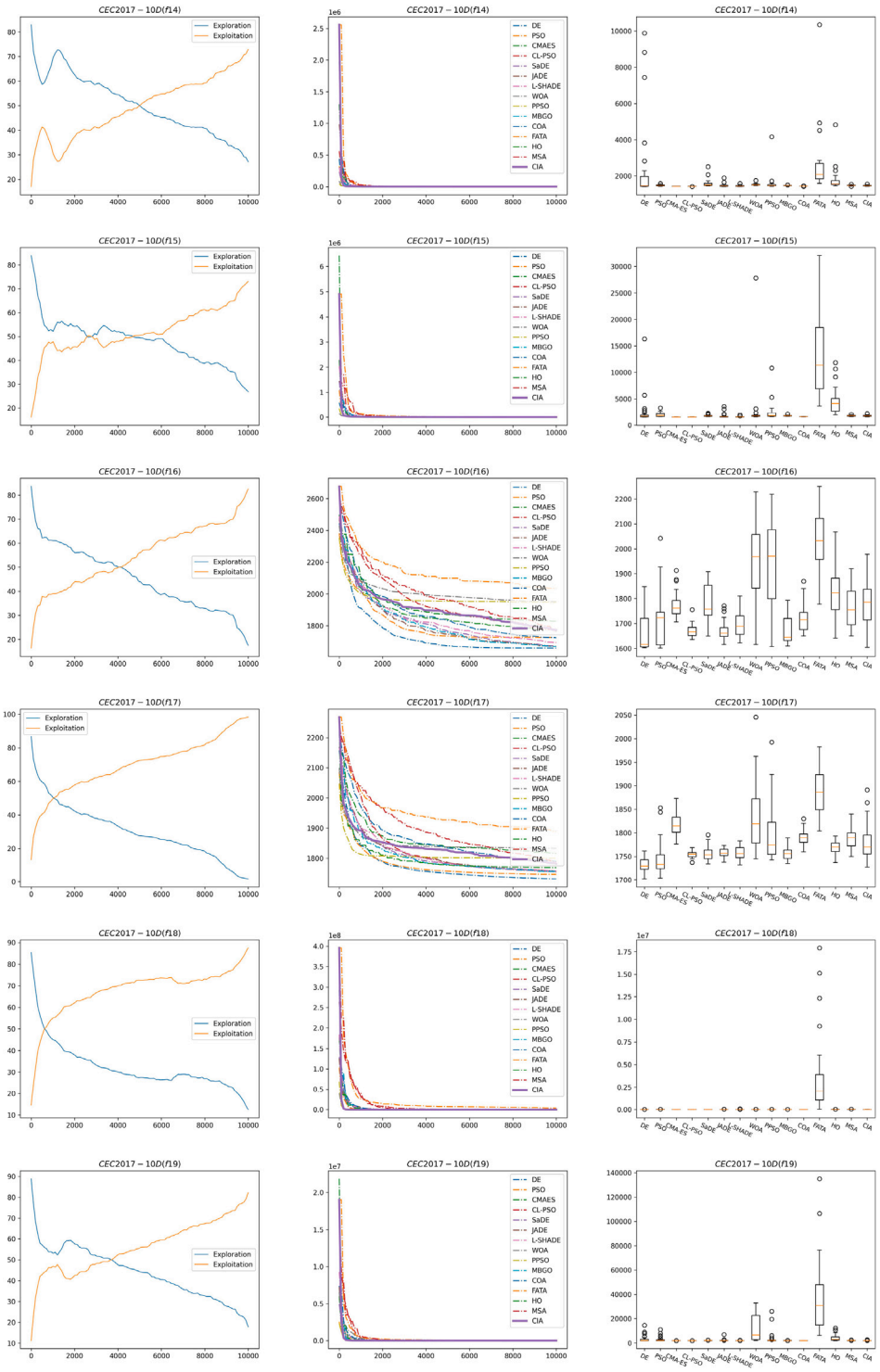

Fig. 15. The results of CEC2017-10D(f_{14}-f_{19}).

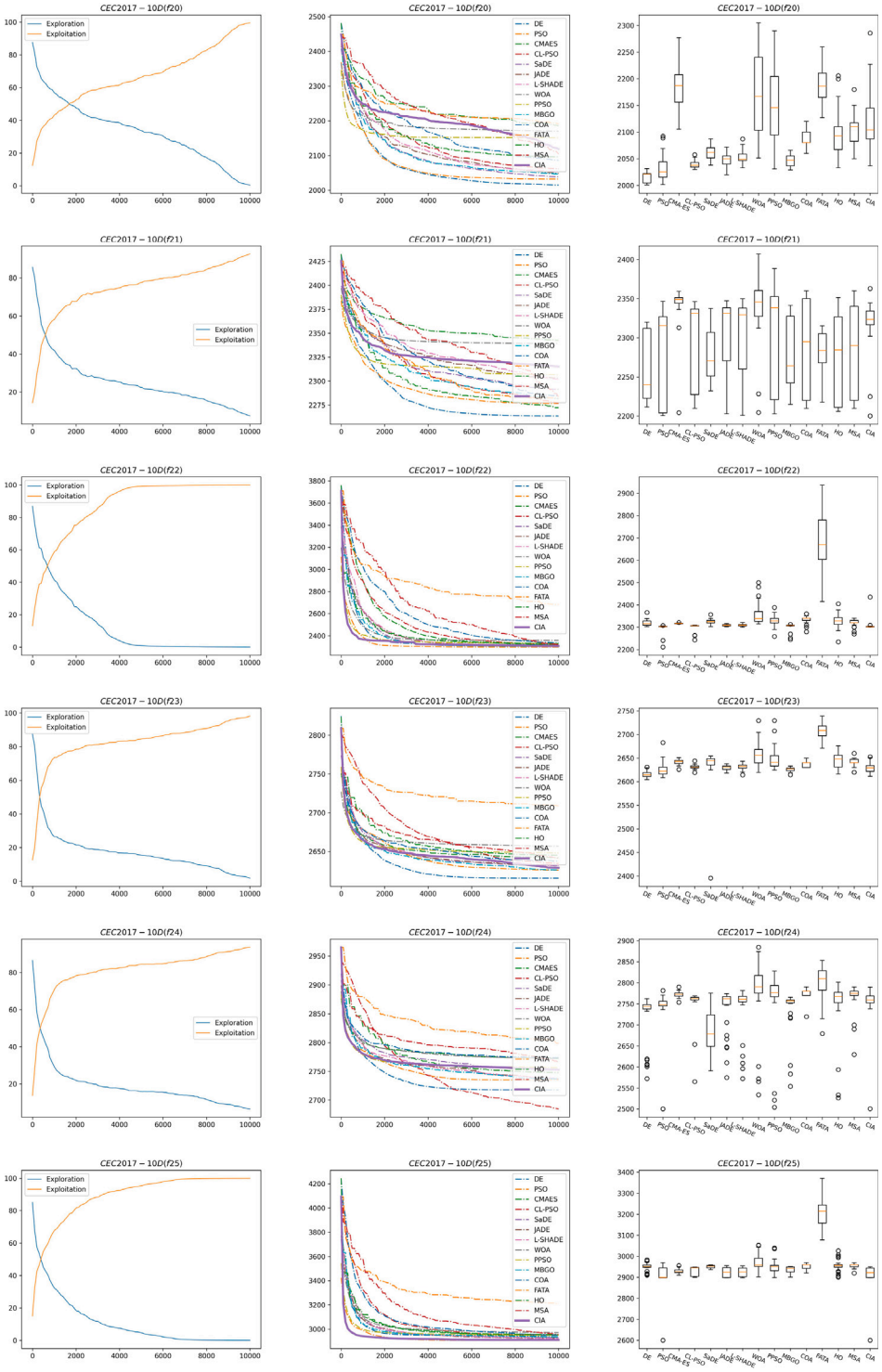

Fig. 16. The results of CEC2017-10D(f_{20}-f_{25}).

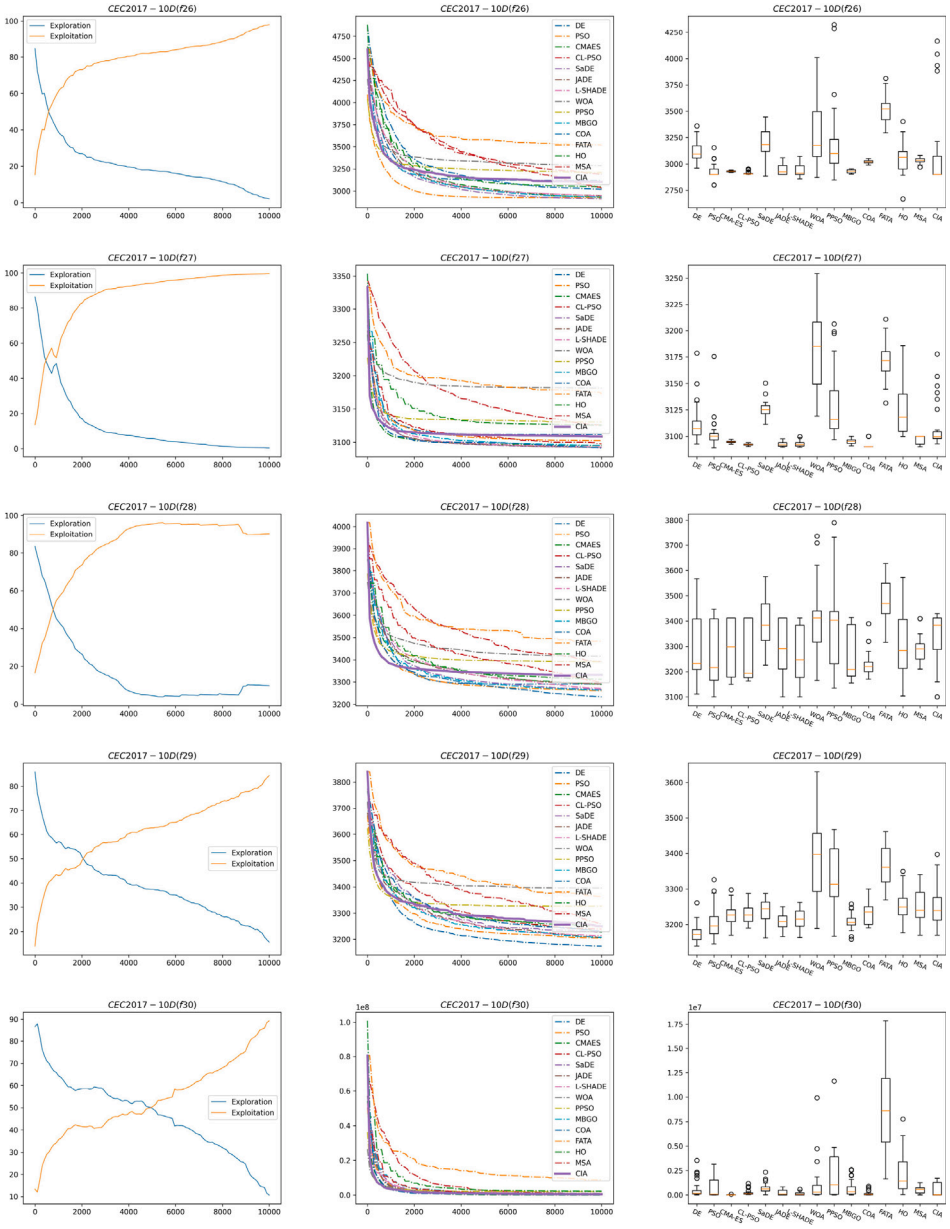

Fig. 17. The results of CEC2017-10D(f_{26}-f_{30}).

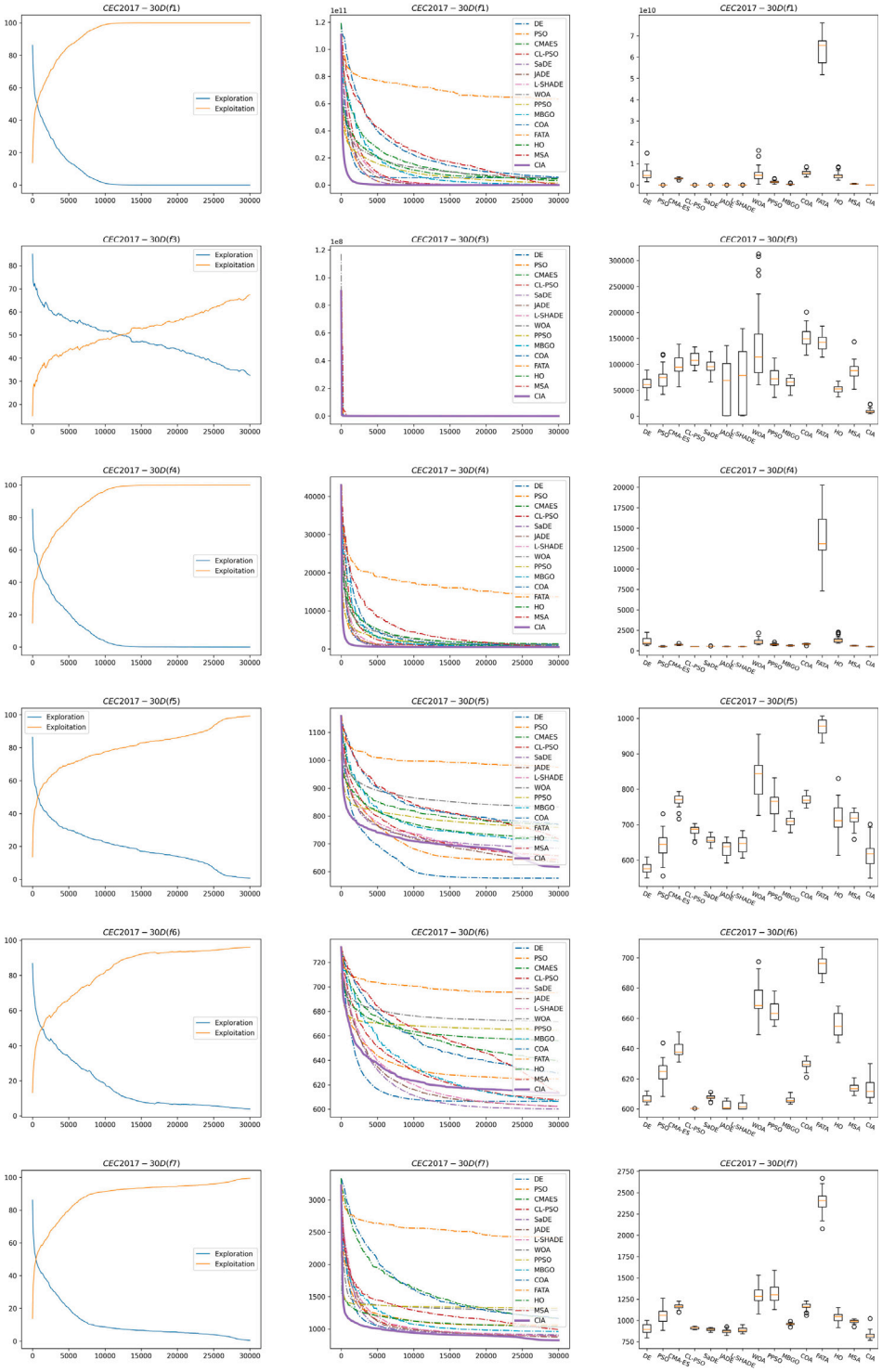

Fig. 18. The results of CEC2017-30D(f_1-f_7).

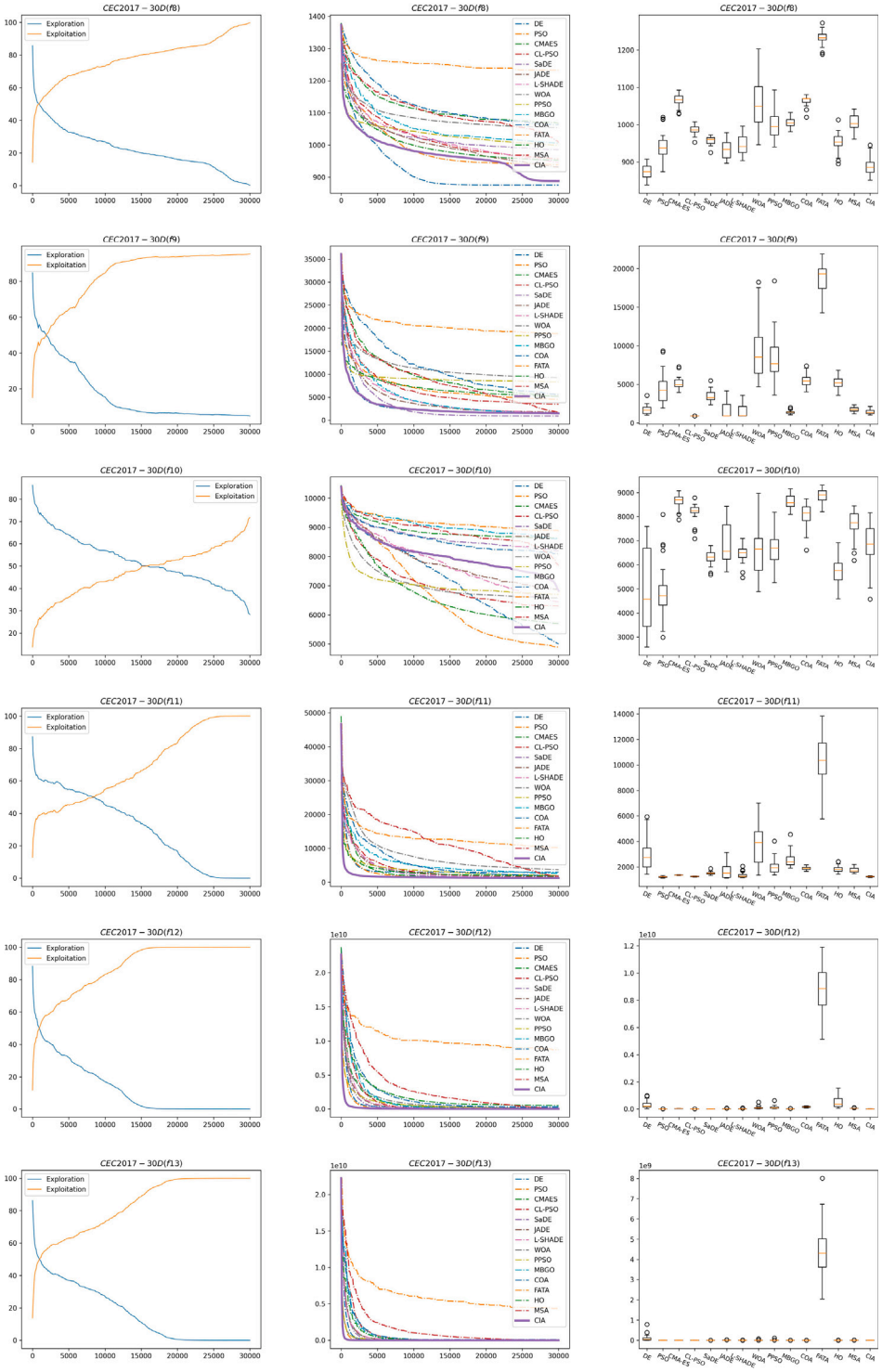

Fig. 19. The results of CEC2017-30D(f_8-f_{13}).

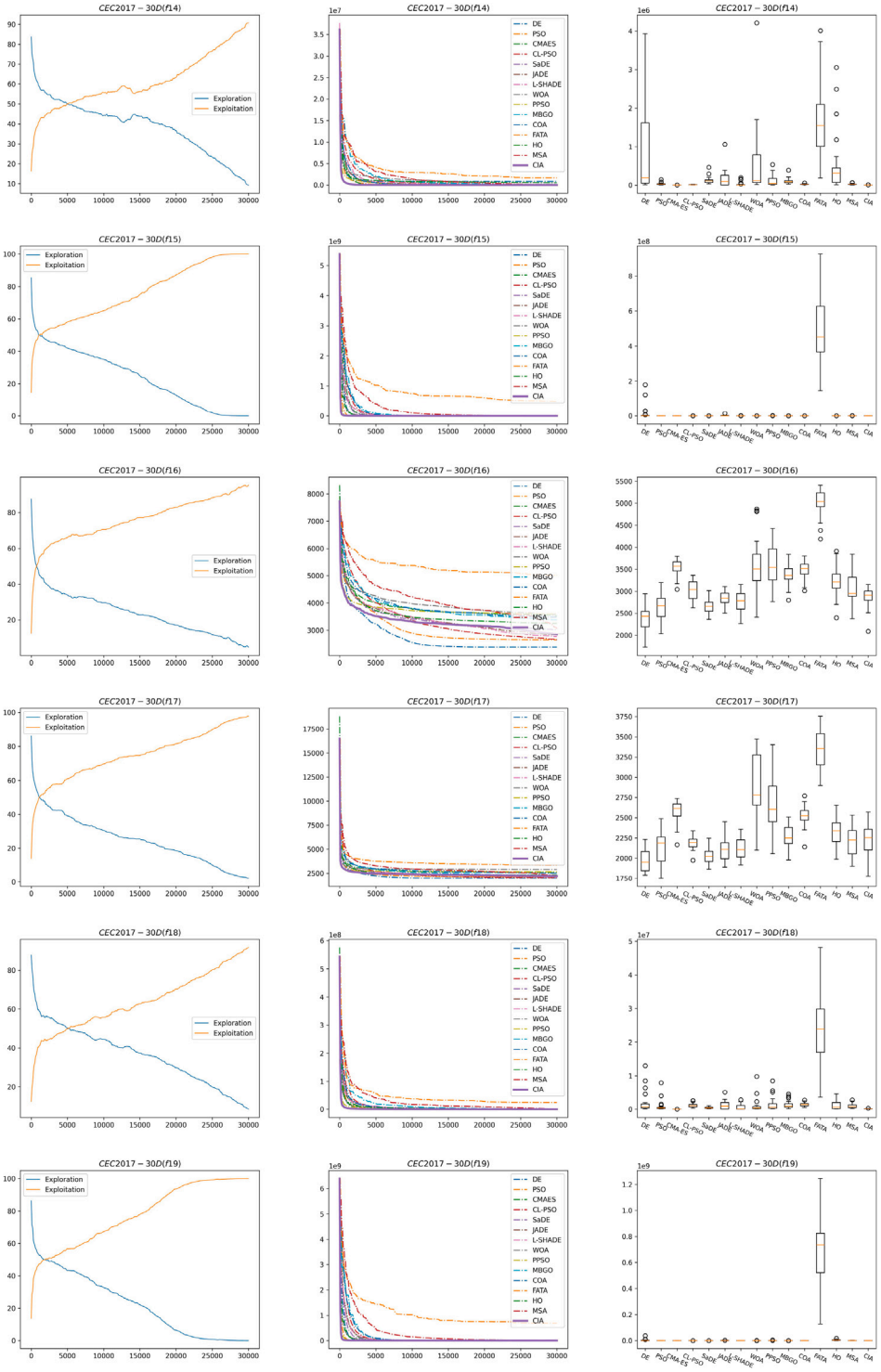

Fig. 20. The results of CEC2017-30D(f_{14}-f_{19}).

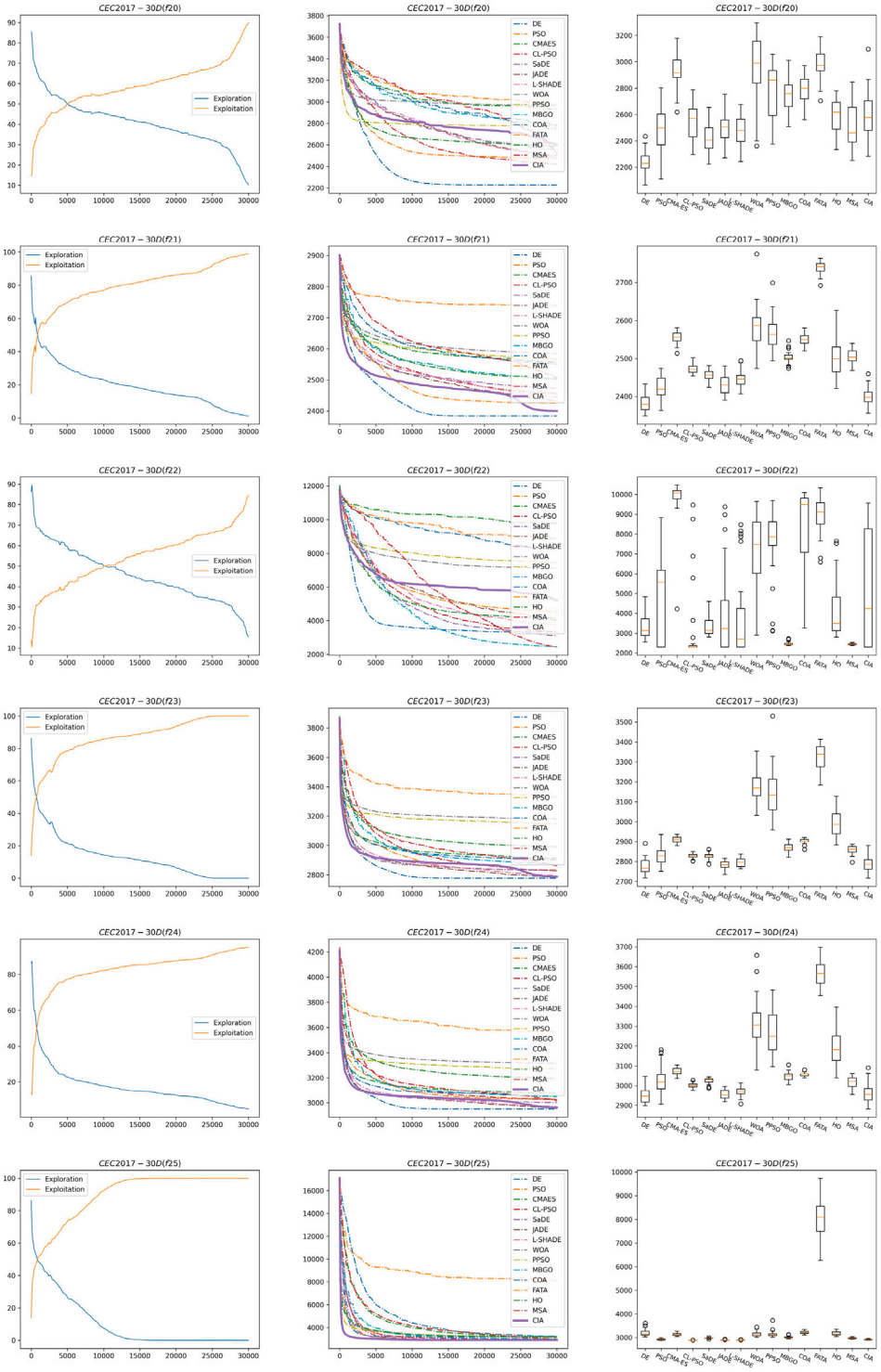

Fig. 21. The results of CEC2017-30D(f_{20}-f_{25}).

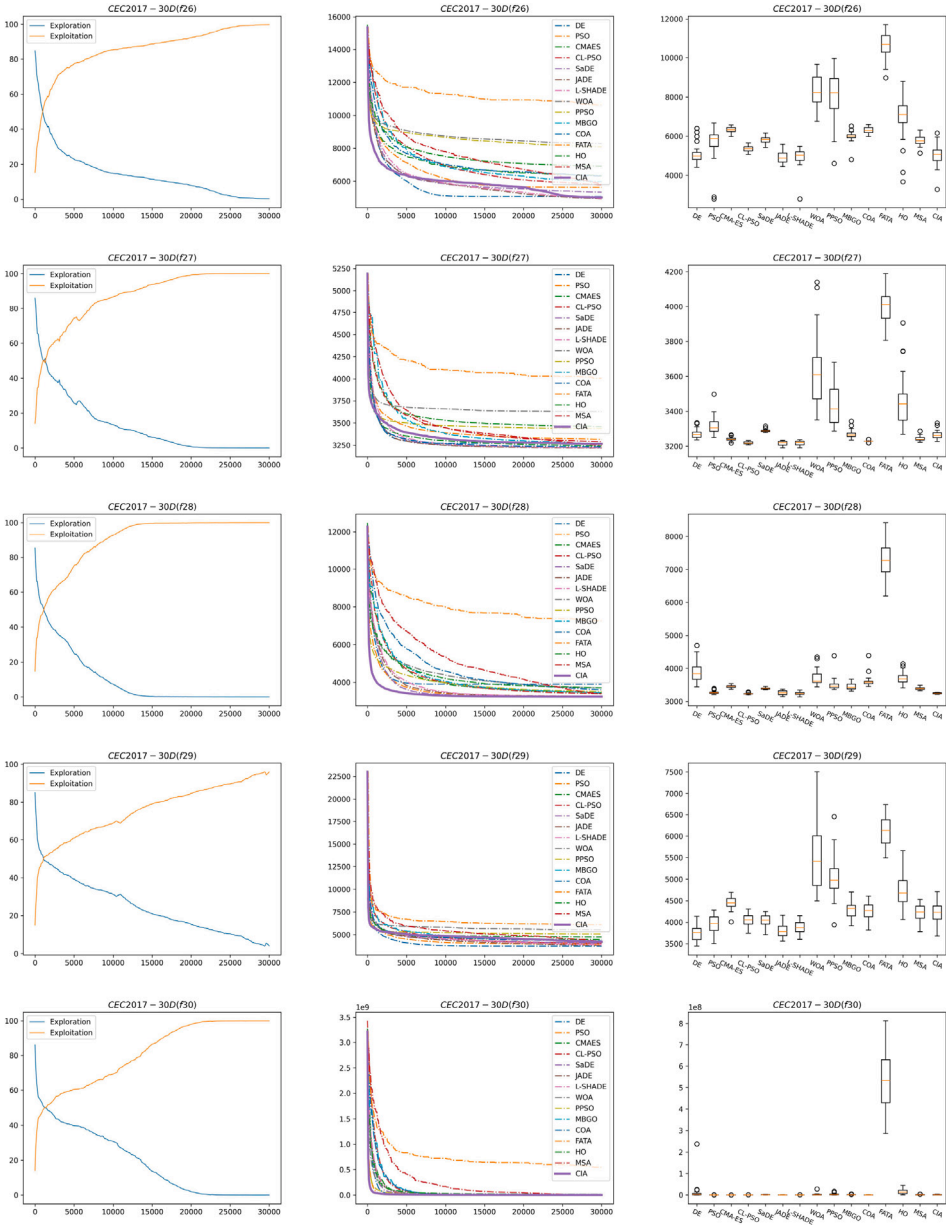

Fig. 22. The results of CEC2017-30D(f_{26}-f_{30}).

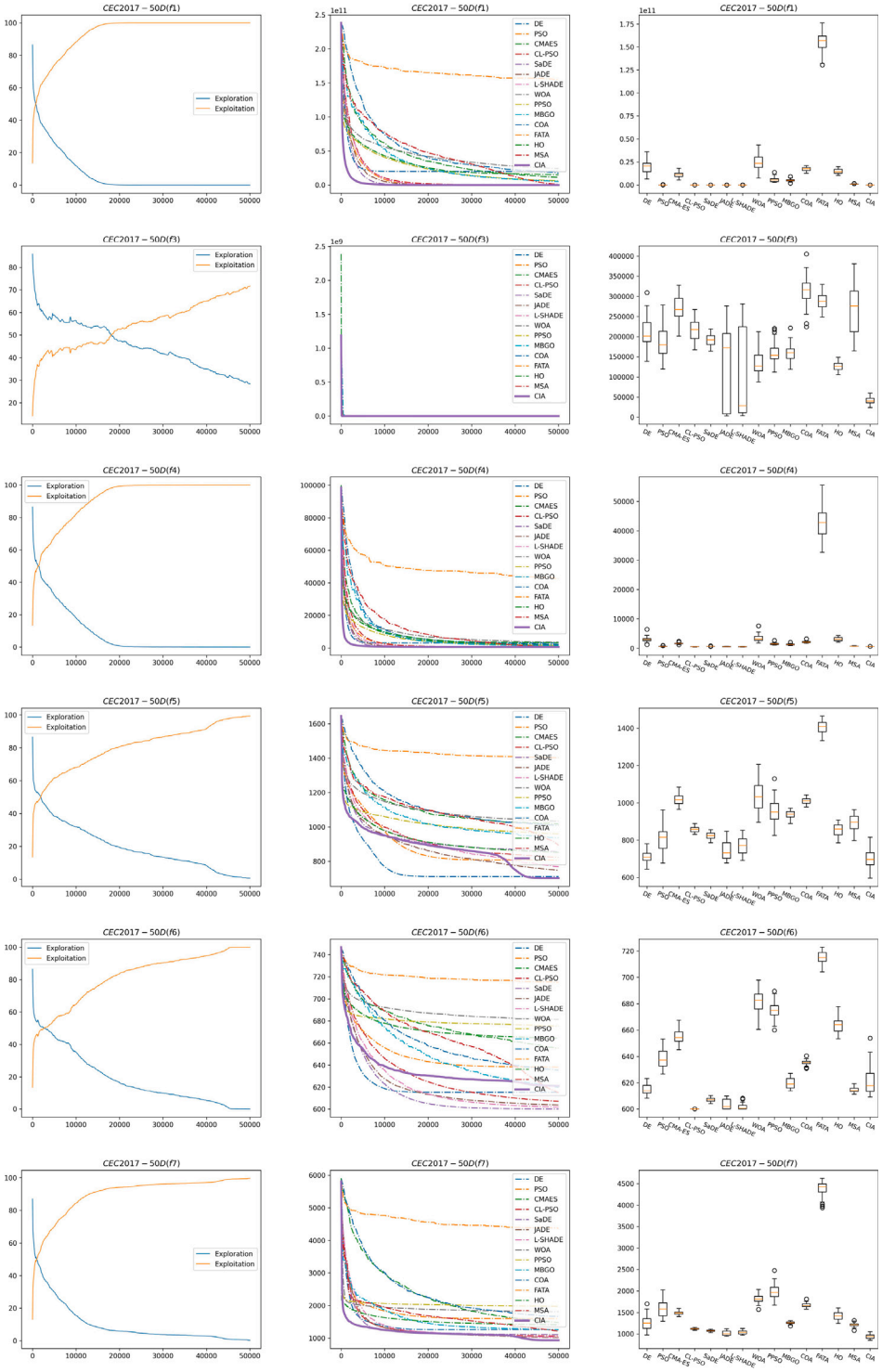

Fig. 23. The results of CEC2017-50D(f_1-f_7).

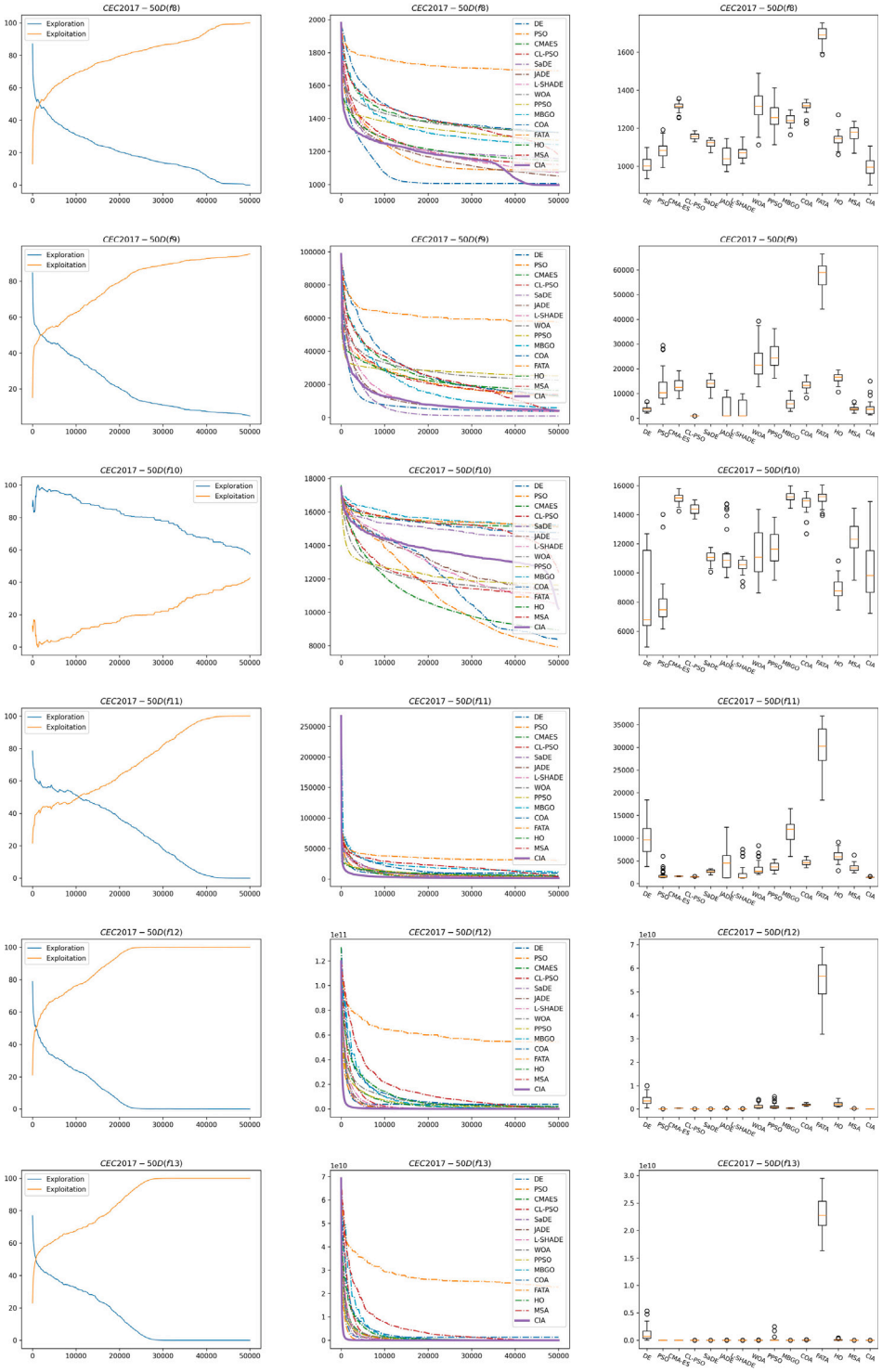

Fig. 24. The results of CEC2017-50D(f_8-f_{13}).

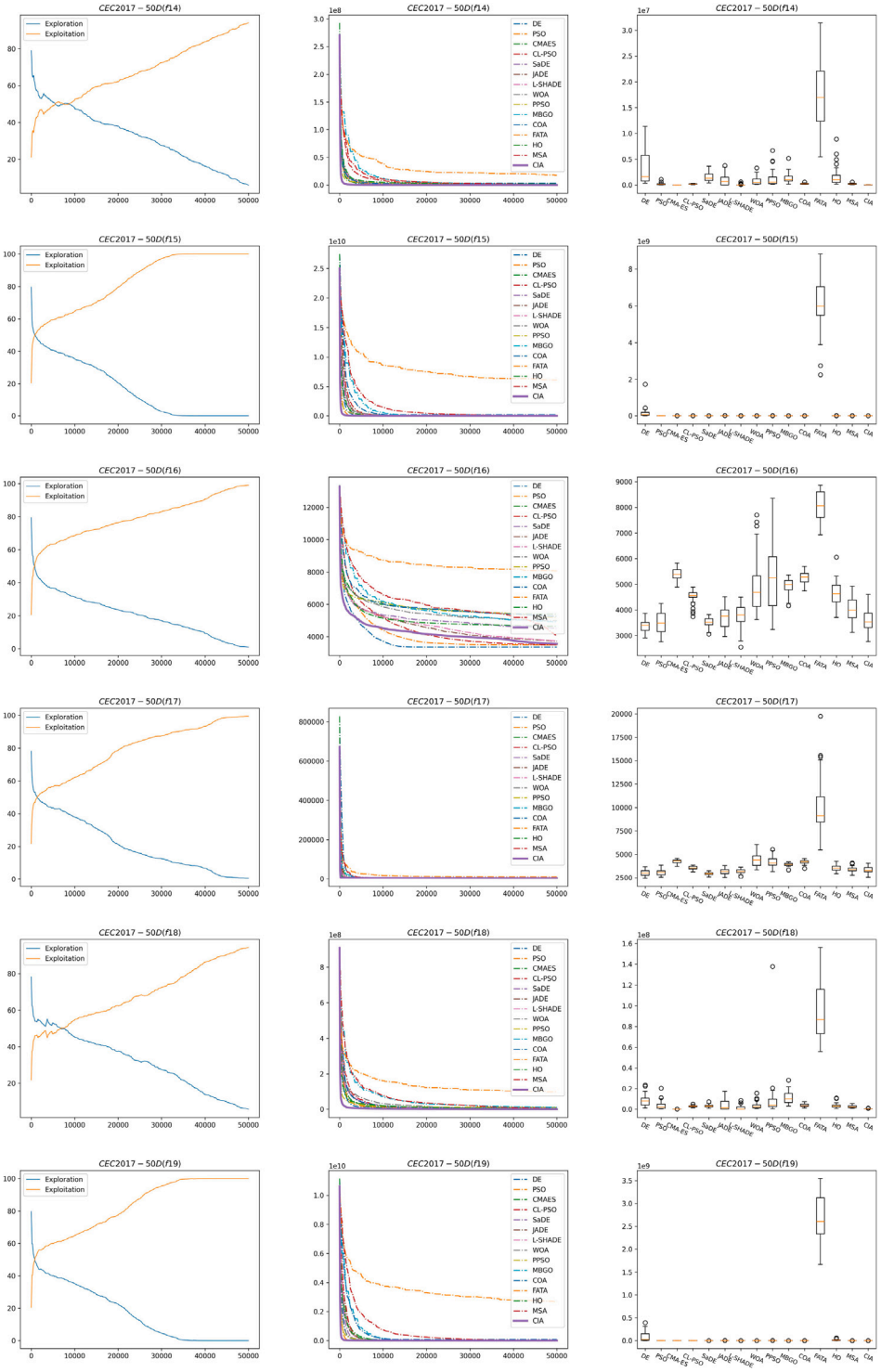

Fig. 25. The results of CEC2017-50D(f_{14}-f_{19}).

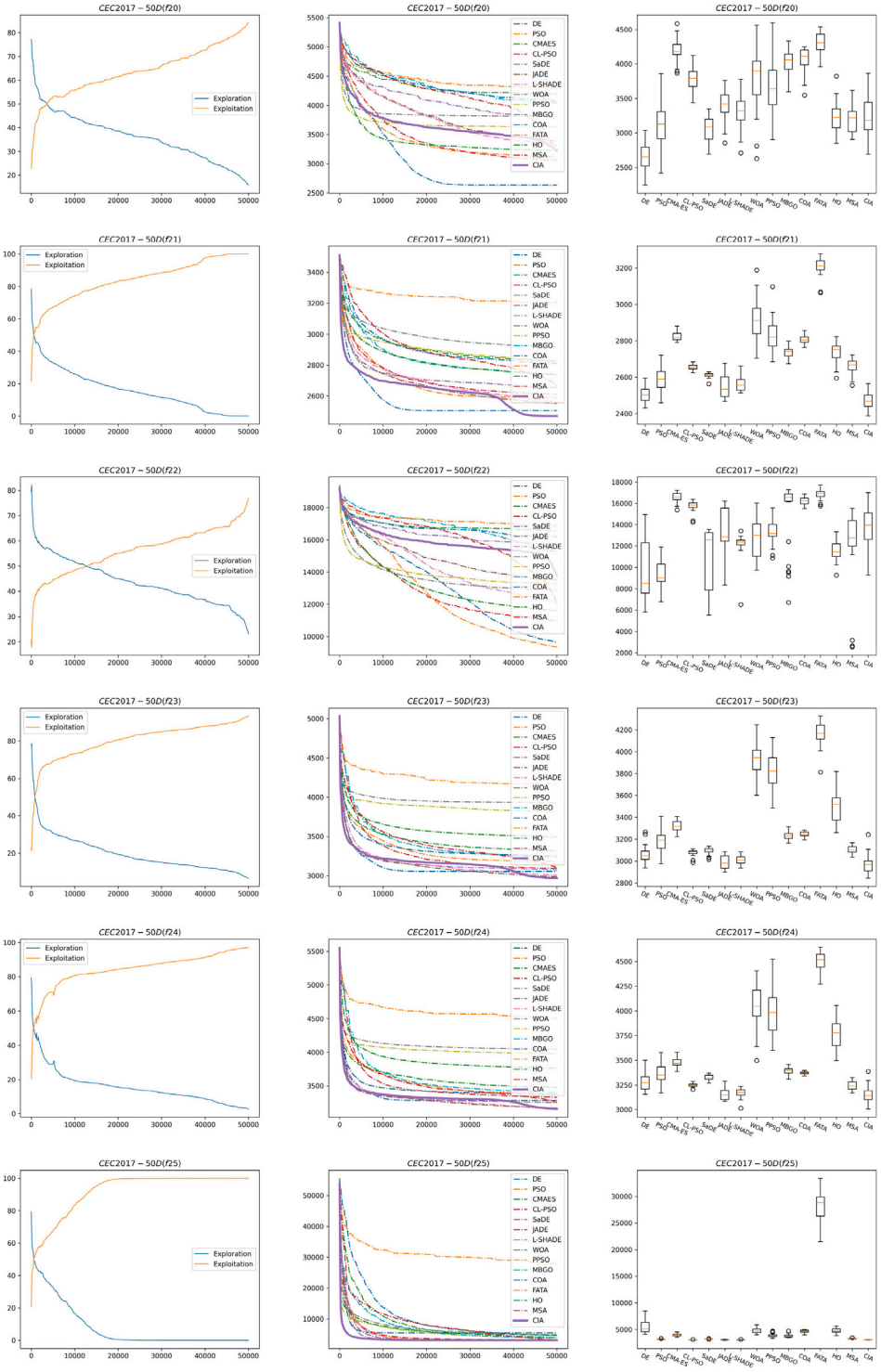

Fig. 26. The results of CEC2017-50D(f_{20}-f_{25}).

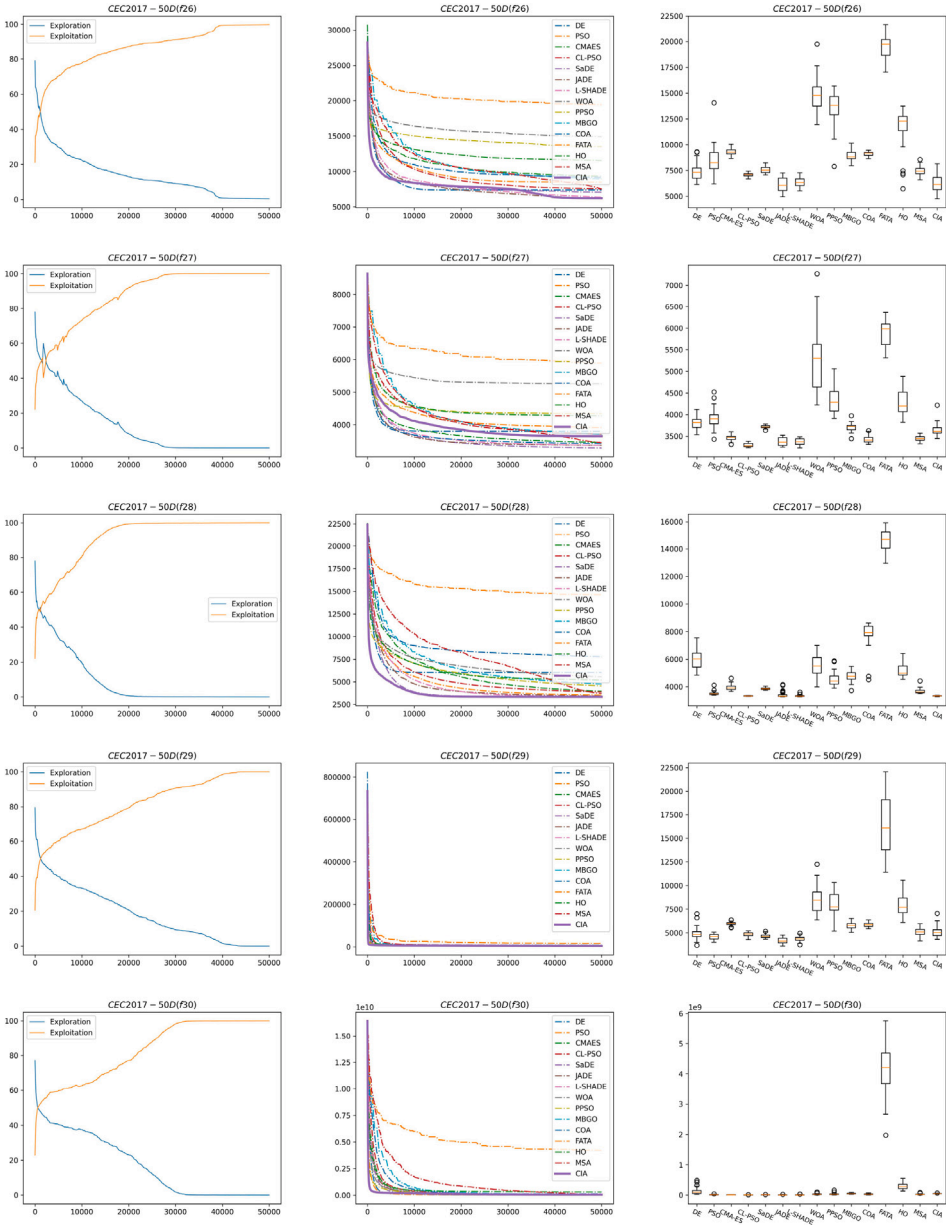

Fig. 27. The results of CEC2017-50D(f_{26}-f_{30}).

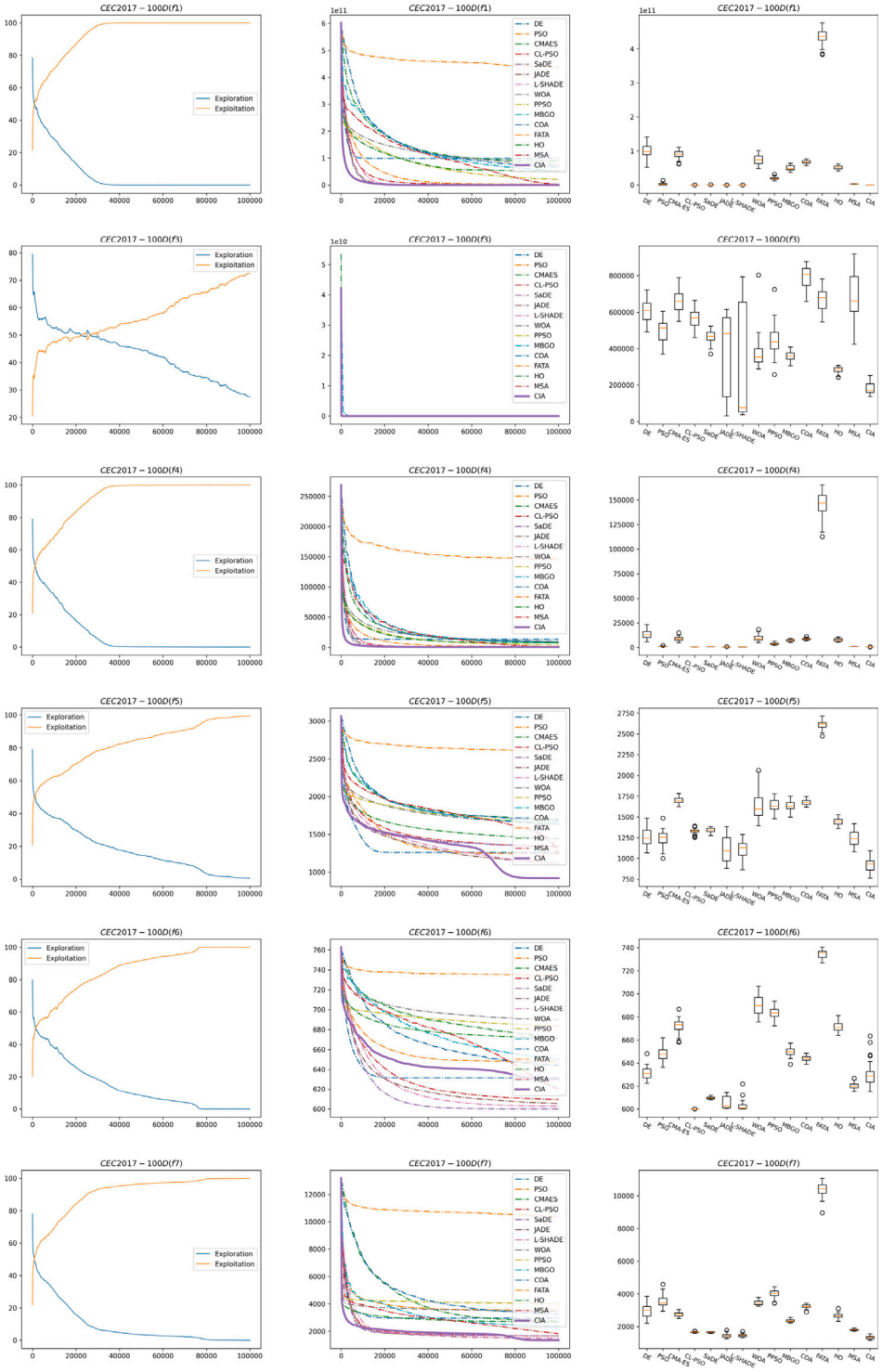

Fig. 28. The results of CEC2017-100D(f_1-f_7).

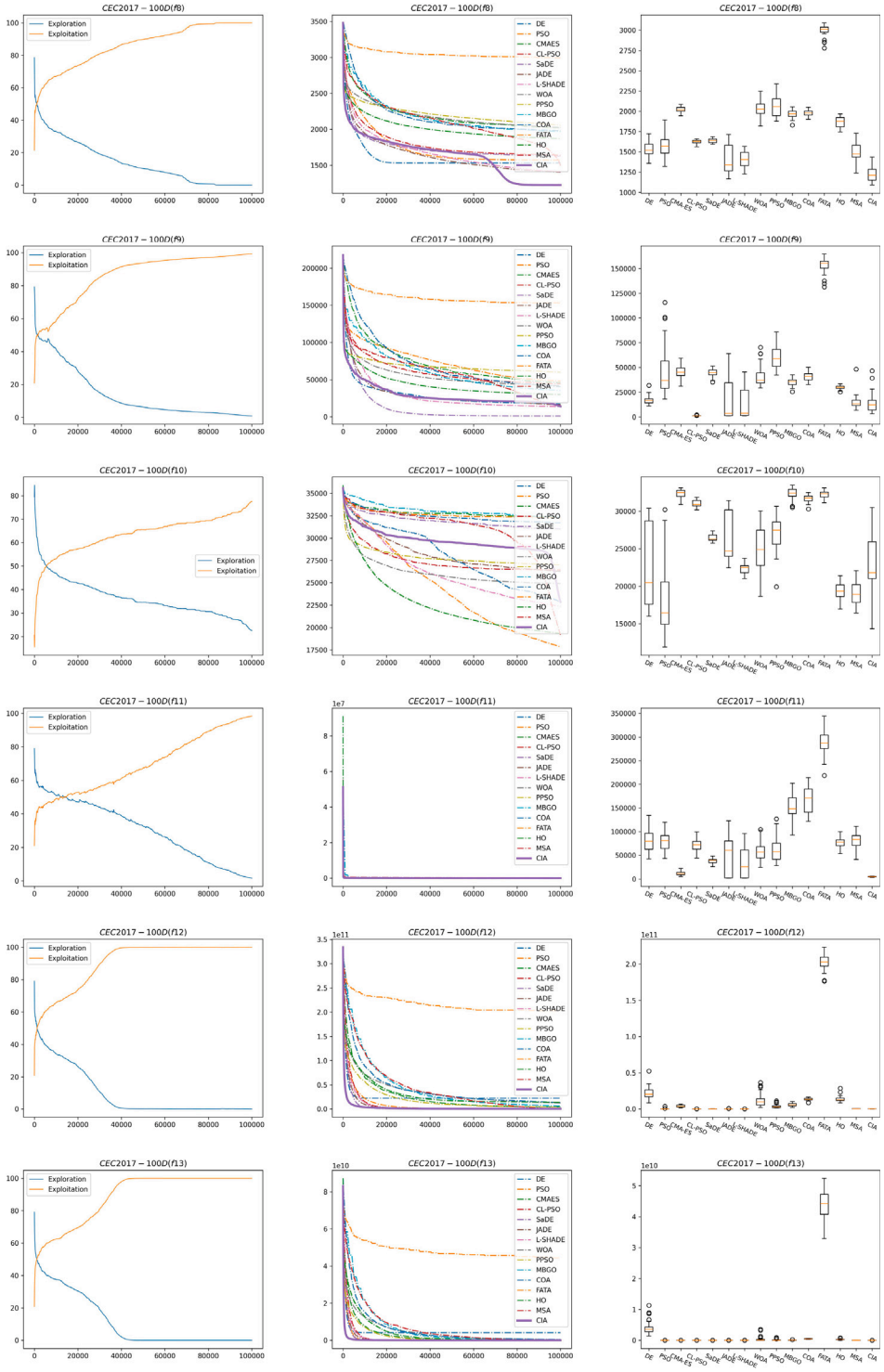

Fig. 29. The results of CEC2017-100D(f_8-f_{13}).

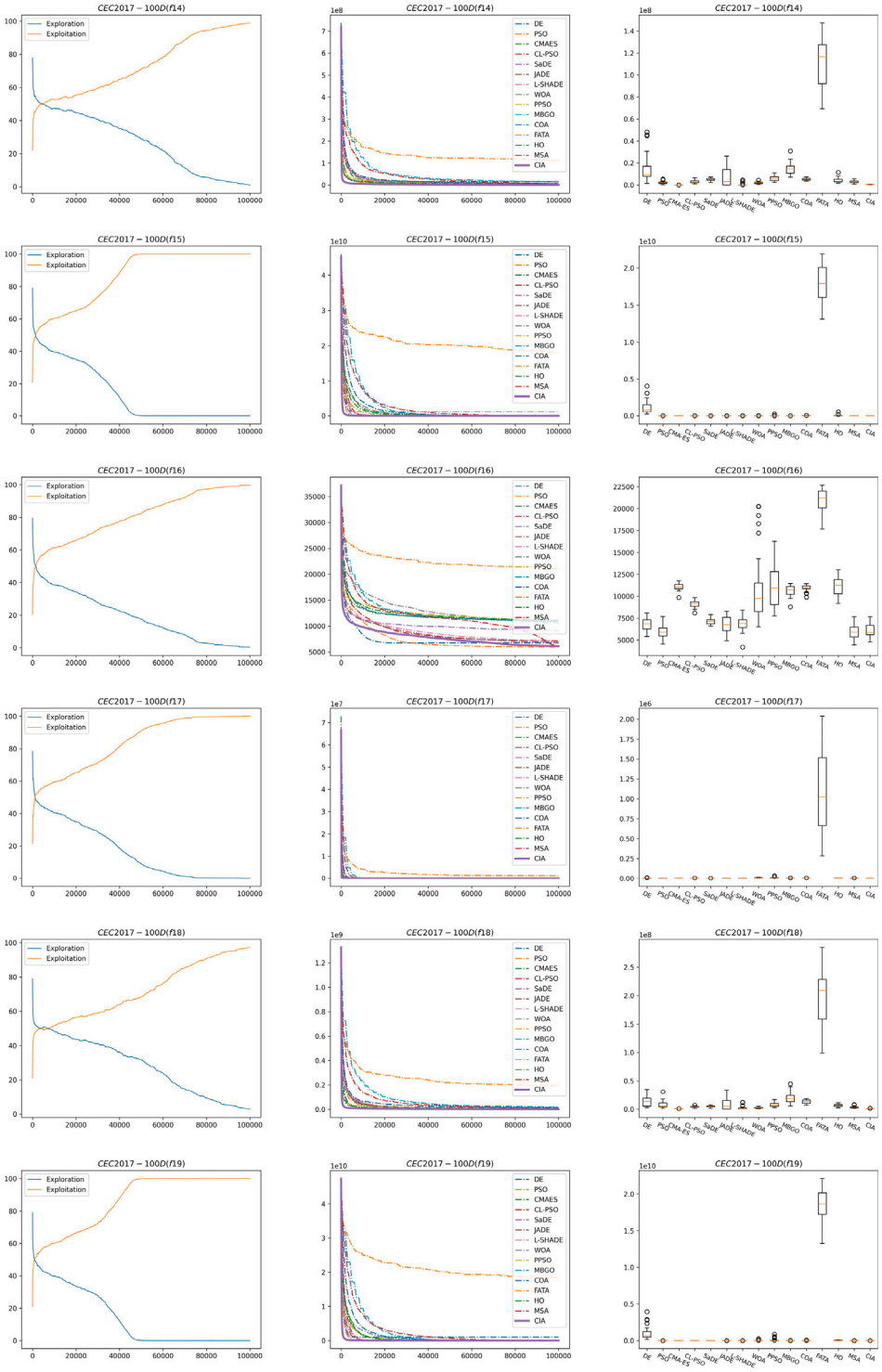

Fig. 30. The results of CEC2017-100D(f_{14}-f_{19}).

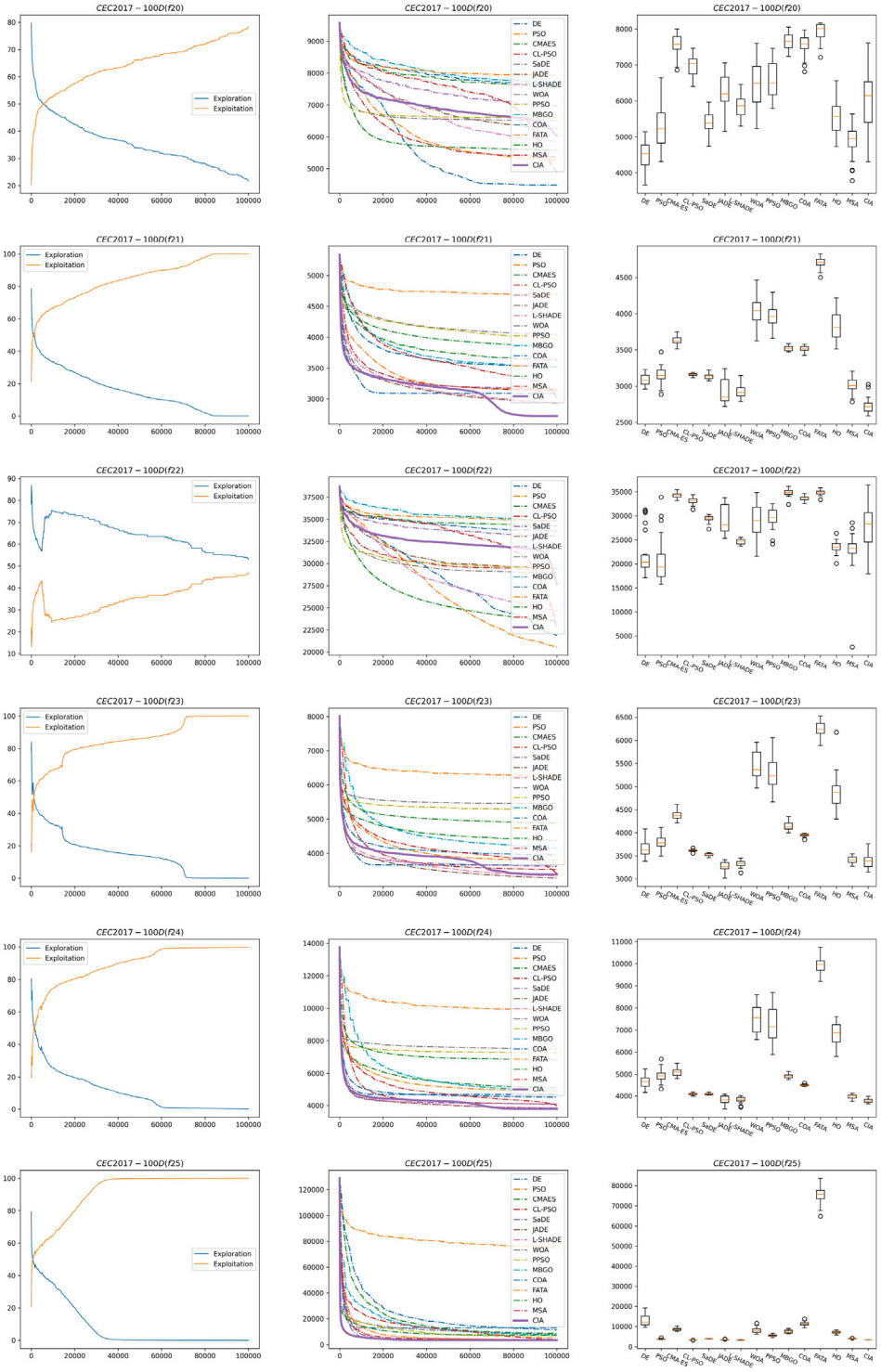

Fig. 31. The results of CEC2017-100D(f_{20}-f_{25}).

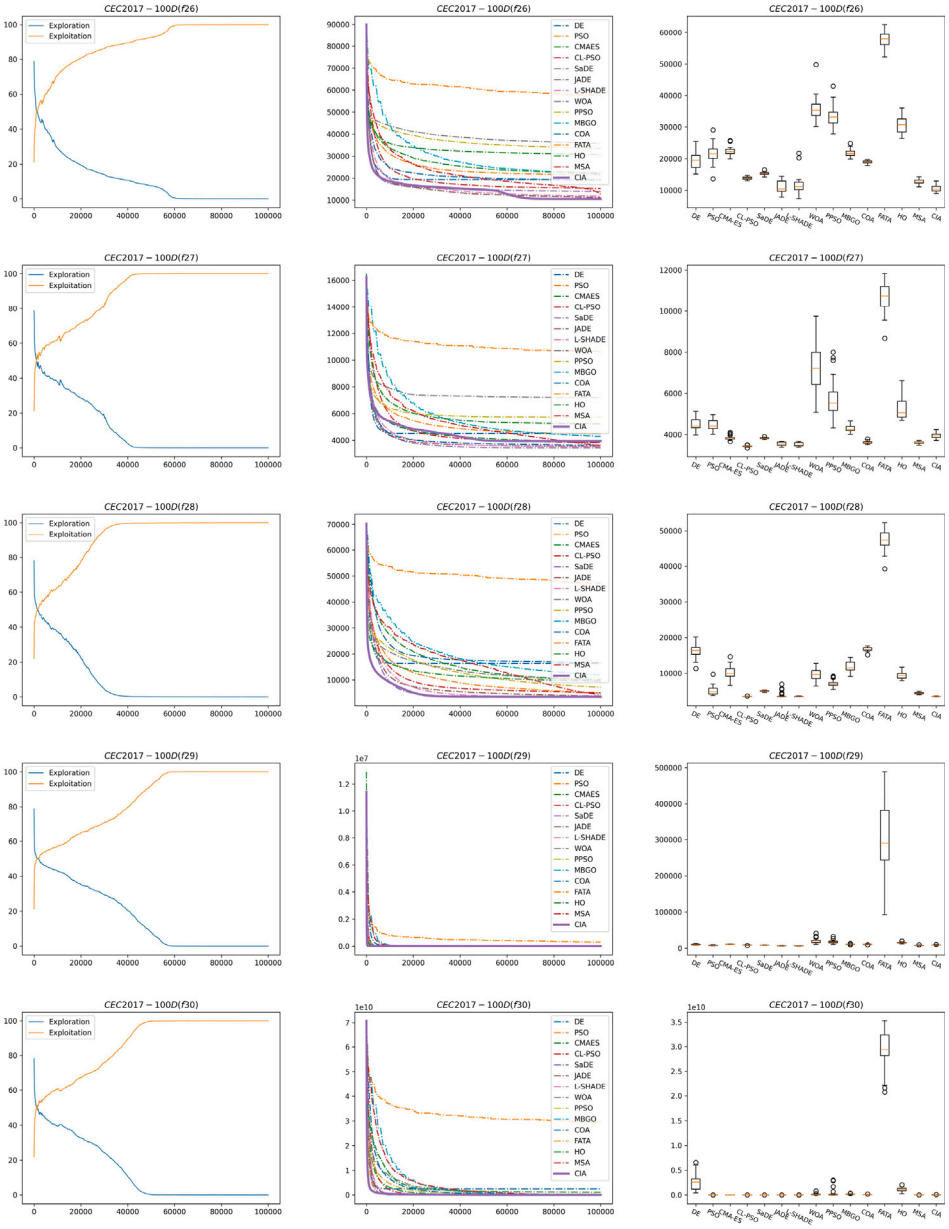

Fig. 32. The results of CEC2017-100D(f_{26}-f_{30}).

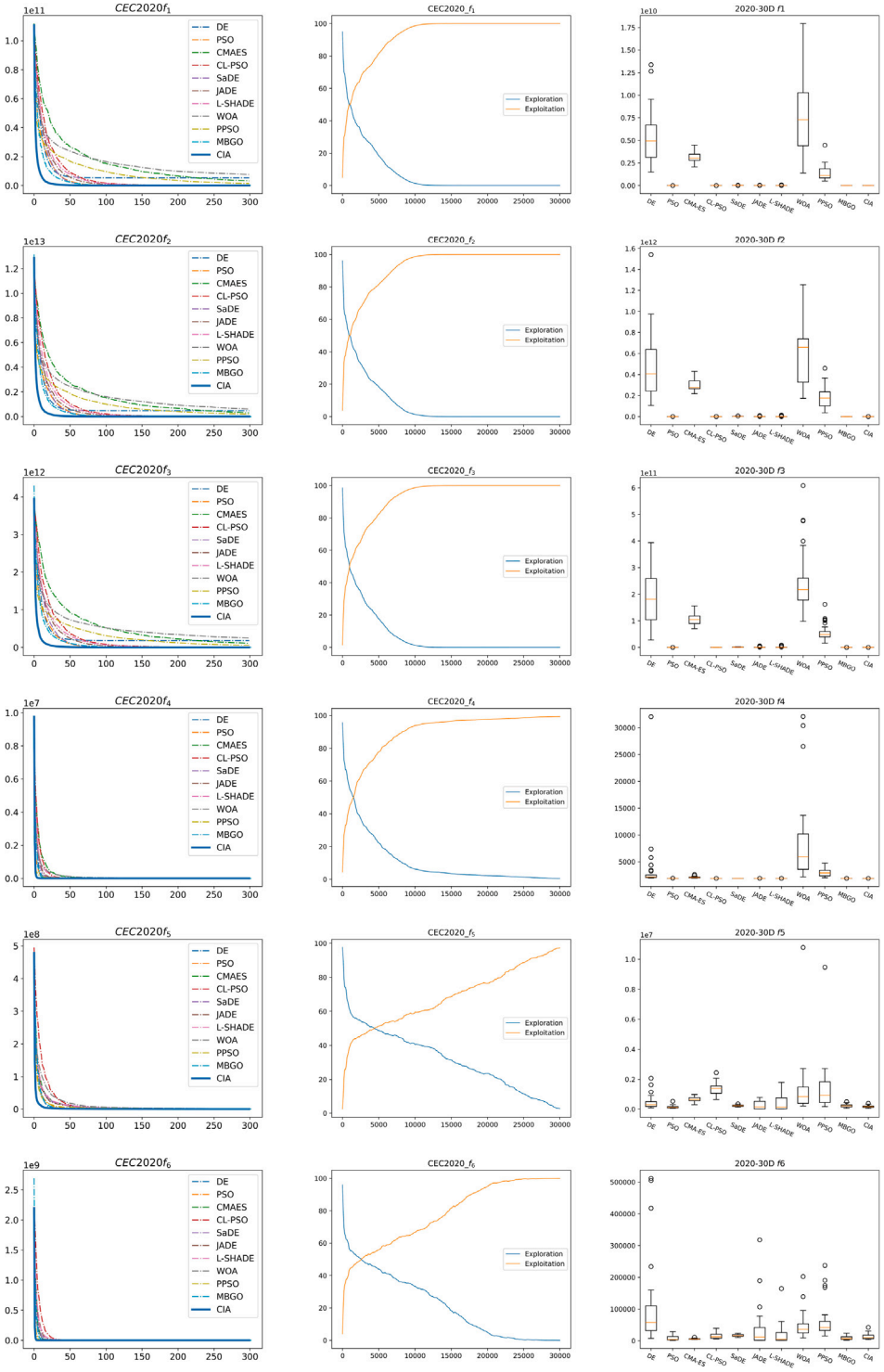

Fig. 33. The results of CEC2022-10D(f_1-f_6).

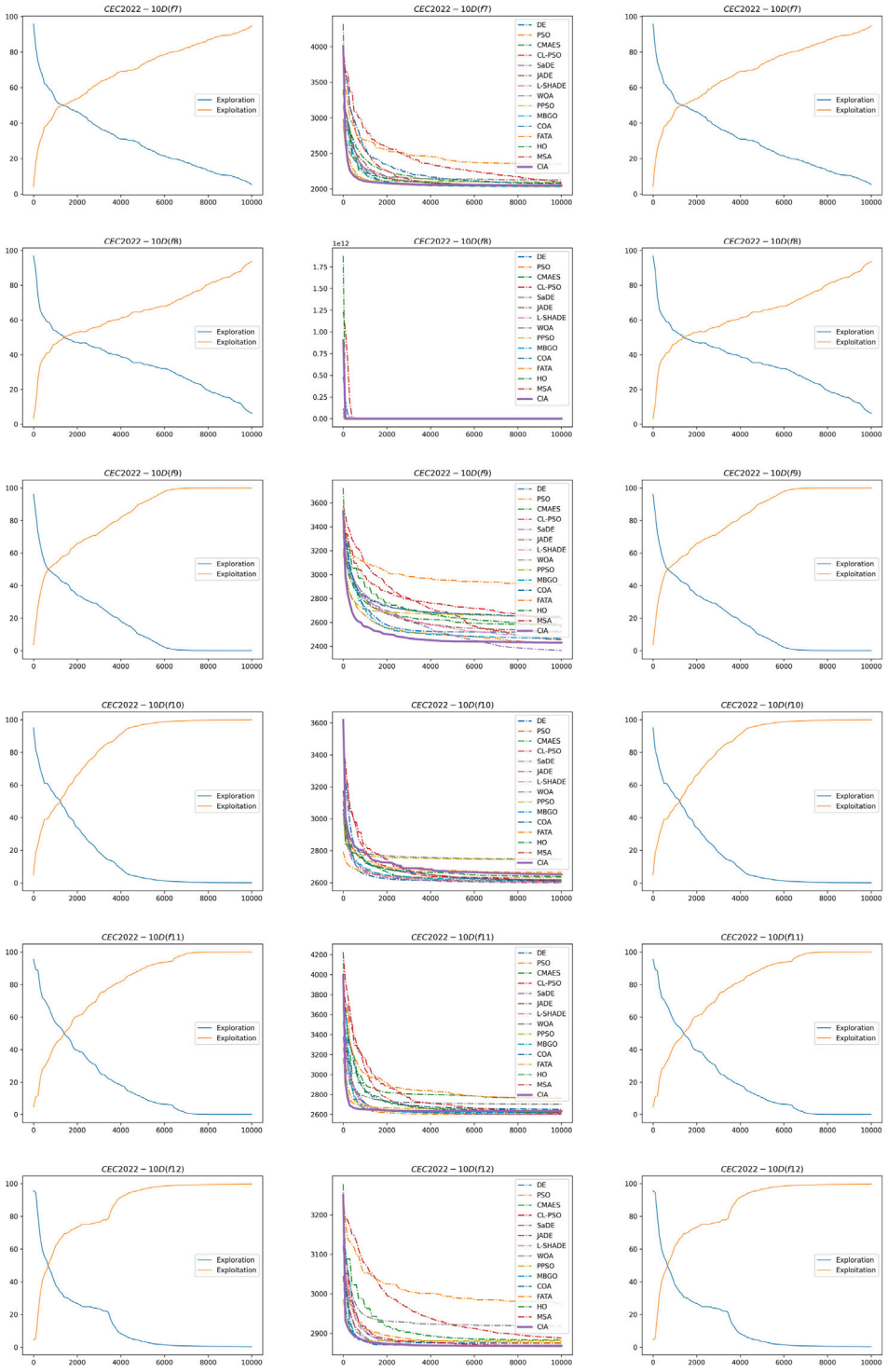

Fig. 34. The results of CEC2022-10D(f_7-f_{12}).

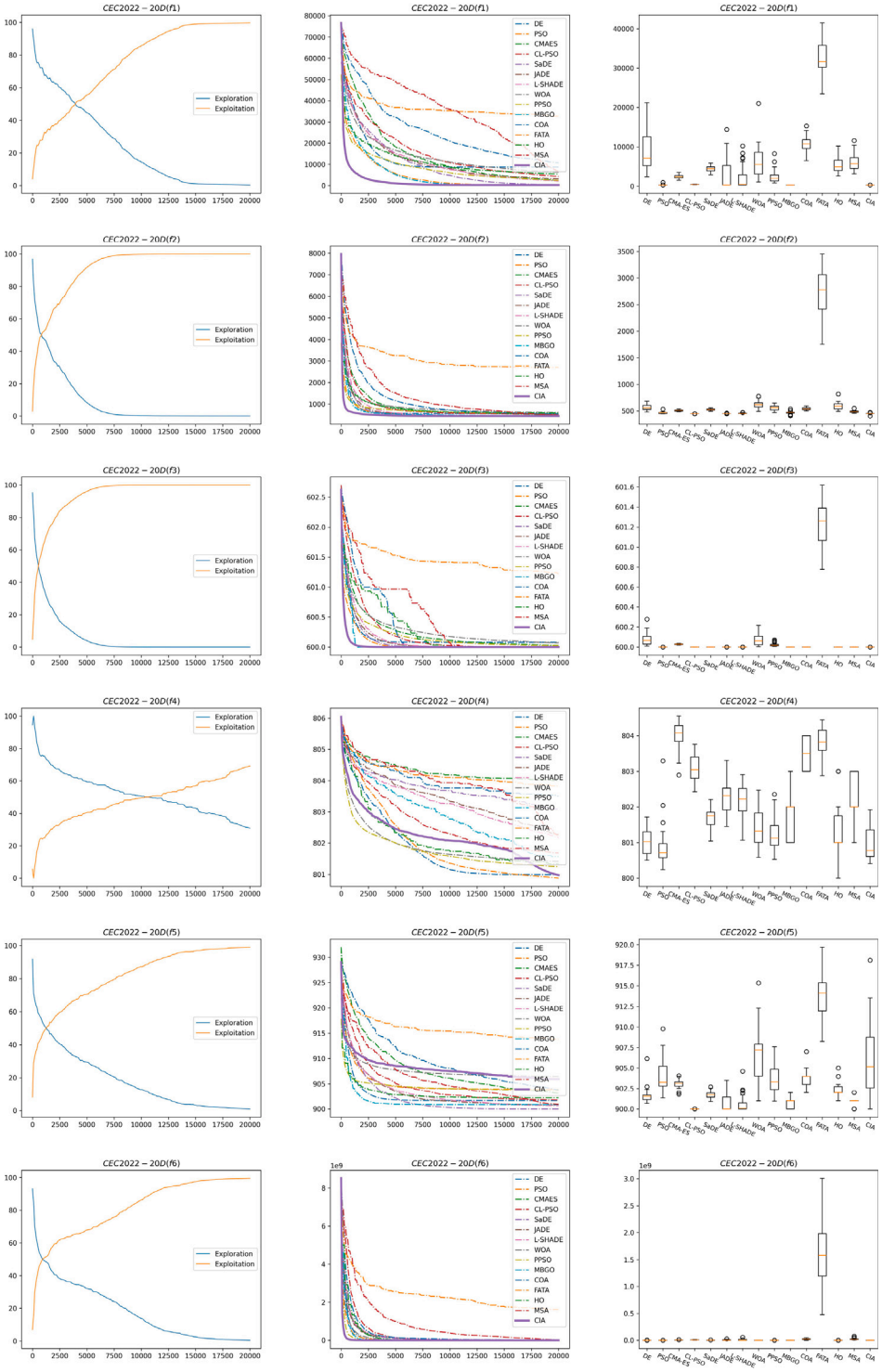

Fig. 35. The results of CEC2022-20D(f_1-f_6).

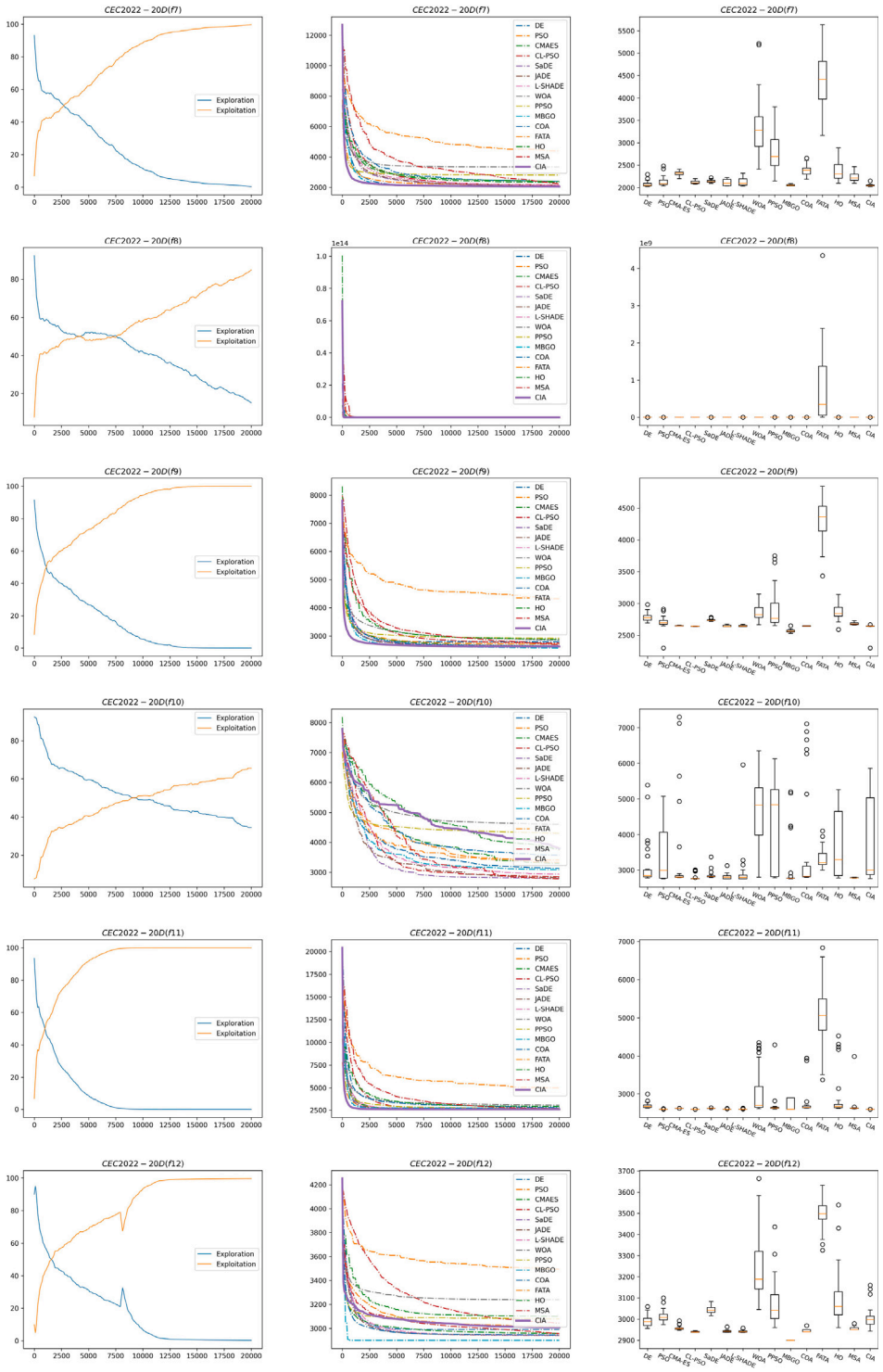

Fig. 36. The results of CEC2022-20D(f_7-f_{12}).

References

[1] T. Back, U. Hammel, H.-P. Schwefel, Evolutionary computation: comments on the history and current state, IEEE Trans. Evol. Comput. 1 (1) (1997) 3–17.
[2] T. Bäck, D. Fogel, Z. Michalewicz, Handbook of evolutionary computation, Release 97 (1) (1997) B1.
[3] A.E. Eiben, J.E. Smith, Introduction to Evolutionary Computing, 2015.
[4] R. Zhong, J. Yu, DEA2H2: Differential evolution architecture based adaptive hyper-heuristic algorithm for continuous optimization, Cluster Comput. (2024) 1–28.
[5] T.J. Ypma, Historical development of the Newton–Raphson method, SIAM Rev. 37 (4) (1995) 531–551.
[6] C.G. Broyden, Quasi-Newton methods and their application to function minimisation, Math. Comp. 21 (1967) 368–381.
[7] A. Schrijver, Theory of linear and integer programming, in: Wiley-Interscience Series in Discrete Mathematics and Optimization, 1986.
[8] S. Amari, Backpropagation and stochastic gradient descent method, Neurocomputing 5 (4) (1993) 185–196.
[9] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, M. Protasi, Complexity and approximation: Combinatorial optimization problems and their approximability properties, 2012.
[10] U.M. Diwekar, Introduction to Applied Optimization, vol. 22, Springer Nature, 2020.
[11] P. Jain, P. Kar, Non-Convex Optimization for Machine Learning, 2017.
[12] S. Das, S. Maity, B. Qu, P.N. Suganthan, Real-parameter evolutionary multimodal optimization — A survey of the state-of-the-art, Swarm Evol. Comput. 1 (2) (2011) 71–88.
[13] M. Fremond, M.V. Shitikova, Non-smooth thermomechanics, Appl. Mech. Rev. 55 (5) (2002) B99–B100.
[14] 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 Comput. Appl. 36 (2024) 6721–6740.
[15] T. Hegazy, Optimization of resource allocation and leveling using genetic algorithms, J. Constr. Eng. Manag. 125 (3) (1999) 167–175.
[16] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, in: P.L. Hammer, E.L. Johnson, B.H. Korte (Eds.), Discrete Optimization II, in: Annals of Discrete Mathematics, vol. 5, Elsevier, 1979, pp. 287–326.
[17] D. Whitley, A genetic algorithm tutorial, Stat. Comput. (1994) 65–85.
[18] L.M. Schmitt, Theory of genetic algorithms, Theoret. Comput. Sci. 259 (1) (2001) 1–61.
[19] S. Furnari, D. Crilly, V.F. Misangyi, T. Greckhamer, P.C. Fiss, R.V. Aguilera, Capturing causal complexity: Heuristics for configurational theorizing, Acad. Manag. Rev. 46 (4) (2021) 778–799.
[20] T. Back, Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms, Oxford University Press, 1996.
[21] J. Yu, Vegetation evolution: An optimization algorithm inspired by the life cycle of plants, Int. J. Comput. Intell. Appl. 21 (02) (2022) 2250010.
[22] H. Beyer, H. Schwefel, Evolution strategies–a comprehensive introduction, Nat. Comput. 1 (2002) 3–52.
[23] S. Das, P.N. Suganthan, Differential evolution: A survey of the state-of-the-art, IEEE Trans. Evol. Comput. 15 (1) (2011) 4–31.
[24] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, vol. 4, 1995, pp. 1942–1948.
[25] S. Mirjalili, A. Lewis, The whale optimization algorithm, Adv. Eng. Softw. 95 (2016) 51–67.
[26] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey Wolf Optimizer, Adv. Eng. Softw. 69 (2014) 46–61.
[27] F.A. Hashim, E.H. Houssein, K. Hussain, M.S. Mabrouk, W. Al-Atabany, Honey badger algorithm: New metaheuristic algorithm for solving optimization problems, Math. Comput. Simulation 192 (2022) 84–110.
[28] Y. Xu, R. Zhong, C. Zhang, J. Yu, Multiplayer battle game-inspired optimizer for complex optimization problems, Cluster Comput. (2024) 1–25.
[29] T. Rahkar Farshi, Battle royale optimization algorithm, Neural Comput. Appl. 33 (4) (2021) 1139–1157.
[30] M. Dehghani, P. Trojovský, Teamwork optimization algorithm: A new optimization approach for function minimization/maximization, Sensors 21 (13) (2021).
[31] P.J.M. van Laarhoven, Emile H.L. Aarts, Simulated annealing, in: Simulated Annealing: Theory and Applications, Springer Netherlands, Dordrecht, 1987, pp. 7–15.
[32] S. Mirjalili, S.M. Mirjalili, A. Hatamlou, Multi-verse optimizer: a nature-inspired algorithm for global optimization, Neural Comput. Appl. 27 (2016) 495–513.
[33] Z. Wei, C. Huang, X. Wang, T. Han, Y. Li, Nuclear reaction optimization: A novel and powerful physics-based algorithm for global optimization, IEEE Access 7 (2019) 66084–66109.
[34] M.W. Krentel, The complexity of optimization problems, J. Comput. System Sci. 36 (3) (1988) 490–509.
[35] G.I. Diaz, A. Fokoue-Nkoutche, G. Nannicini, H. Samulowitz, An effective algorithm for hyperparameter optimization of neural networks, IBM J. Res. Dev. 61 (4/5) (2017) 9:1–9:11.
[36] J. Frankle, D.J. Schwab, A.S. Morcos, The early phase of neural network training, 2020, http://dx.doi.org/10.48550/arXiv.2002.10365, arXiv preprint arXiv:2002.10365.
[37] M.I. Jordan, T.M. Mitchell, Machine learning: Trends, perspectives, and prospects, Science 349 (6245) (2015) 255–260.
[38] B. Zoph, Q. Le, Neural architecture search with reinforcement learning, in: International Conference on Learning Representations, 2017.
[39] X. Wang, Y. Zhao, F. Pourpanah, Recent advances in deep learning, Int. J. Mach. Learn. Cybern. 11 (2020) 747–750.
[40] A. Gandomi, M. Haider, Beyond the hype: Big data concepts, methods, and analytics, Int. J. Inf. Manage. 35 (2) (2015) 137–144.
[41] N.D. S, S.P. Rajagopalan, A study on feature selection techniques in bio-informatics, Int. J. Adv. Comput. Sci. Appl. 2 (1) (2011).
[42] T. Fischer, C. Krauss, Deep learning with long short-term memory networks for financial market predictions, European J. Oper. Res. 270 (2) (2018) 654–669.
[43] F. Altiparmak, M. Gen, L. Lin, T. Paksoy, A genetic algorithm approach for multi-objective optimization of supply chain networks, Comput. Ind. Eng. 51 (1) (2006) 196–215, Special Issue on Computational Intelligence and Information Technology: Applications to Industrial Engineering.
[44] Z. Chen, X. Li, M. Yang, H. Zhang, X. Xu, Optimization of deep learning models for the prediction of gene mutations using unsupervised clustering, J. Pathology: Clin. Res. 9 (1) (2023) 3–17.
[45] N. Namura, Single and multi-objective benchmark problems focusing on human-powered aircraft design, 2023, http://dx.doi.org/10.48550/arXiv.2312.08953, arXiv preprint arXiv:2312.08953.
[46] N. Hansen, A. Ostermeier, Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation, in: Proceedings of IEEE International Conference on Evolutionary Computation, 1996, pp. 312–317.
[47] W. Shahzad, F.A. Khan, A.B. Siddiqui, Clustering in mobile ad hoc networks using Comprehensive Learning Particle Swarm Optimization (CLPSO), in: Communication and Networking, Springer Berlin Heidelberg, 2009, pp. 342–349.
[48] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput. 13 (2) (2009) 398–417.
[49] J. Zhang, A.C. Sanderson, JADE: Adaptive differential evolution with optional external archive, IEEE Trans. Evol. Comput. 13 (5) (2009) 945–958.
[50] 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, pp. 1658–1665.
[51] M. Ghasemi, E. Akbari, A. Rahimnejad, S.E. Razavi, S. Ghavidel, L. Li, Phasor particle swarm optimization: a simple and efficient variant of PSO, Soft Comput. 23 (2019) 9701–9718.
[52] Ailiang Qi, Dong Zhao, Ali Asghar Heidari, Lei Liu, Yi Chen, Huiling Chen, FATA: An efficient optimization method based on geophysics, Neurocomputing 607 (2024) 128289.
[53] Mohamed Abdel-Basset, Reda Mohamed, Mahinda Zidan, Mohammed Jameel, Mohamed Abouhawwash, Mantis search algorithm: A novel bio-inspired algorithm for global optimization and engineering design problems, Comput. Methods Appl. Mech. Engrg. 415 (2023) 116200.
[54] Mohammad Hussein Amiri, Nastaran Mehrabi Hashjin, Mohsen Montazeri, Seyedali Mirjalili, Nima Khodadadi, Hippopotamus optimization algorithm: a novel nature-inspired optimization algorithm, Sci. Rep. 14 (1) (2024) 5032.
[55] Mohammad Dehghani, Zeinab Montazeri, Eva Trojovská, Pavel Trojovský, Coati optimization algorithm: A new bio-inspired metaheuristic algorithm for solving optimization problems, Knowl.-Based Syst. 259 (2023) 110011.
[56] John C. W., S. I., M. K., S. W., H. Sakai, N. Y., M., K. C., Biology of a critically endangered species, the Toki (Japanese Crested Ibis) Nipponia nippon, Ibis 142 (1) (2000) 1–11.
[57] Yuanxing Ye, Yiting Jiang, Canshi Hu, Yao Liu, Baoping Qing, Chao Wang, Esteban Fernández-Juricic, Changqing Ding, What makes a tactile forager join mixed-species flocks? A case study with the endangered Crested Ibis (Nipponia nippon), Auk 134 (2) (2017) 421–431.
[58] Q. Ye, X. Wu, S. Lu, J. Zhang, The foraging behavior of the crested ibis (Nipponia nippon) in different habitats in the wild, Zoöl. Res. 38 (3) (2017) 200–210.
[59] Z.W. Zhang, C.Q. Ding, Y. Wu, J.Q. Zhang, Foraging habitat selection by the crested ibis (Nipponia nippon) in the wild, Zoöl. Res. 25 (2) (2004) 167–172.
[60] P. Domenici, R.W. Blake, The kinematics and performance of fish fast-start swimming, J. Exp. Biol. 200 (8) (1997) 1165–1178.
[61] H. Higuchi, The conservation of the crested ibis and other endangered bird species in Japan, J. Ornithol. 138 (1) (1997) 29–35.
[62] BirdLife International, Threatened Birds of Asia: The Birdlife International Red Data Book, BirdLife International, Cambridge, UK, 2001.
[63] J. Yang, X. Han, Y. Wang, Population viability analysis of the crested ibis (Nipponia nippon): Research and conservation, Ecol. Res. 21 (5) (2006) 681–690.
[64] N. Van Thieu, S. Mirjalili, MEALPY: An open-source library for latest meta-heuristic algorithms in Python, J. Syst. Archit. (2023).
[65] N. Van Thieu, Opfunu: An open-source Python library for optimization benchmark functions, J. Open Res. Softw. (2024).