mathematics

MDPI

Article

Elite Guided Jaya Algorithm with Mutation Strategy for Global Optimization and Parameter Extraction

Yiying Zhang

College of Electrical Engineering and Automation, Shandong University of Science and Technology, Qingdao 266590, China; zhangyiying@sdust.edu.cn

Abstract

During the development of metaheuristic algorithms, many classic algorithms have emerged, and the Jaya algorithm (JAYA) is one of them. The inspiration of JAYA comes from the idea that the population should stay away from the worst solution already discovered during the search process, and should approach the best solution already discovered. Although JAYA demonstrates excellent performance in solving simple problems, it is prone to getting stuck in local optima when solving complex problems due to its reliance on the worst and best solutions already discovered. In order to improve the performance of JAYA, this paper proposes an improved JAYA, namely an elite guided Jaya algorithm with mutation strategy (EDJAYA). EDJAYA adopts an elite population driven mechanism to optimize its search strategy, and designs a mutation strategy to enhance its ability to escape from local optima. The excellent optimization performance of EDJAYA has been verified through solving numerical problems and practical problems of photovoltaic model parameter estimation.

Keywords: jaya algorithm; elite individuals; global optimization; photovoltaic model

MSC: 65K10

1. Introduction

The Jaya algorithm (JAYA) is a very simple but efficient metaheuristic method [1]. The core idea of JAYA is that in the process of population search, individuals in the population should be as far away as possible from the worst solution already obtained and as close as possible to the optimal solution already obtained. Under the guidance of this core idea, the excellent optimization performance of JAYA has been validated [1]. In addition, since its proposal, JAYA and its variants have successfully solved various types of optimization problems, demonstrating its outstanding application value. Rao and Saroj [2] designed a self-adaptive multi-population-based JAYA to solve the engineering optimization problems. Warid et al. [3] aimed to use JAYA to deal with different optimum power flow problems. Mesut and Murat [4] reported a discrete JAYA to solve the classical traveling salesman problem. Ali [5] proposed a fuzzy reinforced Jaya algorithm for solving the structural optimization problems. Cheng et al. [6] presented an improved JAYA to solve the optimization design of a brushless direct Current motor. Das et al. [7] solved the optimal feature selection problem in supervised classification by the JAYA-based wrapper method. Rao et al. [8] showed the application of JAYA in the dimensional optimization of a micro-channel heat sink. Huang et al. [9] employed a prediction-model-guided Jaya algorithm to solve the photovoltaic system maximum power point. Warid et al. [10] put forward a

Check for updates

Academic Editor: Jonathan Blackledge

Received: 7 July 2026
Revised: 31 July 2026
Accepted: 5 August 2026
Published: 12 August 2026

Copyright: © 2026 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

modified JAYA with a new quasi-oppositional to solve the multi-objective optimal power flow solution. Wang et al. [11] used the joint method of stationary and JAYA to solve the intelligent facial emotion recognition problem. Ding et al. [12] adopted an improved JAYA for structural damage identification with the modified objective function based on sparse regularization and Bayesian inference. Chaudhuri and Sahu [13] addressed the micro-array data classification problem by a hybrid feature selection algorithm based on binary JAYA.

Although the above literature has shown that JAYA has achieved success in real-world problems, it is still necessary to design improved versions of JAYA that are more efficient, inspired by the No Free Lunch theory [14]. No Free Lunch theory has various forms of expression:

- An algorithm that has shown success in solving particular optimization problems may fail when applied to others [15].
- While a given metaheuristic algorithm may exhibit optimal performance when applied to a specific set of optimization problems, such success does not engender an unequivocal assurance of efficacy across all conceivable optimization scenarios [16].
- No single algorithm is universally optimal for all problem domains [17].

In summary, according to the No Free Lunch theory, there is no single algorithm that outperforms all others across every problem. In other words, we need to continuously develop new, efficient algorithms to address the diverse types of optimization problems that exist today or may arise in the future. Therefore, from this perspective, it is essential to propose new improved versions of JAYA. This is the first motivation for our current work; the second stems from the inherent shortcomings of JAYA itself, which can be summarized as follows:

- The search strategy of JAYA relies too heavily on the worst and best solutions found by the population. Once the best solution identified by the population is a local optimum, the entire population is liable to become trapped in that local optimum.
- JAYA employs only a single search strategy. All individuals in the population utilize the same update mechanism during the search process, which is detrimental to the maintenance of population diversity.

Based on the two aforementioned sources of motivation, this paper proposes an improved JAYA algorithm: the elite-guided Jaya algorithm with a mutation strategy (EDJAYA). The main features of EDJAYA are as follows: (1) EDJAYA designs an elite-guided search strategy by constructing local and global elite populations; (2) EDJAYA designs a mutation strategy based on local and global elite populations as well as the population's average position to enhance population diversity; and (3) EDJAYA constructs three search strategies from different perspectives to enrich population diversity. The main contribution of this paper can be stated as follows:

- A new variant of JAYA, namely EDJAYA, is proposed.
- According to two types of constructed elite populations, three search strategies incorporating the best and worst solutions found by the population, the population's average position, and elite individuals are introduced to EDJAYA.
- EDJAYA is employed to solve the complex CEC 2022 test suite [18].
- EDJAYA is used for photovoltaic model parameter estimation problems.

The rest of this paper is as follows: Section 2 presents the basic JAYA. Section 3 describes the proposed EDJAYA. Section 4 conducts numerical experiments and presents an analysis and discussion of the results. Section 5 conducts engineering optimization experiments and presents an analysis and discussion of the results. Section 6 summarizes the entire text.

2. JAYA

As previously mentioned, the fundamental idea of JAYA is to drive the individuals in the population as close as possible to the best solution found so far while keeping them as far as possible from the worst solution, thereby achieving the optimization objective. Therefore, JAYA's search strategy is also very simple and can be expressed as follows [1]:

$$v_{i,j} = x_{i,j} + \beta_1 \times (x_{\mathrm{b},i,j} - |x_{i,j}|) - \beta_2 \times (x_{\mathrm{w},i,j} - |x_{i,j}|), i = 1, 2, \ldots, N, j = 1, 2, \ldots, D, \quad (1)$$

where N is the population size, D is the number of decision variables, $v_{i,j}$ is the jth element of vector i, $x_{i,j}$ is the jth element of individual i, $x_{\mathrm{b},j}$ is the jth element of the best solution in the population, $x_{\mathrm{w},j}$ is the jth element of the worst solution in the population, and β_1 and β_2 are two random numbers between 0 and 1. In addition, individual i can be denoted by $\boldsymbol{x}_i (\boldsymbol{x}_i = \{x_{i,j}, i = 1, 2, \ldots, N, j = 1, 2, \ldots, D\})$, the best solution can be denoted by $\boldsymbol{x}_\mathrm{b} (\boldsymbol{x}_\mathrm{b} = \{x_{\mathrm{b},j}, j = 1, 2, \ldots, D\})$, the worst solution can be denoted by $\boldsymbol{x}_\mathrm{w} (\boldsymbol{x}_\mathrm{w} = \{x_{\mathrm{w},j}, j = 1, 2, \ldots, D\})$, and vector i can be denoted by $\boldsymbol{v}_i (\boldsymbol{v}_i = \{v_{i,j}, i = 1, 2, \ldots, N, j = 1, 2, \ldots, D\})$. JAYA has a selection mechanism, which can be written by:

$$\boldsymbol{x}_i = \begin{cases} \boldsymbol{v}_i, \text{if } f(\boldsymbol{v}_i) \leq f(\boldsymbol{x}_i), \\ \boldsymbol{x}_i, \text{if } f(\boldsymbol{v}_i) > f(\boldsymbol{x}_i). \end{cases} \quad (2)$$

Like other metaheuristic algorithms, the population initialization method of JAYA is also randomly generated, which randomly selects N solutions from the solution space that satisfy the constraint conditions, which can be stated as follows:

$$x_{i,j} = l_j + (u_j - l_j) \times \beta_3, i = 1, 2, \ldots, N, j = 1, 2, \ldots, D, \quad (3)$$

where l_j is the lower boundary of the jth element of individual i, and u_j is the upper boundary of the jth element of individual i, and β_3 is a random number between 0 and 1.

Algorithm 1 describes the implementation process of JAYA. Looking at Algorithm 1, JAYA has a very simple structure that can be easily applied to different types of optimization problems.

Algorithm 1 The implementation process of JAYA.

Require: The population size N, the number of decision variables D, the lower boundary of the decision variables $l_\mathrm{j} (j \in [1, D])$, the upper boundary of decision variables $u_\mathrm{j} (j \in [1, D])$, the maximum number of iterations T

```
Implement population initialization by Equation (3);
Evaluate the population and find the optimal solution x_b;
for each t ∈ [1, T] do
    Find out the best solution x_b and the worst solution x_w;
    for each i ∈ [1, N] do
        Generate the vectors v_i by Equation (1);
        Update x_i by Equation (2);
    end for
end for
Output the best solution x_b.
```

3. EDJAYA

This section introduces the proposed EDJAYA, which is divided three parts. The first part describes the created two elite population. The second part presents the designed three learning strategies. The third part shows the detailed implementation of EDJAYA.

3.1. Elite Population

For metaheuristic algorithms, making full use of the already obtained optimal solutions during the population search process is very helpful for finding the global optimum. This is because the obtained optimal solutions usually contain more information about

reaching the global optimum. Therefore, in many classical metaheuristic algorithms, the already obtained optimal solutions are used to guide the search process, such as in Particle Swarm Optimization [19], Grey Wolf Optimization [20], Cuckoo search [21], and so on. Motivated by this, this paper improves the global search capability of the algorithm by saving promising elite individuals and then reusing these elite individuals. The idea of creating local elite populations and global elite populations originated from our previous work [22]. The local elite population $\boldsymbol{X}_{\mathrm{J}}$ is a collection of individuals with potential in the process of individual iteration, and it has N individuals. The update method of $\boldsymbol{X}_{\mathrm{J}}$ can be described as follows: If individual i obtains a better solution in one iteration, that is, it meets $f(\boldsymbol{v}_i) \leq f(\boldsymbol{x}_i)$, then this is the candidate solution selected for $\boldsymbol{X}_{\mathrm{J}}$. If the individuals in $\boldsymbol{X}_{\mathrm{J}}$ have not yet reached N, add $\boldsymbol{v}_i$ directly to $\boldsymbol{X}_{\mathrm{J}}$. Otherwise, randomly select an individual from $\boldsymbol{X}_{\mathrm{J}}$ and replace it with $\boldsymbol{v}_i$. The global elite population $\boldsymbol{X}_{\mathrm{Q}}$ is a collection of individuals with potential in the process of population iteration, and it has N individuals. The update method of $\boldsymbol{X}_{\mathrm{Q}}$ can be described as follows: after completing one iteration of the population, the optimal solution in the newly generated population is calculated. If it is better than the optimal solution obtained after the previous iteration, it will be considered as a candidate solution selected for $\boldsymbol{X}_{\mathrm{Q}}$. If the individuals in $\boldsymbol{X}_{\mathrm{Q}}$ have not yet reached N, add the optimal solution in the newly generated population directly to $\boldsymbol{X}_{\mathrm{Q}}$. Otherwise, randomly select an individual from $\boldsymbol{X}_{\mathrm{Q}}$ and replace it with the optimal solution in the newly generated population.

The implementation process of updating methods for the local elite population and global elite population is presented in Algorithm 2 and Algorithm 3, respectively.

Algorithm 2 The update mechanism of the local elite population.

Require: The population size N, the local elite population $\boldsymbol{X}_{\mathrm{J}}$, vector $\boldsymbol{v}_i$
1: Compute the number of individuals in $\boldsymbol{X}_{\mathrm{J}}$ and represent it as N_{J};
2: **if** $N_{\mathrm{J}} < N$ **then**
3: Add $\boldsymbol{v}_i$ into $\boldsymbol{X}_{\mathrm{J}}$;
4: **else**
5: Randomly select an individual from $\boldsymbol{X}_{\mathrm{J}}$ and replace it with $\boldsymbol{v}_i$;
6: **end if**
7: Output $\boldsymbol{X}_{\mathrm{J}}$.

Algorithm 3 The update mechanism of the global elite population.

Require: The population size N, the global elite population $\boldsymbol{X}_{\mathrm{Q}}$, the optimal solution in the newly generated population $\boldsymbol{x}_{\mathrm{n}}$
1: Compute the number of individuals in $\boldsymbol{X}_{\mathrm{Q}}$ and represent it as N_{Q};
2: **if** $N_{\mathrm{Q}} < N$ **then**
3: Add $\boldsymbol{x}_{\mathrm{n}}$ into $\boldsymbol{X}_{\mathrm{Q}}$;
4: **else**
5: Randomly select an individual from $\boldsymbol{X}_{\mathrm{Q}}$ and replace it with $\boldsymbol{x}_{\mathrm{n}}$;
6: **end if**
7: Output $\boldsymbol{X}_{\mathrm{Q}}$.

3.2. The Designed Search Strategies

The proposed EDJAYA has designed three search strategies based on the local elite population and global elite population created above, which can be expressed as follows:

- The first search strategy is exactly the same as JAYA, as shown in Equation (1).

- The second search strategy is based on the local elite population and global elite population. Let $\boldsymbol{x}_{\mathrm{s}}$ and $\boldsymbol{x}_{\mathrm{p}}$ represent an individual randomly selected from $\boldsymbol{X}_{\mathrm{J}}$ and $\boldsymbol{X}_{\mathrm{Q}}$, respectively. This search strategy can be represented by:

$$v_{i,j} = \begin{cases} x_{i,j} + \beta_4 \times (x_{\mathrm{s},j} - x_{i,j}) \times \theta, \text{if } f(\boldsymbol{x}_{\mathrm{s}}) \leq f(\boldsymbol{x}_i), \\ x_{i,j} - \beta_4 \times (x_{\mathrm{s},j} - x_{i,j}) \times \theta, \text{if } f(\boldsymbol{x}_{\mathrm{s}}) > f(\boldsymbol{x}_i), \end{cases} \tag{4}$$

$$v_{i,j} = \begin{cases} x_{i,j} + \beta_5 \times (x_{\mathrm{p},j} - x_{i,j}) \times \theta, \text{if } f(\boldsymbol{x}_{\mathrm{p}}) \leq f(\boldsymbol{x}_i), \\ x_{i,j} - \beta_5 \times (x_{\mathrm{p},j} - x_{i,j}) \times \theta, \text{if } f(\boldsymbol{x}_{\mathrm{p}}) > f(\boldsymbol{x}_i), \end{cases} \tag{5}$$

  where β_4 and β_5 are two random numbers between 0 and 1, $x_{\mathrm{s},j}$ is the jth element of $\boldsymbol{x}_{\mathrm{s}}$, $x_{\mathrm{p},j}$ is the jth element of $\boldsymbol{x}_{\mathrm{p}}$, and θ is an integer that can only take 0 and 1. Equations (4) and (5) are two search strategies designed from the perspectives of local elite population and global elite population, respectively. From Equations (4) and (5), it can be seen that the aim is to guide individuals in the elite population towards directions that are more likely to search for better solutions. In addition, θ can be seen as a crossover factor, which helps maintain population diversity.
- The third strategy is a mutation strategy. Let $\boldsymbol{m}(\boldsymbol{m} = [m_1, m_2, \ldots, m_{\mathrm{D}}])$ denote the mean position of the population. This search strategy can be expressed by:

$$v_{i,j} = m_j + \gamma_1 \times (x_{i,j} - m_j) - \gamma_2 \times (x_{\mathrm{p},j} - x_{\mathrm{s},j}), i = 1, 2, \ldots, N, j = 1, 2, \ldots, D, \tag{6}$$

  where γ_1 and γ_2 are two random numbers that follow a normal distribution, and m_j is the jth element of $\boldsymbol{m}$. $\boldsymbol{m}$ can be computed by

$$\boldsymbol{m} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{x}_i. \tag{7}$$

  According to Equation (6), the designed mutation strategy is based on the average position of the population, mainly because in one iteration, as long as the position of an individual changes, the average position of the population will also change. It can be said that the average position of the population has dynamic characteristics during the search process. In addition, the designed mutation strategy also introduces random numbers that follow a normal distribution, which has stronger volatility compared to random numbers that follow a uniform distribution between 0 and 1. The above considerations will enable the designed mutation strategy to demonstrate better advantages in escaping from local optima. It is very common to use normally distributed random numbers in the search strategies of meta-heuristic algorithms, such as in the Backtracking Search Algorithm [23], Power Method Algorithm [24], Generalized Normal Distribution Optimization [25], Adaptive Distributed Differential Evolution [26], and modified Grey Wolf Optimizer [27]. Random numbers following a normal distribution can achieve primarily local fine-grained search while occasionally making global jumps, thereby attaining a better balance between exploration and exploitation than a uniform distribution.

Compared with the existing variants of JAYA, the features of the EDJAYA proposed in this paper can be described as follows. EDJAYA incorporates two novel search strategies: one is based on the definition of two elite populations, and the other is a mutation strategy based on the mean position of the population. The former not only designs two search rules hierarchically according to the local elite population and the global elite population, but also introduces a 0–1 parameter for random crossover, thereby enhancing the global search capability of the designed search strategy; the latter, based on the population's mean position, introduces normally distributed random numbers as parameters to enhance the effect of mutation.

3.3. The Implementation of EDJAYA

Algorithm 4 shows the implementation of EDJAYA, which can be expressed as follows.

Algorithm 4 The implementation process of the proposed EDJAYA.

Require: The population size N, the number of decision variables D, the lower boundary of the decision variables $l_j (j \in [1, D])$, the upper boundary of decision variables $u_j (j \in [1, D])$, the maximum number of iterations T

1: Implement population initialization by Equation (3);
2: Initialize $\boldsymbol{X}_J$ and $\boldsymbol{X}_Q$ by $\boldsymbol{X}_J = \emptyset$ and $\boldsymbol{X}_Q = \emptyset$;
3: Evaluate the population and find the optimal solution $\boldsymbol{x}_b$;
4: Update $\boldsymbol{X}_J$ and $\boldsymbol{X}_Q$ by $\boldsymbol{X}_J = \boldsymbol{x}_b$ and $\boldsymbol{X}_Q = \boldsymbol{x}_b$;
5: **for** each $t \in [1, T]$ **do**
6: Find out $\boldsymbol{x}_b$ and $\boldsymbol{x}_w$;
7: **for** each $i \in [1, N]$ **do**
8: Generate a random number between 0 and 1 and represent it as η;
9: Get $\boldsymbol{x}_s$ and $\boldsymbol{x}_p$ from $\boldsymbol{X}_J$ and $\boldsymbol{X}_Q$, respectively;
10: **if** $\eta < \frac{1}{3}$ **then**
11: Get $\boldsymbol{v}_i$ by Equation (1);
12: **else if** $\eta < \frac{2}{3}$ **then**
13: **if** $f(\boldsymbol{x}_s) < f(\boldsymbol{x}_b)$ **then**
14: Get $\boldsymbol{v}_i$ by Equation (4);
15: **else**
16: Get $\boldsymbol{v}_i$ by Equation (5);
17: **end if**
18: **else**
19: Get $\boldsymbol{v}_i$ by Equations (6) and (7) ;
20: **end if**
21: Get $\boldsymbol{x}_i$ by Equation (2) and update $\boldsymbol{X}_J$ by Algorithm 2;
22: **end for**
23: Find out the optimal solution $\boldsymbol{x}_b$;
24: Update $\boldsymbol{X}_Q$ by Algorithm 3;
25: **end for**
26: Output $\boldsymbol{x}_b$.

- Line 1 is to initialize the population by Equation (3), whose computational complexity is $O(N \times D)$.
- Line 2 is to initialize $\boldsymbol{X}_J$ and $\boldsymbol{X}_Q$ by $\emptyset$, whose computational complexity is $O(N \times D)$.
- Line 3 is to get the best solution $\boldsymbol{x}_b$ by evaluating the population, whose computational complexity is $O(N \times D)$.
- Line 4 is to initialize $\boldsymbol{X}_J$ and $\boldsymbol{X}_Q$ by $\boldsymbol{x}_b$, whose computational complexity is $O(N \times D)$.
- Line 5 to line 25 are the main loops of the iterative process.
- Line 6 is to get $\boldsymbol{x}_b$ and $\boldsymbol{x}_w$, which will be used in line 11, whose computational complexity is $O(N)$.
- Line 7 to line 22 are the update processes of individuals in each iteration.
- Line 8 is to adjust the random number used to select the search strategy, whose computational complexity is $O(1)$.
- Line 9 is to randomly select an individual from both the local elite population and the global elite population, which will be used in line 13 to line 17, whose computational complexity is $O(1)$.
- Line 10 and line 11 are to execute the first search strategy when the condition is met, whose computational complexity is $O(N \times D)$.
- Line 12 to line 17 are to execute the second search strategy when the condition is met, whose computational complexity is $O(N \times D)$.

- Line 18 and line 19 are to execute the third search strategy when the condition is met, whose computational complexity is $O(N \times D)$.
- Line 21 is to update the position of individuals and the local elite population, whose computational complexity is $O(D)$.
- Line 23 and line 24 are to update x_b and the global elite population, whose computational complexity is $O(N)$.

According to the implementation process of EDJAYA presented in Algorithm 4, three search strategies are introduced. The execution probabilities of the three search strategies are the same and can be explained as follows. The three search strategies are designed from three different perspectives, the first of which is JAYA's benchmark strategy; the second of which is a design based on elite guided search strategy; and the third of which is the designed mutation strategy. In other words, these three search strategies guide individuals in generating the next generation from completely different perspectives. Therefore, without prior knowledge, the three search strategies give the same probability of being selected. It is worth mentioning that in the second search strategy, the priority of the search strategy based on local elites is higher than that of the search strategy based on global elites. This is because according to the design concept, the updates of local elite populations are more frequent than those of global elite populations, which can quickly accommodate the latest potential individuals found.

The characteristics of global exploration and local exploitation in EDJAYA can be described as follows. The first search strategy included in EDJAYA is based on the original JAYA search strategy, which inherently possesses both global exploration and local exploitation characteristics; the second search strategy included in EDJAYA is based on the elite population, guiding individuals in the population towards the positions of selected elite individuals, with a greater emphasis on global exploration; and the third search strategy included in EDJAYA is a mutation strategy based on the average position of the population, with a greater emphasis on local exploitation. That is to say, EDJAYA possesses the characteristics of both global exploration and local exploitation, which is beneficial for EDJAYA to have stronger global search capability when solving optimization problems.

4. Numerical Problems

4.1. Experimental Preparation

This section aims to evaluate EDJAYA's global search capability in numerical optimization problems. The numerical optimization problem used comes from the classic and challenging CEC 2022 test set with 12 test functions. The CEC 2022 test set includes multiple types of challenging numerical optimization functions, making it an ideal choice for evaluating the performance of optimization algorithms. At present, the CEC 2022 test set has been widely used in testing the performance of optimization algorithms [28–32]. The CEC 2022 test set can be described in detail as follows. The 12 test functions included in the CEC 2022 test suite can be expressed as: 1 unimodal function, 4 multimodal functions, 3 hybrid functions, and 4 composition functions. Overall, in the CEC 2022 test set, multimodal functions make up the vast majority and are suitable for testing the performance of optimization algorithms in solving complex optimization problems. The compared algorithms are JAYA, a comprehensive learning Jaya algorithm (CLJAYA) [33], an improved Jaya algorithm (IJAYA) [6], the Salp swarm algorithm (SSA) [34], Multi-Verse Optimizer (MVO) [35], Grey Wolf Optimizer (GWO) [20], and the Archimedes Optimization Agorithm (AOA) [36]. MVO, GWO, AOA, and SSA are four very popular metaheuristic algorithms with excellent global search capabilities. Their optimization performance has been widely recognized, and they have attracted attention since their introduction, with numerous improved versions continuously emerging. In addition, these four algorithms are frequently

used as benchmark algorithms to validate the performance of other algorithms [37–39]. For fair comparison, the population size and number of iterations for all algorithms are set to 30 and 3000, respectively. The other parameters of the compared algorithms are all from the corresponding original literature, and the proposed EDJAYA does not need any extral parameters. For the CEC 2022 testing suite, the experiment considers two dimensions, namely 10 dimensions and 20 dimensions. Each algorithm performs 30 independent repeated experiments on each dimension for each test function. Based on the obtained experimental results, further data analysis and processing will be conducted.

4.2. Experimental Results

Table 1 and Table 2 presents the experimental results of EDJAYA and the compared algorithms in solving the CEC 2022 test suite with 10 dimensions and 20 dimensions, respectively. In the two tables, MEAN and STD represent the average solution and standard deviation of the optimal solutions obtained from 30 independent repeated experiments, respectively. From Table 1, EDJAYA, MVO, AOA, and SSA can get the best MEAN in F_1, which is 3.00×10^2. However, EDJAYA has a lower STD than MVO, AOA, and SSA in F_1, which means that EDJAYA has a stronger stability than MVO, AOA, and SSA in solving F_1. In F_2, AOA can get the best MEAN (i.e., 4.04×10^2). EDJAYA also shows excellent global search ability in F_2, whose MEAN is 4.06×10^2, which is very close to that of AOA. EDJAYA and MVO outperform the other algorithms in solving F_3 and F_5 in terms of MEAN. EDJAYA can achieve the best MEAN in F_4, F_6, F_7, and F_8. AOA can obtain the best MEAN in F_9, which is 2.50×10^3. In terms of MEAN, JAYA, CLJAYA, MVO, SSA, and EDJAYA demonstrate results similar to AOA in F_9. In F_{10} and F_{11}, SSA is the best of all algorithms in terms of MEAN. JAYA, CLJAYA, MVO, SSA, and EDJAYA can achieve the best MEAN, i.e., 2.86×10^3. Based on the above analysis, EDJAYA is able to obtain the optimal MEAN or the parallel optimal MEAN in eight functions including F_1, F_3, F_4, and F_5, F_6, F_7, F_8, and F_{12}, which is the best among all algorithms. This indicates that the improvement strategy designed is highly effective in enhancing JAYA's global search capability. Looking at Table 2, EDJAYA, MVO, and SSA can get the best MEAN in F_1, which is 3.00×10^2. EDJAYA can obtain the best MEAN in F_2, F_3, F_4, and F_5, F_6, F_7, F_8, F_{10}, and F_{11}. In addition, AOA is superior to the other algorithms in F_9 and F_{12} in terms of MEAN. Overall, EDJAYA can achieve the optimal MEAN or share the optimal MEAN among 10 functions including F_1, F_2, F_3, F_4, and F_5, F_6, F_7, F_8, F_{10}, and F_{11}. This indicates that EDJAYA has extremely strong global search capability compared to the comparative algorithm. Comparing the experimental results in Tables 1 and 2, EDJAYA shows better optimization performance compared to the comparative algorithm as the dimensionality increased, indicating that the designed improvement strategy is less sensitive to dimensionality increase.

Tables 3 and 4 show the ranking results based on the Friedman test. The last row in these two tables displays the average ranking of each algorithm in 12 functions. According to Table 3, EDJAYA can obtain the best ranking in 8 functions, which are F_1, F_3, F_4, and F_5, F_6, F_7, F_8, and F_9. In F_2, AOA is the best of all algorithms and EDJAYA is the second best algorithm. SSA achieves the best ranking in F_{10} and F_{11}, while EDJAYA obtain the second best ranking in both functions. MVO has the best performance in F_{12}, while EDJAYA has the second best performance in this function. Overall, EDJAYA has demonstrated excellent optimization performance in every function, whether it is optimal or suboptimal. Based on the average ranking, all algorithms can be sorted from best to worst as EDJAYA, MVO, AOA, SSA, GWO, CLJAYA, JAYA, and IJAYA. Obviously, EDJAYA is the best among all algorithms. As shown in Table 4, SSA can obtain the best ranking in F_1. EDJAYA can achieve the best ranking in 9 functions, which are F_2, F_3, F_4, and F_5, F_6, F_7, F_9, F_{10}, and F_{11}. GWO and AOA outperform the other algorithms in F_8 and F_{12}, respectively. Note that,

in F_1, F_8, and F_{12}, although EDJAYA does not achieve the best ranking, it is the second best ranking. According to the average ranking results, all algorithms are ranked in descending order from best to worst: EDJAYA, MVO, AOA, SSA, GWO, CLJAYA, JAYA, and IJAYA. Obviously, EDJAYA is still the algorithm with the best ranking.

Table 1. The experimental results in optimizing the CEC 2022 test suite with 10 dimensions.

No.	Indicator	JAYA	CLJAYA	MVO	GWO	AOA	IJAYA	SSA	EDJAYA
F_1	MEAN	1.68×10^{3}	2.01×10^{3}	3.00×10^{2}	1.36×10^{3}	3.00×10^{2}	2.01×10^{4}	3.00×10^{2}	3.00×10^{2}
	STD	4.85×10^{2}	6.92×10^{2}	1.25×10^{-3}	1.33×10^{3}	9.24×10^{-3}	9.32×10^{3}	3.25×10^{-10}	2.79×10^{-14}
F_2	MEAN	4.10×10^{2}	4.19×10^{2}	4.12×10^{2}	4.20×10^{2}	4.04×10^{2}	5.21×10^{2}	4.08×10^{2}	4.06×10^{2}
	STD	2.71×10^{0}	2.70×10^{1}	2.06×10^{1}	2.03×10^{1}	4.18×10^{0}	1.26×10^{2}	1.25×10^{1}	3.15×10^{0}
F_3	MEAN	6.06×10^{2}	6.07×10^{2}	6.00×10^{2}	6.01×10^{2}	6.03×10^{2}	6.38×10^{2}	6.07×10^{2}	6.00×10^{2}
	STD	1.51×10^{0}	1.35×10^{0}	9.63×10^{-1}	6.08×10^{-1}	4.19×10^{0}	1.40×10^{1}	7.53×10^{0}	6.77×10^{-4}
F_4	MEAN	8.34×10^{2}	8.36×10^{2}	8.16×10^{2}	8.14×10^{2}	8.16×10^{2}	8.65×10^{2}	8.21×10^{2}	8.06×10^{2}
	STD	6.75×10^{0}	4.81×10^{0}	6.85×10^{0}	5.89×10^{0}	5.24×10^{0}	1.18×10^{1}	9.33×10^{0}	3.00×10^{0}
F_5	MEAN	9.13×10^{2}	9.17×10^{2}	9.00×10^{2}	9.10×10^{2}	9.09×10^{2}	2.33×10^{3}	9.02×10^{2}	9.00×10^{2}
	STD	6.29×10^{0}	5.06×10^{0}	2.15×10^{-4}	1.81×10^{1}	3.05×10^{1}	5.65×10^{2}	3.91×10^{0}	1.16×10^{-1}
F_6	MEAN	4.54×10^{5}	8.15×10^{5}	4.62×10^{3}	5.25×10^{3}	2.84×10^{3}	2.94×10^{7}	3.73×10^{3}	1.81×10^{3}
	STD	4.05×10^{5}	7.36×10^{5}	2.20×10^{3}	2.37×10^{3}	2.33×10^{3}	6.26×10^{7}	1.48×10^{3}	1.02×10^{1}
F_7	MEAN	2.04×10^{3}	2.04×10^{3}	2.04×10^{3}	2.03×10^{3}	2.03×10^{3}	2.09×10^{3}	2.03×10^{3}	2.01×10^{3}
	STD	5.88×10^{0}	6.02×10^{0}	4.15×10^{1}	1.05×10^{1}	1.02×10^{1}	2.90×10^{1}	1.08×10^{1}	7.89×10^{0}
F_8	MEAN	2.22×10^{3}	2.22×10^{3}	2.24×10^{3}	2.22×10^{3}	2.22×10^{3}	2.28×10^{3}	2.22×10^{3}	2.21×10^{3}
	STD	4.63×10^{0}	3.88×10^{0}	4.03×10^{1}	4.70×10^{0}	3.70×10^{0}	6.07×10^{1}	5.67×10^{0}	8.58×10^{0}
F_9	MEAN	2.53×10^{3}	2.53×10^{3}	2.53×10^{3}	2.56×10^{3}	2.50×10^{3}	2.57×10^{3}	2.53×10^{3}	2.53×10^{3}
	STD	2.08×10^{-1}	3.42×10^{0}	4.01×10^{-3}	2.79×10^{1}	5.75×10^{0}	5.21×10^{1}	2.50×10^{-3}	0.00×10^{0}
F_{10}	MEAN	2.53×10^{3}	2.51×10^{3}	2.55×10^{3}	2.56×10^{3}	2.57×10^{3}	2.75×10^{3}	2.50×10^{3}	2.53×10^{3}
	STD	5.72×10^{1}	3.56×10^{1}	6.99×10^{1}	6.63×10^{1}	6.11×10^{1}	5.09×10^{2}	7.70×10^{-2}	4.79×10^{1}
F_{11}	MEAN	2.67×10^{3}	2.69×10^{3}	2.72×10^{3}	2.84×10^{3}	2.74×10^{3}	3.25×10^{3}	2.61×10^{3}	2.69×10^{3}
	STD	1.31×10^{2}	1.60×10^{2}	1.63×10^{2}	1.88×10^{2}	1.53×10^{2}	5.33×10^{2}	5.70×10^{1}	1.35×10^{2}
F_{12}	MEAN	2.86×10^{3}	2.86×10^{3}	2.86×10^{3}	2.87×10^{3}	2.88×10^{3}	2.88×10^{3}	2.86×10^{3}	2.86×10^{3}
	STD	1.01×10^{0}	1.13×10^{0}	1.88×10^{0}	6.22×10^{0}	2.02×10^{1}	1.64×10^{1}	1.47×10^{0}	1.90×10^{0}

Table 2. The experimental results in optimizing the CEC 2022 test suite with 20 dimensions.

No.	Indicator	JAYA	CLJAYA	MVO	GWO	AOA	IJAYA	SSA	EDJAYA
F_1	MEAN	1.77×10^{4}	1.92×10^{4}	3.00×10^{2}	1.01×10^{4}	4.05×10^{2}	1.14×10^{5}	3.00×10^{2}	3.00×10^{2}
	STD	4.58×10^{3}	4.71×10^{3}	5.15×10^{-2}	3.58×10^{3}	1.59×10^{2}	2.39×10^{4}	2.69×10^{-9}	5.05×10^{-3}
F_2	MEAN	5.33×10^{2}	5.39×10^{2}	4.46×10^{2}	4.98×10^{2}	4.72×10^{2}	2.61×10^{3}	4.48×10^{2}	4.34×10^{2}
	STD	3.70×10^{1}	3.25×10^{1}	7.84×10^{0}	4.01×10^{1}	3.97×10^{1}	7.56×10^{2}	2.05×10^{1}	2.36×10^{1}
F_3	MEAN	6.18×10^{2}	6.18×10^{2}	6.06×10^{2}	6.04×10^{2}	6.18×10^{2}	6.78×10^{2}	6.24×10^{2}	6.01×10^{2}
	STD	3.70×10^{0}	4.17×10^{0}	5.19×10^{0}	3.06×10^{0}	1.34×10^{1}	1.77×10^{1}	1.13×10^{1}	6.20×10^{-1}
F_4	MEAN	9.34×10^{2}	9.31×10^{2}	8.52×10^{2}	8.51×10^{2}	8.68×10^{2}	1.04×10^{3}	8.55×10^{2}	8.42×10^{2}
	STD	1.36×10^{1}	1.07×10^{1}	1.64×10^{1}	2.64×10^{1}	1.23×10^{1}	2.75×10^{1}	1.71×10^{1}	1.83×10^{1}
F_5	MEAN	1.70×10^{3}	1.84×10^{3}	1.28×10^{3}	1.19×10^{3}	1.60×10^{3}	9.28×10^{3}	1.70×10^{3}	9.27×10^{2}
	STD	3.09×10^{2}	3.66×10^{2}	1.11×10^{3}	2.65×10^{2}	4.13×10^{2}	1.53×10^{3}	6.46×10^{2}	4.08×10^{1}
F_6	MEAN	4.06×10^{7}	3.71×10^{7}	1.11×10^{4}	1.50×10^{6}	6.53×10^{4}	6.82×10^{8}	7.11×10^{3}	2.60×10^{3}
	STD	1.32×10^{7}	8.65×10^{6}	8.73×10^{3}	4.44×10^{6}	2.44×10^{5}	8.54×10^{8}	6.46×10^{3}	9.44×10^{2}
F_7	MEAN	2.13×10^{3}	2.13×10^{3}	2.10×10^{3}	2.09×10^{3}	2.10×10^{3}	2.23×10^{3}	2.09×10^{3}	2.05×10^{3}
	STD	1.72×10^{1}	1.47×10^{1}	6.46×10^{1}	5.38×10^{1}	3.47×10^{1}	4.52×10^{1}	4.20×10^{1}	1.33×10^{1}
F_8	MEAN	2.24×10^{3}	2.24×10^{3}	2.28×10^{3}	2.24×10^{3}	2.24×10^{3}	2.40×10^{3}	2.27×10^{3}	2.23×10^{3}
	STD	4.74×10^{0}	4.44×10^{0}	7.54×10^{1}	3.55×10^{1}	3.92×10^{1}	1.58×10^{2}	6.12×10^{1}	3.08×10^{0}
F_9	MEAN	2.51×10^{3}	2.50×10^{3}	2.48×10^{3}	2.51×10^{3}	2.47×10^{3}	2.79×10^{3}	2.48×10^{3}	2.48×10^{3}
	STD	1.59×10^{1}	9.34×10^{0}	5.79×10^{-2}	2.50×10^{1}	4.15×10^{0}	1.25×10^{2}	1.13×10^{-1}	2.92×10^{-12}
F_{10}	MEAN	3.93×10^{3}	4.21×10^{3}	3.41×10^{3}	3.09×10^{3}	2.95×10^{3}	6.03×10^{3}	3.22×10^{3}	2.70×10^{3}
	STD	1.66×10^{3}	1.71×10^{3}	7.17×10^{2}	6.43×10^{2}	4.12×10^{2}	1.61×10^{3}	9.64×10^{2}	4.10×10^{2}
F_{11}	MEAN	4.02×10^{3}	3.95×10^{3}	2.94×10^{3}	3.47×10^{3}	3.04×10^{3}	7.79×10^{3}	2.92×10^{3}	2.90×10^{3}
	STD	3.61×10^{2}	4.48×10^{2}	2.25×10^{2}	3.50×10^{2}	1.95×10^{2}	2.06×10^{3}	1.24×10^{2}	9.28×10^{1}
F_{12}	MEAN	2.96×10^{3}	2.95×10^{3}	2.95×10^{3}	2.97×10^{3}	2.90×10^{3}	3.15×10^{3}	2.95×10^{3}	2.95×10^{3}
	STD	9.14×10^{0}	7.93×10^{0}	1.88×10^{1}	2.01×10^{1}	1.20×10^{1}	8.69×10^{1}	1.41×10^{1}	8.48×10^{0}

Table 3. The ranking of eight algorithms in optimizing the CEC 2022 test suite with 10 dimensions.

No.	JAYA	CLJAYA	MVO	GWO	AOA	IJAYA	SSA	EDJAYA
F_1	6.000	6.400	3.667	5.600	3.267	8.000	2.067	1.000
F_2	5.233	5.833	3.633	5.567	2.300	7.767	2.867	2.800
F_3	5.933	5.967	2.933	3.200	3.567	8.000	5.400	1.000
F_4	6.100	6.400	3.367	3.233	3.400	8.000	4.200	1.300
F_5	5.867	6.300	2.267	5.067	4.033	8.000	2.967	1.500
F_6	6.500	6.900	3.733	4.200	2.533	7.600	3.533	1.000
F_7	5.400	6.233	3.233	4.200	3.767	7.867	4.133	1.167
F_8	5.267	5.267	3.633	4.700	2.633	7.800	4.633	2.067
F_9	6.000	5.167	3.900	7.200	1.000	7.633	3.100	2.000
F_{10}	5.433	4.900	3.700	4.467	5.300	6.533	2.500	3.167
F_{11}	4.400	4.800	4.333	6.100	4.000	7.800	1.900	2.667
F_{12}	5.033	5.067	1.867	5.067	4.900	7.500	3.433	3.133
Avg.	5.597	5.769	3.356	4.883	3.392	7.708	3.394	1.900

Table 4. The ranking of eight algorithms in optimizing the CEC 2022 test suite with 20 dimensions.

No.	JAYA	CLJAYA	MVO	GWO	AOA	IJAYA	SSA	EDJAYA
F_1	6.233	6.600	3.000	5.167	4.000	8.000	1.000	2.000
F_2	6.133	6.300	2.600	4.900	3.400	8.000	2.733	1.933
F_3	5.233	5.400	3.100	2.433	4.800	8.000	5.867	1.167
F_4	6.600	6.367	3.000	2.600	4.267	8.000	3.033	2.133
F_5	5.367	5.700	2.033	3.333	4.967	8.000	4.867	1.733
F_6	6.733	6.733	3.567	4.267	2.833	7.533	2.733	1.600
F_7	5.600	5.800	3.833	3.400	4.100	7.900	3.733	1.633
F_8	5.300	5.067	5.100	2.300	2.567	7.633	5.200	2.833
F_9	6.167	5.700	3.200	6.133	1.033	8.000	3.800	1.967
F_{10}	5.133	5.467	4.067	3.667	3.400	7.833	3.733	2.700
F_{11}	6.500	6.267	2.700	5.200	3.500	7.967	2.600	1.267
F_{12}	5.300	4.533	3.767	6.000	1.133	8.000	4.367	2.900
Avg.	5.858	5.828	3.331	4.117	3.333	7.906	3.639	1.989

Tables 5 and 6 describe the Wilcoxon signed rank test results between EDJAYA and the compared algorithms, and the significance level is set to 0.05. In these two tables, the symbol + indicates that EDJAYA outperforms the compared algorithm; the symbol = represents that EDJAYA has similar performance to the comparison algorithm; and the symbol − represents that the performance of EDJAYA is inferior to the comparison algorithm. Observing Table 5, EDJAYA outperforms GWO in all functions. JAYA and IJAYA achieve the same performance as EDJAYA in F_{11} and F_{10}, respectively, but are inferior to EDJAYA in the remaining 11 functions. CLJAYA and EDJAYA have the same performance in F_{11} and F_{10}, but EDJAYA outperforms CLJAYA in the remaining 10 functions. MVO and AOA outperform EDJAYA in F_{12} and F_9, respectively, but EDJAYA outperforms MVO and AOA in 8 and 9 functions, respectively. Although SSA obtains the same performance as EDJAYA in 4 functions including F_2, F_{10}, F_{11}, and F_{12}, it cannot compete with EDJAYA in the remaining 8 functions. Overall, EDJAYA outperforms every algorithm in at least two-thirds of its functions, demonstrating excellent optimization performance. Looking at Table 6, EDJAYA outperforms JAYA, CLJAYA, and IJAYA in all functions. SSA can only beat EDJAYA in F_1, but it cannot compete with EDJAYA in the other 11 functions. MVO and EDJAYA have the same performance on three functions including F_2, F_5, and F_{12}, but EDJAYA outperforms MVO in the remaining nine functions. GWO has the same performance as EDJAYA in F_4 and F_8; however, EDJAYA demonstrates a performance advantage in the remaining 10 functions. AOA has better performance than EDJAYA in F_9 and F_{12}; AOA has the same performance with EDJAYA in F_8. AOA performs worse than EDJAYA in optimizing

the remaining nine functions. According to Table 6, EDJAYA demonstrates performance advantages over each comparative algorithm on at least three-quarters of the functions.

Table 5. The Wilcoxon signed rank test results of EDJAYA and the compared algorithms in optimizing the CEC 2022 test suite with 10 dimensions (the significance level is set at 0.05).

No.	EDJAYA vs.						
	JAYA	CLJAYA	MVO	GWO	AOA	IJAYA	SSA
F_1	+	+	+	+	+	+	+
F_2	+	+	=	+	=	+	=
F_3	+	+	+	+	+	+	+
F_4	+	+	+	+	+	+	+
F_5	+	+	=	+	+	+	+
F_6	+	+	+	+	+	+	+
F_7	+	+	+	+	+	+	+
F_8	+	+	+	+	=	+	+
F_9	+	+	+	+	−	+	+
F_{10}	+	=	=	+	+	=	=
F_{11}	=	=	+	+	+	+	=
F_{12}	+	+	−	+	+	+	=
+/=/−	11/1/0	10/2/0	8/3/1	12/0/0	9/2/1	11/1/0	8/4/0

Table 6. The Wilcoxon signed rank test results of EDJAYA and the compared algorithms in optimizing the CEC 2022 test suite with 20 dimensions (the significance level is set at 0.05).

No.	EDJAYA vs.						
	JAYA	CLJAYA	MVO	GWO	AOA	IJAYA	SSA
F_1	+	+	+	+	+	+	−
F_2	+	+	=	+	+	+	+
F_3	+	+	+	+	+	+	+
F_4	+	+	+	=	+	+	+
F_5	+	+	=	+	+	+	+
F_6	+	+	+	+	+	+	+
F_7	+	+	+	+	+	+	+
F_8	+	+	+	=	=	+	+
F_9	+	+	+	+	−	+	+
F_{10}	+	+	+	+	+	+	+
F_{11}	+	+	+	+	+	+	+
F_{12}	+	+	=	+	−	+	+
+/=/−	12/0/0	12/0/0	9/3/0	10/2/0	9/1/2	12/0/0	11/0/1

Figure 1 shows the convergence curves of EDJAYA and the comparison algorithms when solving the CEC 2022 test suite with 10 dimensions. As shown in Figure 1, compared to the contrasting algorithms, EDJAYA shows a very obvious convergence advantage in F_1, F_3, F_4, F_6, and F_7, being able to quickly search for better solutions. Although EDJAYA does not show a very obvious convergence advantage in F_2, F_5, F_8, and F_{12}, its convergence performance is still excellent. AOA achieves better convergence performance in F_9 compared to other algorithms. SSA, on the other hand, has the best convergence performance among all algorithms in F_{10} and F_{11}. Figure 2 presents the convergence curves of EDJAYA and the comparison algorithms when solving the CEC 2022 test suite with 20 dimensions. From Figure 2, EDJAYA has the best convergence performance in nine functions, which are F_1, F_2, F_3, F_4, F_5, F_6, F_7, F_{10}, and F_{11}. EDJAYA does not demonstrate a significant advantage in the convergence performance in F_8 and F_9. AOA exhibits the most outstanding convergence performance in F_{12}. So, based on the analysis above, EDJYA has the best convergence performance among all the algorithms. Based on the above convergence performance analysis, compared with the contrasting algorithms, EDJAYA's convergence advantage is

very significant. This is mainly attributed to EDJAYA's three search strategies. The first search strategy is the original JAYA search strategy, which can guide the population toward the location of the best-obtained individuals while avoiding moving toward the location of the worst-obtained individuals. The second search strategy is based on the elite population, which can guide the population in the direction with the potential to find the optimal individuals. The third search strategy is a mutation strategy based on the average position, which enhances the population's ability to escape local optima. The coordination of these three search strategies makes EDJAYA's convergence performance quite outstanding.

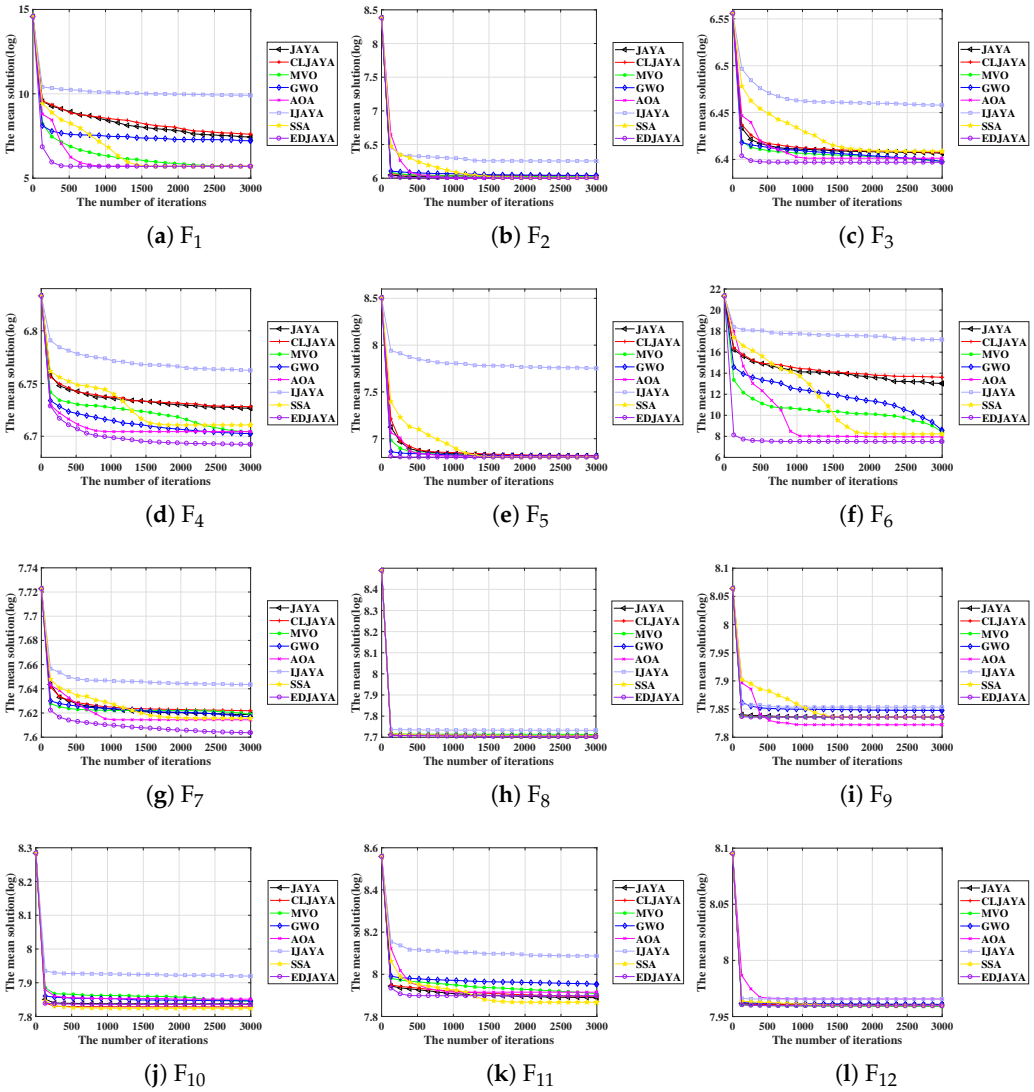
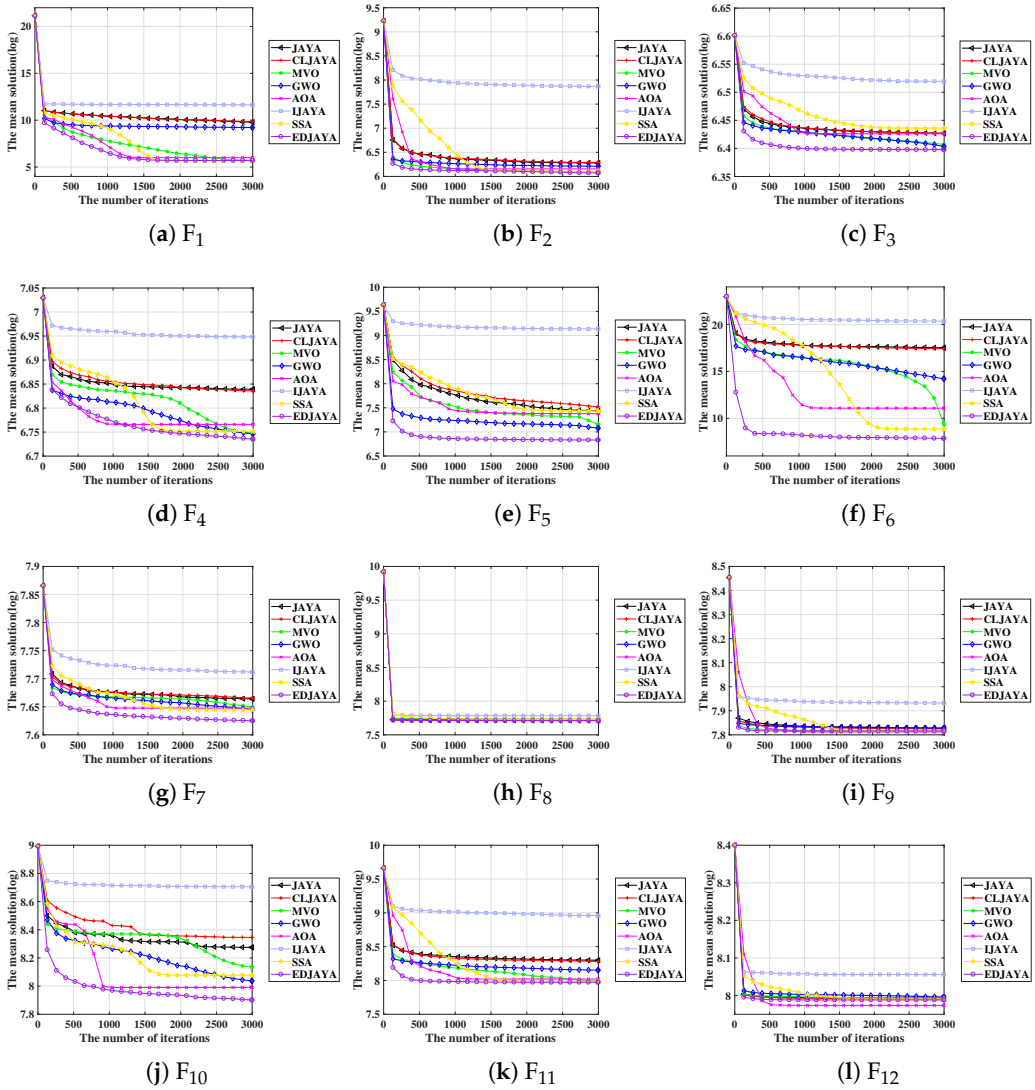


Figure 1. Convergence curves in optimizing the CEC 2022 test suite with 10 dimensions.

As mentioned earlier, the search mechanism of EDJAYA consists of three search strategies. To better analyze the contribution of each search strategy, the following comparative experiments were designed. First, the search mechanism composed only of the second and third search strategies is referred to as EDJAYA1; the search mechanism composed only of the first and third search strategies is referred to as EDJAYA2; and the search mechanism composed only of the first and second search strategies is referred to as EDJAYA3. In addition, EDJAYA1, EDJAYA2, and EDJAYA3 all include two search strategies, and these two strategies have the same execution probability. Other experimental parameters were set the same as in the previous experiments in this section. Table 7 presents the experimental results of EDJAYA1, EDJAYA2, EDJAYA3, and EDJAYA in solving the CEC 2022 test suite with 10 dimensions. From Table 7, EDJAYA1, EDJAYA3, and EDJAYA can get the best

MEAN in F_1, F_3, and F_5. EDJAYA can beat the other three algorithms in F_2, F_4, and F_{10} in terms of MEAN. EDJAYA1 and EDJAYA outperform EDJAYA2 and EDJAYA3 in F_6 in terms of MEAN. EDJAYA1 is the best of all algorithms in F_7 and F_{11} in terms of MEAN. EDJAYA3 and EDJAYA can obtain the best MEAN in F_8. The four algorithms can get the same MEAN in F_9 and F_{12}. Overall, based on the experimental results in Table 7, EDJAYA has a relatively significant performance advantage compared to the other three versions, while EDJAYA1 also demonstrates excellent competitiveness. Table 8 shows the experimental results of EDJAYA1, EDJAYA2, EDJAYA3, and EDJAYA in solving the CEC 2022 test suite with 20 dimensions. Looking at Table 8, EDJAYA1 and EDJAYA can achieve the best MEAN in F_1, F_2, and F_{11}. EDJAYA3 is the best of all algorithms in F_3 and F_5 in terms of MEAN. According to MEAN, EDJAYA1 has a performance advantage over the other three algorithms in F_4, F_6, F_7, and F_{10}. These four algorithms have the same MEAN in F_8. EDJAYA1, EDJAYA3, and EDJAYA have the best MEAN in F_9. EDJAYA3 and EDJAYA can obtain the best MEAN in F_{12}. According to Table 8, the overall performance of EDJAYA1 is superior to EDJAYA, while EDJAYA has advantages over EDJAYA2 and EDJAYA3. Based on the aforementioned comparative experiments, the two improvement strategies introduced in EDJAYA are very effective when combined.



(**a**) F_1 (**b**) F_2 (**c**) F_3

(**d**) F_4 (**e**) F_5 (**f**) F_6

(**g**) F_7 (**h**) F_8 (**i**) F_9

(**j**) F_{10} (**k**) F_{11} (**l**) F_{12}

Figure 2. Convergence curves of eight algorithms in optimizing the CEC 2022 test suite with 20 dimensions.

Table 7. Performance comparison of different EDJAYA versions in solving the CEC 2022 test suite with 10 dimensions.

No.	Indicator	EDJAYA1	EDJAYA2	EDJAYA3	EDJAYA
F_1	MEAN	3.00×10^{2}	3.70×10^{2}	3.00×10^{2}	3.00×10^{2}
	STD	2.11×10^{-14}	9.96×10^{1}	1.86×10^{0}	2.79×10^{-14}
F_2	MEAN	4.07×10^{2}	4.08×10^{2}	4.09×10^{2}	4.06×10^{2}
	STD	1.26×10^{1}	2.27×10^{0}	1.23×10^{1}	3.15×10^{0}
F_3	MEAN	6.00×10^{2}	6.02×10^{2}	6.00×10^{2}	6.00×10^{2}
	STD	1.33×10^{-1}	1.26×10^{0}	2.84×10^{-5}	6.77×10^{-4}
F_4	MEAN	8.08×10^{2}	8.10×10^{2}	8.13×10^{2}	8.06×10^{2}
	STD	3.48×10^{0}	6.60×10^{0}	3.29×10^{0}	3.00×10^{0}
F_5	MEAN	9.00×10^{2}	9.05×10^{2}	9.00×10^{2}	9.00×10^{2}
	STD	4.21×10^{-1}	4.40×10^{0}	8.56×10^{-2}	1.16×10^{-1}
F_6	MEAN	1.81×10^{3}	1.86×10^{3}	3.35×10^{3}	1.81×10^{3}
	STD	6.79×10^{0}	2.70×10^{1}	1.44×10^{3}	1.02×10^{1}
F_7	MEAN	2.00×10^{3}	2.02×10^{3}	2.01×10^{3}	2.01×10^{3}
	STD	7.56×10^{0}	9.88×10^{0}	9.00×10^{0}	7.89×10^{0}
F_8	MEAN	2.22×10^{3}	2.22×10^{3}	2.21×10^{3}	2.21×10^{3}
	STD	7.89×10^{0}	5.88×10^{0}	8.45×10^{0}	8.58×10^{0}
F_9	MEAN	2.53×10^{3}	2.53×10^{3}	2.53×10^{3}	2.53×10^{3}
	STD	0.00×10^{0}	1.38×10^{-4}	0.00×10^{0}	0.00×10^{0}
F_{10}	MEAN	2.54×10^{3}	2.54×10^{3}	2.54×10^{3}	2.53×10^{3}
	STD	5.40×10^{1}	5.41×10^{1}	5.29×10^{1}	4.79×10^{1}
F_{11}	MEAN	2.64×10^{3}	2.69×10^{3}	2.66×10^{3}	2.69×10^{3}
	STD	6.77×10^{1}	1.62×10^{2}	1.38×10^{2}	1.35×10^{2}
F_{12}	MEAN	2.86×10^{3}	2.86×10^{3}	2.86×10^{3}	2.86×10^{3}
	STD	1.10×10^{0}	2.24×10^{0}	1.46×10^{0}	1.90×10^{0}

Table 8. Performance comparison of different EDJAYA versions in solving the CEC 2022 test suite with 20 dimensions.

No.	Indicator	EDJAYA1	EDJAYA2	EDJAYA3	EDJAYA
F_1	MEAN	3.00×10^{2}	6.96×10^{3}	7.43×10^{3}	3.00×10^{2}
	STD	1.00×10^{-7}	3.08×10^{3}	3.00×10^{3}	5.05×10^{-3}
F_2	MEAN	4.34×10^{2}	5.58×10^{2}	4.50×10^{2}	4.34×10^{2}
	STD	2.37×10^{1}	5.49×10^{1}	1.96×10^{1}	2.36×10^{1}
F_3	MEAN	6.01×10^{2}	6.14×10^{2}	6.00×10^{2}	6.01×10^{2}
	STD	1.42×10^{0}	4.11×10^{0}	1.05×10^{-2}	6.20×10^{-1}
F_4	MEAN	8.40×10^{2}	8.66×10^{2}	8.91×10^{2}	8.42×10^{2}
	STD	1.37×10^{1}	2.19×10^{1}	1.59×10^{1}	1.83×10^{1}
F_5	MEAN	9.53×10^{2}	1.16×10^{3}	9.17×10^{2}	9.27×10^{2}
	STD	9.73×10^{1}	1.35×10^{2}	5.82×10^{1}	4.08×10^{1}
F_6	MEAN	2.33×10^{3}	4.67×10^{3}	1.02×10^{4}	2.60×10^{3}
	STD	8.54×10^{2}	3.47×10^{3}	8.39×10^{3}	9.44×10^{2}
F_7	MEAN	2.04×10^{3}	2.06×10^{3}	2.05×10^{3}	2.05×10^{3}
	STD	1.21×10^{1}	1.84×10^{1}	1.34×10^{1}	1.33×10^{1}
F_8	MEAN	2.23×10^{3}	2.23×10^{3}	2.23×10^{3}	2.23×10^{3}
	STD	2.66×10^{0}	2.79×10^{0}	3.13×10^{0}	3.08×10^{0}
F_9	MEAN	2.48×10^{3}	2.52×10^{3}	2.48×10^{3}	2.48×10^{3}
	STD	4.39×10^{-13}	3.26×10^{1}	3.95×10^{-5}	2.92×10^{-12}
F_{10}	MEAN	2.57×10^{3}	3.45×10^{3}	2.63×10^{3}	2.70×10^{3}
	STD	1.46×10^{2}	1.41×10^{3}	2.08×10^{2}	4.10×10^{2}
F_{11}	MEAN	2.90×10^{3}	3.84×10^{3}	2.92×10^{3}	2.90×10^{3}
	STD	9.10×10^{1}	2.88×10^{2}	1.06×10^{2}	9.28×10^{1}
F_{12}	MEAN	2.96×10^{3}	2.96×10^{3}	2.95×10^{3}	2.95×10^{3}
	STD	1.66×10^{1}	2.42×10^{1}	1.08×10^{1}	8.48×10^{0}

5. Experimental Results in Engineering Optimization Problems

This section evaluates the performance of EDJAYA in solving practical engineering problems by using it to address the actual photovoltaic (PV) model parameter estimation problem.

5.1. Model Creation

This section considers PV models including the single-diode model (SDM) and the double-diode model (DDM). The mathematical models of these two photovoltaic models are as follows:

- SDM. The mathematical model of SDM can be expressed as [40–42]:

$$I_{\mathrm{L}} = I_{\mathrm{ph}} - I_{\mathrm{sh}} - I_{\mathrm{d}}, \tag{8}$$

$$I_{\mathrm{sh}} = \frac{V_{\mathrm{L}} + R_{\mathrm{s}} \times I_{\mathrm{L}}}{R_{\mathrm{sh}}}, \tag{9}$$

$$I_{\mathrm{d}} = I_{\mathrm{sd}} \times \left[\exp\left(\frac{V_{\mathrm{L}} \times q + R_{\mathrm{s}} \times I_{\mathrm{L}} \times q}{n \times T \times k} \right) - 1 \right], \tag{10}$$

  where (R_{s}) is a series resistance, (R_{sh}) is a shunt resistance, (I_{ph}) is a photo-generated controlled current source, I_{d} is a current flowing through the diode, I_{sh} is a current flowing through R_{sh}, I_{L} is a output current, k is the Boltzmann constant $(k = 1.380649 \times 10^{-23} J/K)$, T is the current temperature in kelvin, I_{sd} is a reverse saturation current of the diode, n is the ideal factor, and q is the electron charge $(q = 1.602176634 \times 10^{-19} C)$. SDM needs to estimate five unknown parameters: I_{ph}, I_{sd}, R_{s}, R_{sh}, and n.
- DDM. The mathematical model of SDM can be expressed as [40–42]:

$$I_{\mathrm{L}} = I_{\mathrm{ph}} - I_{\mathrm{sh}} - I_{\mathrm{d1}} - I_{\mathrm{d2}}, \tag{11}$$

$$I_{\mathrm{sh}} = \frac{V_{\mathrm{L}} + R_{\mathrm{s}} \times I_{\mathrm{L}}}{R_{\mathrm{sh}}}, \tag{12}$$

$$I_{\mathrm{d1}} = I_{\mathrm{sd1}} \times \left[\exp\left(\frac{V_{\mathrm{L}} \times q + R_{\mathrm{s}} \times I_{\mathrm{L}} \times q}{n_1 \times T \times k} \right) - 1 \right], \tag{13}$$

$$I_{\mathrm{d2}} = I_{\mathrm{sd2}} \times \left[\exp\left(\frac{V_{\mathrm{L}} \times q + R_{\mathrm{s}} \times I_{\mathrm{L}} \times q}{n_2 \times T \times k} \right) - 1 \right], \tag{14}$$

  where I_{d1} is a current flowing through diode 1, I_{d2} is a current flowing through diode 2, n_1 is the ideal factor of diode 1, n_2 is the ideal factor of diode 2, I_{sd1} is a reverse saturation current of diode 1, and I_{sd2} is a reverse saturation current of diode 2. DDM needs to estimate seven unknown parameters: I_{ph}, I_{sd1}, I_{sd2}, R_{s}, R_{sh}, n_1, and n_2.

To obtain the unknown parameters of the two types of PV mentioned above, a common approach is to construct the root mean square error (RMSE) between experimental data and real data. The smaller the RMSE, the smaller the error. This essentially transforms the parameter estimation problem of the PV model into an optimization problem. The expression for RMSE can be written as:

$$\mathrm{Min}(\mathrm{RMSE}) = f(\boldsymbol{x}) = \sqrt{\frac{1}{N_{\mathrm{d}}} \sum_{i=0}^{N_{\mathrm{d}}} \left[I_{\mathrm{L},i}^{\mathrm{r}} - I_{\mathrm{L},i}^{\mathrm{e}} \right]^2}, \tag{15}$$

$$I_{\mathrm{L},i}^{\mathrm{e}} = h(\boldsymbol{x}, V_{\mathrm{L},i}^{\mathrm{r}}), \tag{16}$$

where $\boldsymbol{x}$ is the unknown parameters, N_{d} is the number of points in the real data, $I_{\mathrm{L},i}^{\mathrm{r}}$ is the real current at point i, $V_{\mathrm{L},i}^{\mathrm{r}}$ is the real voltage at point i, $I_{\mathrm{L},i}^{\mathrm{e}}$ is the estimated current at point i and $h(\cdot)$ is a function used to compute $I_{\mathrm{L},i}^{\mathrm{e}}$. $h(\cdot)$ can be obtained via Equations (8)–(14) .

The real data of SDM and DDM are obtained from [40,41]. The lower and upper boundaries of the unknown parameters are presented in Table 9 [40,41]. Further, looking at

Table 9, UB and LB represent the upper boundary and the lower boundary, respectively. The algorithm for comparison is the same as the algorithm used in the numerical experiments. The population size and maximum number of iterations for all algorithms are set to 30 and 5000, respectively. Each algorithm runs 30 independent trials for each PV model parameter estimation problem to get the results.

Table 9. The upper and lower boundaries of PV model parameters.

Parameter	SDM		DDM	
	UB	**LB**	**UB**	**LB**
I_{ph}(A)	1	0	1	0
I_d(A)	1	0	-	-
I_{d1}(A)	-	-	1	0
I_{d2}(A)	-	-	1	0
$R_s(\Omega)$	0.5	0	0.5	0
$R_{sh}(\Omega)$	100	0	100	0
n	2	1	-	-
n_1	-	-	2	1
n_2	-	-	2	1

5.2. Results Analysis and Discussion

Table 10 and Table 11 present the statical results of EDJAYA and the compared algorithms in SDM and DDM, respectively. In these two tables, MAX represents the worst solution obtained from 30 independent repeated experiments; MEAN represents the average solution obtained from 30 independent repeated experiments; MEDIAN represents the median solution obtained from 30 independent repeated experiments; MIN represents the optimal solution obtained from 30 independent repeated experiments; and STD represents the standard deviation obtained from 30 independent repeated experiments. From Table 10, EDJAYA outperforms the compared algorithms in MAX, MEAN, MEDIAN, and STD. It is worth mentioning that CLJAYA, IJAYA, and EDJAYA achieved the same solution in indicator MIN, which is 0.000986022. However, IJAYA and CLJAYA cannot compete with EDJAYA in the other four indicators. In addition, regarding the obtained MIN, the solutions obtained by MVO, GWO, and SSA are 0.001176245, 0.001025490, and 0.001091342, respectively. These three solutions are far superior to those of other algorithms. This indicates that MVO, GWO, and SSA are not applicable in solving the parameter estimation problem of SDM. Therefore, overall, EDJAYA is the best algorithm for extracting unknown parameters from SDM among all algorithms. Table 12 shows the best solutions of EDJAYA and the compared algorithms in SDM, which provides the estimated parameters corresponding to the obtained optimal RMSE. As presented in Table 11, EDJAYA outperforms the compared algorithms in all five considered metrics, namely MAX, MEAN, MEDIAN, MIN, and STD. This indicates that EDJAYA has advantages over comparative algorithms in terms of solving quality and stability. The minimum RMSE that EDJAYA can obtain is 0.0009824849. In addition, the minimum RMSE obtained by CLJAYA and IJAYA are 0.0009826558 and 0.0009826103, respectively. In other words, in terms of obtaining the optimal RMSE, CLJAYA and IJAYA are very close to EDJAYA, demonstrating competitiveness. The minimum RMSE obtained by JAYA and AOA are 0.0009837765 and 0.0009835687, respectively. In terms of the obtained optimal RMSE, although JAYA and AOA are still inferior to EDJAYA, IJAYA, and CLJAYA, they are superior to MVO, GWO, and SSA. Therefore, overall, EDJAYA is the best algorithm for extracting unknown parameters from DDM among all algorithms. Table 13 displays the best solutions of EDJAYA and the compared algorithms in DDM, which provides the estimated parameters corresponding to the obtained optimal RMSE.

Table 10. The statical results of EDJAYA and the compared algorithms in SDM.

Algorithm	MAX	MEAN	MEDIAN	MIN	STD
JAYA	0.001438660	0.001047166	0.000990175	0.000986050	1.39×10^{-4}
CLJAYA	0.000986265	0.000986058	0.000986027	0.000986022	5.97×10^{-8}
MVO	0.011130774	0.003496621	0.002400606	0.001176245	2.72×10^{-3}
GWO	0.038150446	0.005138962	0.002909240	0.001025490	8.46×10^{-3}
AOA	0.002457040	0.001186392	0.001055008	0.000987319	3.71×10^{-4}
IJAYA	0.000986022	0.000986022	0.000986022	0.000986022	2.58×10^{-11}
SSA	0.004983451	0.002121489	0.001832293	0.001091342	1.06×10^{-3}
EDJAYA	0.000986022	0.000986022	0.000986022	0.000986022	2.52×10^{-17}

Table 11. The statical results of EDJAYA and the compared algorithms in DDM.

Algorithm	MAX	MEAN	MEDIAN	MIN	STD
JAYA	0.0014411309	0.0010830361	0.0010003940	0.0009837765	1.56×10^{-4}
CLJAYA	0.0012054527	0.0009932941	0.0009857651	0.0009826558	4.02×10^{-5}
MVO	0.0043370908	0.0028668898	0.0028902365	0.0014312921	8.23×10^{-4}
GWO	0.0333924871	0.0037418207	0.0026379446	0.0010432468	5.79×10^{-3}
AOA	0.0024745096	0.0013205231	0.0010541523	0.0009835687	5.32×10^{-4}
IJAYA	0.0009988646	0.0009856402	0.0009854003	0.0009826103	3.01×10^{-6}
SSA	0.0064303396	0.0032188257	0.0028724531	0.0012304276	1.47×10^{-3}
EDJAYA	0.0009891230	0.0009836639	0.0009825131	0.0009824849	1.96×10^{-6}

Table 12. The best results of EDJAYA and the compared algorithms in SDM.

Algorithm	I_{ph} (A)	I_{sd} (A)	R_s (Ω)	R_{sh} (Ω)	n	RMSE
JAYA	0.760785897	0.000000323	0.036377269	53.645990877	1.481172603	0.000986050
CLJAYA	0.760775035	0.000000323	0.036376825	53.725198276	1.481191038	0.000986022
MVO	0.760156742	0.000000274	0.036827188	49.457482423	1.465043852	0.001176245
GWO	0.760421368	0.000000354	0.036093741	61.240807632	1.490481337	0.001025490
AOA	0.760692220	0.000000328	0.036314478	54.688021329	1.482788069	0.000987319
IJAYA	0.760775530	0.000000323	0.036377093	53.718522272	1.481183583	0.000986022
SSA	0.761176804	0.000000260	0.037161393	45.291000345	1.459602177	0.001091342
EDJAYA	0.760775531	0.000000323	0.036377093	53.718522534	1.481183589	0.000986022

Table 13. The best results of EDJAYA and the compared algorithms in DDM.

Algorithm	I_{ph} (A)	I_{sd1} (A)	R_s (Ω)	R_{sh} (Ω)	n_1	I_{sd2} (A)	n_2	RMSE
JAYA	0.7607735769	2.61×10^{-7}	0.0366416689	54.3389479504	1.4628946861	4.32×10^{-7}	2.0000000000	0.0009837765
CLJAYA	0.7607789648	2.45×10^{-7}	0.0366569823	55.1428094098	1.4576064536	5.96×10^{-7}	1.9999692718	0.0009826558
MVO	0.7600452473	9.38×10^{-7}	0.0355364985	98.0150776823	1.6781232913	4.01×10^{-8}	1.3552537585	0.0014312921
GWO	0.7605499568	6.82×10^{-8}	0.0367132708	64.1091143484	1.3735191767	8.26×10^{-7}	1.7195215984	0.0010432468
AOA	0.7607756487	2.32×10^{-7}	0.0366412501	55.3322507350	1.4542454445	5.16×10^{-7}	1.9188603412	0.0009835687
IJAYA	0.7607833809	2.41×10^{-7}	0.0366659280	55.1878317900	1.4565570290	6.24×10^{-7}	1.9999776440	0.0009826103
SSA	0.7603649752	2.69×10^{-7}	0.0353561966	75.5863336812	1.4739295380	7.45×10^{-7}	1.8725463034	0.0012304276
EDJAYA	0.7607810792	7.49×10^{-7}	0.0367404302	55.4854409107	2.0000000000	2.26×10^{-7}	1.4510167457	0.0009824849

Figure 3 and Figure 4 show the comparison of the convergence curves of EDJAYA and the comparative algorithm in SDM and DDM, respectively. From these two figures, the following conclusions can be drawn:

- The convergence performance of MVO, GWO, and SSA is far inferior to other algorithms.
- Although AOA has better convergence performance than MVO, GWO, and SSA, it is still inferior to JAYA, IJAYA, CLJAYA, and EDJAYA.
- Although JAYA can converge to a better solution faster than MVO, SSA, GWO, and AOA, it still cannot beat IJAYA, CLJAYA, and EDJAYA in terms of convergence quality.
- Although the convergence quality of CLJAYA and IJAYA is higher than that of JAYA, MVO, SSA, GWO, and AOA, they still cannot compete with EDJAYA.

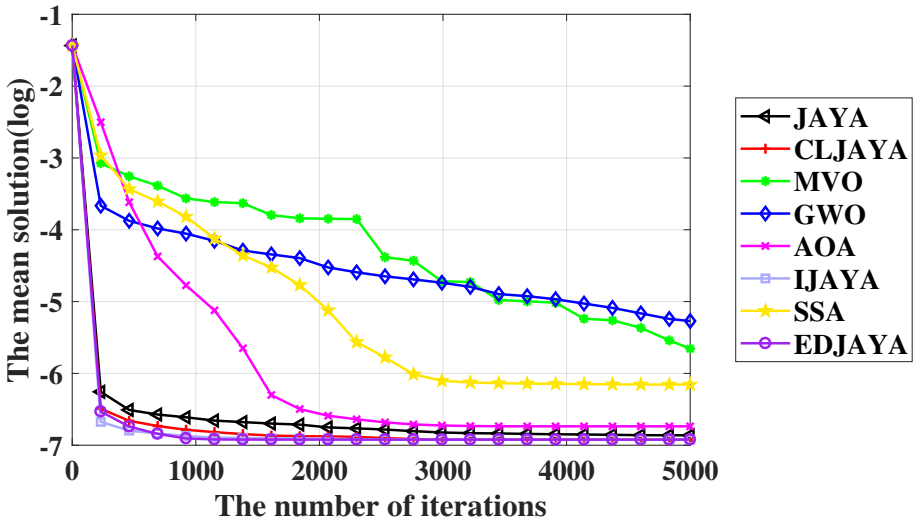


Figure 3. The convergence curves of EDJAYA and the compared algorithms in SDM.

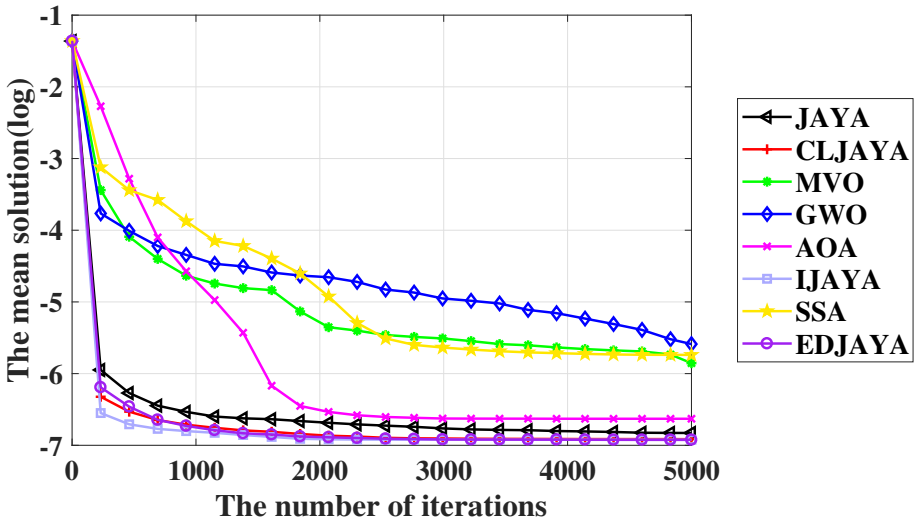


Figure 4. The convergence curves of EDJAYA and the compared algorithms in DDM.

Therefore, based on the above statistical analysis and convergence analysis, EDJAYA is the best among all algorithms. This indicates that the proposed improvement strategy is highly effective, enabling EDJAYA to have excellent global search capabilities and the ability to jump out of local optima, which demonstrates outstanding optimization performance in solving complex problems. In addition, in order to further evaluate the competitiveness of EDJAYA in the problem of photovoltaic model parameter extraction, Tables 14 and 15 present the optimal solutions obtained by EDJAYA and some recently proposed competitive algorithms in the photovoltaic model problem. From Table 14, EDJAYA and the other 10 comparative algorithms obtain approximately the same optimal solution when solving SDM. As can be seen from Table 15, when addressing the parameter estimation problem of DDM, the optimal solutions obtained by AEBSA, PDC-DE, LaPSO, Pro-DEAM, CDVSO, SIADE, and EDJAYA are almost identical and superior to those of RLRTLBO, RITLBO, SPGWO, and CNMSMA. In other words, compared with recently proposed competitive algorithms, EDJAYA still demonstrates excellent competitiveness in addressing the photovoltaic model parameter estimation problem.

Table 14. The best results of EDJAYA and the reported algorithms in SDM.

Year	Algorithm	I_{ph} (A)	I_{sd} (A)	R_s (Ω)	R_{sh} (Ω)	n	RMSE
2026	AEBSA [43]	0.760776	3.23×10^{-7}	0.036377	53.71853	1.481184	9.860219×10^{-4}
2026	PDC-DE [44]	0.76077553	0.00000032302081	0.03637709	53.71852235	1.48118359	$9.86021877891430 \times 10^{-4}$
2025	RLRTLBO [45]	0.7608	0.000000323	0.0364	53.7185	1.4812	0.000986022
2024	RITLBO [46]	0.7608	0.000000323	0.0364	53.7185	1.4812	$9.8602187789 \times 10^{-4}$
2023	SPGWO [47]	0.7607755	3.23×10^{-7}	0.0363771	53.7185219	1.4811836	9.8602×10^{-4}
2022	LaPSO [48]	0.76078	3.2302×10^{-7}	0.03638	53.71852	1.48118	9.86022×10^{-4}
2026	Pro-DEAM [49]	0.76077	2.323021×10^{-7}	0.036377	53.718522	1.481183	$9.8602187789 \times 10^{-4}$
2025	CDVSO [50]	0.7607755	0.000000323021	0.036377	53.71852	1.4811835	9.8602×10^{-4}
2021	CNMSMA [51]	0.760776	0.000000323017	0.036377	53.71821	1.481182	9.8602×10^{-4}
2026	SIADE [52]	0.76077553	0.00000032302082	0.03637709	53.71852613	1.48118359	$9.86021877891520 \times 10^{-4}$
This study	EDJAYA	0.760775531	0.000000323	0.036377093	53.718522534	1.481183589	0.000986022

Table 15. The best results of EDJAYA and the reported algorithms in DDM.

Year	Algorithm	I_{ph}(A)	I_{sd1} (A)	$R_s(\Omega)$	R_{sh} (Ω)	n_1	I_{sd2} (A)	n_2	RMSE
2026	AEBSA [43]	0.760781	2.26×10^{-7}	0.03674	55.48545	1.451017	7.49×10^{-7}	2	9.824849×10^{-4}
2026	PDC-DE [44]	0.76078108	2.2597447×10^{-7}	0.03674043	55.48543414	1.45101684	7.4934582×10^{-7}	2.00000000	$9.82484851784154 \times 10^{-4}$
2025	RLRTLBO [45]	0.7608	2.425×10^{-7}	0.0367	55.0916	1.4570	6.11×10^{-7}	1.9990	9.82695×10^{-4}
2024	RITLBO [46]	0.7608	2.308×10^{-7}	0.0367	55.3738	1.4528	7.079×10^{-7}	2.0000	$9.8249776623 \times 10^{-4}$
2023	SPGWO [47]	0.7607739	2.50×10^{-7}	0.0365767	55.6058032	1.4599666	4.85×10^{-7}	1.9495149	9.8422×10^{-4}
2022	LaPSO [48]	0.76078	7.4935×10^{-7}	0.03674	55.48545	2.00000	2.2597×10^{-7}	1.45102	9.82485×10^{-4}
2026	Pro-DEAM [49]	0.7608	7.4934×10^{-7}	0.03674	55.4854	1.9999	2.2597×10^{-7}	1.451017	9.82485×10^{-4}
2025	CDVSO [50]	0.76078	7.4935×10^{-7}	0.03674	55.4854	2	2.2597×10^{-7}	1.9999776440	9.82484×10^{-4}
2021	CNMSMA [51]	0.760781	2.25976×10^{-7}	0.036740	55.48545	1.451017	7.50681×10^{-7}	1.999999	9.8249×10^{-4}
2026	SIADE [52]	0.76078108	2.2597426×10^{-7}	0.03674043	55.48543365	1.45101676	$7.49347416 \times 10^{-7}$	2.00000	$9.82484851785167 \times 10^{-4}$
This study	EDJAYA	0.7607810792	7.49×10^{-7}	0.0367404302	55.4854409107	2.0000000000	2.26×10^{-7}	1.4510167457	0.0009824849

6. Conclusions

This paper proposes an improved JAYA, namely EDJAYA. Compared to JAYA, EDJAYA's improvement strategy introduces two elite populations to guide the search direction of the population. In addition, EDJAYA has designed a mutation strategy based on the average position of the population to enhance the algorithm's ability to escape from local optima. The improvement strategy designed in EDJAYA did not introduce new control parameters or complex calculations. The numerical experiments using EDJAYA to solve the CEC 2022 test set and the problem of identifying PV model parameters demonstrate the excellent global search capability of EDJAYA. Although EDJAYA demonstrates outstanding performance advantages, it still has some potential weaknesses. The designed search mechanism includes three search strategies, and how to achieve the best solution by adjusting the execution time of each search strategy when facing different optimization problems remains a challenge. According to the results of the ablation experiments conducted in the text, for solving simple problems, equal execution time can be allocated to each strategy. When the problems are more complex, the execution time of the second search strategy (the elitism-based search strategy) and the third search strategy (the mutation strategy based on the population's average position) can be appropriately increased, while the execution time of the first search strategy (the original JAYA search strategy) can be reduced.

Future research will focus on two directions: one is to verify the applicability of EDJAYA in more challenging large-scale optimization problems, such as the path planning problem for the Internet of Things data collection system based on large-scale unmanned aerial vehicles; the other is to design more efficient improved versions of JAYA through adaptive mechanisms, hybrid strategies, and the introduction of reinforcement learning.

Funding: The project was supported in part by the Special Fund of the Taishan Scholar Program under Grant tsqn202408187, and in part by the Shandong Provincial Natural Science Foundation under Grant ZR2025QC1548.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The author declares no conflicts of interest.

References

1. Rao, R. Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. *Int. J. Ind. Eng. Comput.* **2016**, *7*, 19–34. [CrossRef]
2. Venkata Rao, R.; Saroj, A. A self-adaptive multi-population based Jaya algorithm for engineering optimization. *Swarm Evol. Comput.* **2017**, *37*, 1–26. [CrossRef]
3. Warid, W.; Hizam, H.; Mariun, N.; Abdul-Wahab, N.I. Optimal Power Flow Using the Jaya Algorithm. *Energies* **2016**, *9*, 678. [CrossRef]
4. Gunduz, M.; Aslan, M. DJAYA: A discrete Jaya algorithm for solving traveling salesman problem. *Appl. Soft Comput.* **2021**, *105*, 107275. [CrossRef]
5. Mortazavi, A. A fuzzy reinforced Jaya algorithm for solving mathematical and structural optimization problems. *Soft Comput.* **2024**, *28*, 2181–2206. [CrossRef]
6. Cheng, Y.; Lyu, X.; Mao, S. Optimization design of brushless DC motor based on improved JAYA algorithm. *Sci. Rep.* **2024**, *14*, 5427. [CrossRef] [PubMed]
7. Das, H.; Naik, B.; Behera, H. A Jaya algorithm based wrapper method for optimal feature selection in supervised classification. *J. King Saud Univ. Comput. Inf. Sci.* **2022**, *34*, 3851–3863. [CrossRef]
8. Rao, R.; More, K.; Taler, J.; Ocłoń, P. Dimensional optimization of a micro-channel heat sink using Jaya algorithm. *Appl. Therm. Eng.* **2016**, *103*, 572–582. [CrossRef]
9. Huang, C.; Wang, L.; Yeung, R.S.C.; Zhang, Z.; Chung, H.S.H.; Bensoussan, A. A prediction model-guided Jaya algorithm for the PV system maximum power point tracking. *IEEE Trans. Sustain. Energy* **2017**, *9*, 45–55. [CrossRef]
10. Warid, W.; Hizam, H.; Mariun, N.; Wahab, N.I.A. A novel quasi-oppositional modified Jaya algorithm for multi-objective optimal power flow solution. *Appl. Soft Comput.* **2018**, *65*, 360–373. [CrossRef]
11. Wang, S.H.; Phillips, P.; Dong, Z.C.; Zhang, Y.D. Intelligent facial emotion recognition based on stationary wavelet entropy and Jaya algorithm. *Neurocomputing* **2018**, *272*, 668–676. [CrossRef]
12. Ding, Z.; Li, J.; Hao, H. Structural damage identification using improved Jaya algorithm based on sparse regularization and Bayesian inference. *Mech. Syst. Signal Process.* **2019**, *132*, 211–231. [CrossRef]
13. Chaudhuri, A.; Sahu, T.P. A hybrid feature selection method based on Binary Jaya algorithm for micro-array data classification. *Comput. Electr. Eng.* **2021**, *90*, 106963.
14. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
15. Trojovský, P. A new human-based metaheuristic algorithm for solving optimization problems based on preschool education. *Sci. Rep.* **2023**, *13*, 21472. [CrossRef] [PubMed]
16. Montazeri, Z.; Niknam, T.; Aghaei, J.; Malik, O.P.; Dehghani, M.; Dhiman, G. Golf Optimization Algorithm: A New Game-Based Metaheuristic Algorithm and Its Application to Energy Commitment Problem Considering Resilience. *Biomimetics* **2023**, *8*, 386. [CrossRef] [PubMed]
17. Lai, G.J.; Li, T.; Shi, B.J. RRT-Based Optimizer: A Novel Metaheuristic Algorithm Based on Rapidly-Exploring Random Trees Algorithm. *IEEE Access* **2025**, *13*, 42744–42776. [CrossRef]
18. Kumar, A.; Price, K.V.; Mohamed, A.W.; Hadi, A.A.; Suganthan, P.N. *Problem Definitions and Evaluation Criteria for the 2022 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization*; Technical Report; Nanyang Technological University: Singapore, 2021.
19. del Valle, Y.; Venayagamoorthy, G.K.; Mohagheghi, S.; Hernandez, J.C.; Harley, R.G. Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems. *IEEE Trans. Evol. Comput.* **2008**, *12*, 171–195. [CrossRef]
20. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
21. Cheng, J.; Wang, L. Cuckoo Search Algorithm with Neighborhood Attraction for Numerical Optimization. *IEEE Access* **2019**, *7*, 122261–122274. [CrossRef]
22. Zhang, Y. Elite archives-driven particle swarm optimization for large scale numerical optimization and its engineering applications. *Swarm Evol. Comput.* **2023**, *76*, 101212. [CrossRef]
23. Civicioglu, P. Backtracking search optimization algorithm for numerical optimization problems. *Appl. Math. Comput.* **2013**, *219*, 8121–8144. [CrossRef]
24. Zhang, H.; Wu, H.; Gong, Y.; Pan, X.; Zhong, Q. A novel transcendental metaphor metaheuristic algorithm based on power method. *Sci. Rep.* **2025**, *15*, 26997. [CrossRef] [PubMed]

25. Zhang, Y.; Jin, Z.; Mirjalili, S. Generalized normal distribution optimization and its applications in parameter extraction of photovoltaic models. *Energy Convers. Manag.* **2020**, *224*, 113301. [CrossRef]
26. Zhan, Z.H.; Wang, Z.J.; Jin, H.; Zhang, J. Adaptive Distributed Differential Evolution. *IEEE Trans. Cybern.* **2020**, *50*, 4633–4647. [CrossRef] [PubMed]
27. Sheng, L.; Wu, S.; Lv, Z. Modified Grey Wolf Optimizer and Application in Parameter Optimization of PI Controller. *Appl. Sci.* **2025**, *15*, 4530. [CrossRef]
28. Deng, B. Baboon optimization algorithm: A novel nature-inspired metaheuristic algorithm for optimization problems. *Ain Shams Eng. J.* **2026**, *17*, 104178. [CrossRef]
29. Roy, S.; Chakraborty, S.; Saha, A.K.; Sharma, M.; Bhattacharya (Halder), S.; Bhowmik, D. A Spiral and Enhanced Random Search Strategies based Slime Mould Algorithm for Global Optimization and Interplanetary Spacecraft Trajectory Problem. *Commun. Nonlinear Sci. Numer. Simul.* **2026**, *153*, 109508. [CrossRef]
30. Song, X.; Zhao, J. ACSBOA: Adaptive secretary bird optimization algorithm with cooperative mechanisms for multi real-world engineering problems. *Results Eng.* **2026**, *30*, 110968. [CrossRef]
31. Shi, H.; Kwok, N. Critical analysis of the animated oat optimization algorithm: Theoretical and empirical limitations. *Knowl.-Based Syst.* **2026**, *339*, 115461. [CrossRef]
32. Atha, R.; Rajan, A.; Mallick, S. An enhanced Equilibrium Optimizer for solving complex optimization problems. *Inf. Sci.* **2024**, *660*, 120077. [CrossRef]
33. Zhang, Y.; Ma, M.; Jin, Z. Comprehensive learning Jaya algorithm for parameter extraction of photovoltaic models. *Energy* **2020**, *211*, 118644. [CrossRef]
34. Mirjalili, S.; Gandomi, A.H.; Mirjalili, S.Z.; Saremi, S.; Faris, H.; Mirjalili, S.M. Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Adv. Eng. Softw.* **2017**, *114*, 163–191. [CrossRef]
35. Mirjalili, S.; Mirjalili, S.M.; Hatamlou, A. Multi-verse optimizer: A nature-inspired algorithm for global optimization. *Neural Comput. Appl.* **2016**, *27*, 495–513. [CrossRef]
36. Hashim, F.A.; Hussain, K.; Houssein, E.H.; Mabrouk, M.S.; Al-Atabany, W. Archimedes optimization algorithm: A new metaheuristic algorithm for solving optimization problems. *Appl. Intell.* **2021**, *51*, 1531–1551. [CrossRef]
37. Siswantoro, J.; Nehemiah Wijaya, J.; Riandaru Prasetyo, V.; Arwoko, H. A hybrid binary Grey Wolf and Whale optimization method for feature selection in classification tasks. *Expert Syst. Appl.* **2026**, *321*, 132277. [CrossRef]
38. S., G.; Mohapatra, P. Chaotic Aquila Optimization algorithm for solving global optimization and engineering problems. *Alex. Eng. J.* **2024**, *108*, 135–157. [CrossRef]
39. Elsisi, M. Optimal design of adaptive model predictive control based on improved GWO for autonomous vehicle considering system vision uncertainty. *Appl. Soft Comput.* **2024**, *158*, 111581. [CrossRef]
40. Yu, X.; Wu, X.; Luo, W. Parameter Identification of Photovoltaic Models by Hybrid Adaptive JAYA Algorithm. *Mathematics* **2022**, *10*, 183. [CrossRef]
41. Yu, K.; Liang, J.; Qu, B.; Chen, X.; Wang, H. Parameters identification of photovoltaic models using an improved JAYA optimization algorithm. *Energy Convers. Manag.* **2017**, *150*, 742–753. [CrossRef]
42. Choulli, I.; Elyaqouti, M.; hanafi Arjdal, E.; Ben hmamou, D.; Saadaoui, D.; Lidaighbi, S.; Elhammoudy, A.; Abazine, I.; El aidi idrissi, Y. DIWJAYA: JAYA driven by individual weights for enhanced photovoltaic model parameter estimation. *Energy Convers. Manag.* **2024**, *305*, 118258. [CrossRef]
43. Miao, F.; Ma, X.; Li, H. High-precision parameter estimation of photovoltaic models via a novel adaptive elite-biased backtracking search algorithm. *Measurement* **2026**, *258*, 119429. [CrossRef]
44. Sahu, D.; Gupta, S. Parameter estimation of solar photovoltaic models using fitness-based diversified cluster division and multi-mutation learned differential evolution. *Comput. Electr. Eng.* **2026**, *132*, 110995. [CrossRef]
45. Wang, H.; Yu, X.; Lu, Y. A reinforcement learning-based ranking teaching-learning-based optimization algorithm for parameters estimation of photovoltaic models. *Swarm Evol. Comput.* **2025**, *93*, 101844. [CrossRef]
46. Wang, H.; Yu, X. A ranking improved teaching-learning-based optimization algorithm for parameters identification of photovoltaic models. *Appl. Soft Comput.* **2024**, *167*, 112371. [CrossRef]
47. Yu, X.; Duan, Y.; Cai, Z. Sub-population improved grey wolf optimizer with Gaussian mutation and Lévy flight for parameters identification of photovoltaic models. *Expert Syst. Appl.* **2023**, *232*, 120827. [CrossRef]
48. Li, Y.; Yu, K.; Liang, J.; Yue, C.; Qiao, K. A landscape-aware particle swarm optimization for parameter identification of photovoltaic models. *Appl. Soft Comput.* **2022**, *131*, 109793. [CrossRef]
49. Saadaoui, D.; Elyaqouti, M.; Choulli, I.; Abazine, I.; Elhammoudy, A.; Lidighbi, S.; Ben Hmamou, D.; Ydir, B.; Arjdal, E.H.; Assalaou, K. Accurate parameter identification of solar cells and photovoltaic modules under real conditions using a new Differential Evolution approach. *Unconv. Resour.* **2026**, *9*, 100269. [CrossRef]
50. Bian, A.; Wang, Y.; Luo, Y. A novel chaotic differential variation snake optimization algorithm for parameters estimation of photovoltaic cell and module models. *Smart Power Energy Secur.* **2025**, *1*, 133–153. [CrossRef]

51. Liu, Y.; Heidari, A.A.; Ye, X.; Liang, G.; Chen, H.; He, C. Boosting slime mould algorithm for parameter identification of photovoltaic models. *Energy* **2021**, *234*, 121164. [CrossRef]
52. Lin, S.; Xu, H. Stagnation Improved Adaptive Differential Evolution for photovoltaic model parameter identification. *Eng. Appl. Artif. Intell.* **2026**, *163*, 112971. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.