mathematics

Article

Modified Sand Cat Swarm Optimization Algorithm for Solving Constrained Engineering Optimization Problems

Di Wu [1], Honghua Rao [2], Changsheng Wen [2], Heming Jia [2,*], Qingxin Liu [3] and Laith Abualigah [4,5,6,7]

1 School of Education and Music, Sanming University, Sanming 365004, China
2 School of Information Engineering, Sanming University, Sanming 365004, China
3 School of Computer Science and Technology, Hainan University, Haikou 570228, China
4 Hourani Center for Applied Scientific Research, Al-Ahliyya Amman University, Amman 19328, Jordan
5 Faculty of Information Technology, Middle East University, Amman 11831, Jordan
6 Faculty of Information Technology, Applied Science Private University, Amman 11931, Jordan
7 School of Computer Sciences, Universiti Sains Malaysia, Pulau Pinang 11800, Malaysia
* Correspondence: jiaheming@fjsmu.edu.cn

Abstract: The sand cat swarm optimization algorithm (SCSO) is a recently proposed metaheuristic optimization algorithm. It stimulates the hunting behavior of the sand cat, which attacks or searches for prey according to the sound frequency; each sand cat aims to catch better prey. Therefore, the sand cat will search for a better location to catch better prey. In the SCSO algorithm, each sand cat will gradually approach its prey, which makes the algorithm a strong exploitation ability. However, in the later stage of the SCSO algorithm, each sand cat is prone to fall into the local optimum, making it unable to find a better position. In order to improve the mobility of the sand cat and the exploration ability of the algorithm. In this paper, a modified sand cat swarm optimization (MSCSO) algorithm is proposed. The MSCSO algorithm adds a wandering strategy. When attacking or searching for prey, the sand cat will walk to find a better position. The MSCSO algorithm with a wandering strategy enhances the mobility of the sand cat and makes the algorithm have stronger global exploration ability. After that, the lens opposition-based learning strategy is added to enhance the global property of the algorithm so that the algorithm can converge faster. To evaluate the optimization effect of the MSCSO algorithm, we used 23 standard benchmark functions and CEC2014 benchmark functions to evaluate the optimization performance of the MSCSO algorithm. In the experiment, we analyzed the data statistics, convergence curve, Wilcoxon rank sum test, and box graph. Experiments show that the MSCSO algorithm with a walking strategy and a lens position-based learning strategy had a stronger exploration ability. Finally, the MSCSO algorithm was used to test seven engineering problems, which also verified the engineering practicability of the proposed algorithm.

Keywords: sand cat swarm optimization algorithm; sound frequency; exploitation ability; wandering strategy; exploration ability; lens opposition-based learning strategy; engineering problem

MSC: 49K35

check for updates

Citation: Wu, D.; Rao, H.; Wen, C.; Jia, H.; Liu, Q.; Abualigah, L. Modified Sand Cat Swarm Optimization Algorithm for Solving Constrained Engineering Optimization Problems. *Mathematics* **2022**, *10*, 4350. https://doi.org/10.3390/math10224350

Academic Editors: Zhiqiang Ma and Yunhe Wang

Received: 24 October 2022
Accepted: 17 November 2022
Published: 19 November 2022

1. Introduction

With the development of science and technology, there are many difficulties in describing and dealing with complex problems. Among these problems, the solution cannot be described in detail. And with the change in application scenarios, the solutions are always different. Meta-heuristic algorithms (MAs) are constructed based on intuition or experience. The optimization problem solved in an acceptable computing time or space provides a feasible solution. This feasible solution cannot be predicted in advance. Many engineering optimization problems often need the optimal solution in the complex and colossal search space. Because of the complexity, nonlinearity, constraints, and modeling difficulties of practical engineering problems, seeking efficient optimization algorithms has become an important research direction.

Inspired by human intelligence, the social behavior of biological groups, and laws of natural phenomena, scholars have invented many Mas to solve complex optimization problems. Mas have been used to solve various complex problems in the past few years. Mas are mainly classified into the following four categories: (1) swarm-based algorithms, (2) evolutionary-based algorithms, (3) physical-based algorithms, and (4) human-based algorithms. As shown in Figure 1. The first category mainly simulates the social behavior of population organisms such as birds, ants, and wolves. Particle Swarm Optimization (PSO) [1] simulates the foraging behavior of birds. Ant Colony Optimization (ACO) [2] simulates the behavior of ants in searching for food. Grey Wolf Optimization (GWO) [3] simulated wolves' hunting and leadership behavior. In addition, there are many excellent algorithms represented by swarm-based algorithms. For example, the Remora Optimization Algorithm (ROA) [4], Ant Lion Optimizer (ALO) [5], the Whale Optimization Algorithm (WOA) [6], and Moth Flame Optimization (MFO) [7]. The second category simulates the evolution process of organisms. The Genetic Algorithm (GA) [8] is inspired by Darwin's theory of evolution and simulates the evolution process of organisms. It is one of the most representative algorithms in natural evolutionary-based algorithms. Similar algorithms include Genetic Programming (GP) [9], Biogeography Based Optimizer (BBO) [10], the Virulence Optimization Algorithm (VOA) [11], Evolutionary Programming (EP) [12], and Differential Evolution (DE) [13]. The third category simulates the laws of physics. Simulated Annealing (SA) [14] simulated the principle of annealing. It starts from a higher initial temperature and then decreases with the decrease of temperature parameters. Algorithms based on this principle include the Sine Cosine Algorithm (SCA) [15], Multi-Verse Optimization (MVO) [16], the Gravitational Search Algorithm (GSA) [17], the Black Hole Algorithm (BH) [18], Thermal Exchange Optimization (TEO) [19], and Ray Optimization (RO) [20]. The last category simulates human behavior. Teaching Learning Based Optimization (TLBO) [21] simulates the teaching process of the class. The Group Teaching Optimization Algorithm (GTOA) [22] simulates group learning behavior and divides students in the class into groups for teaching. Similar algorithms include Harmony Search (HS) [23], Social Group Optimization (SGO) [24], and the Exchanged Market Algorithm (EMA) [25]. These algorithms are representative of MAs. They have a good effect on solving optimization problems.

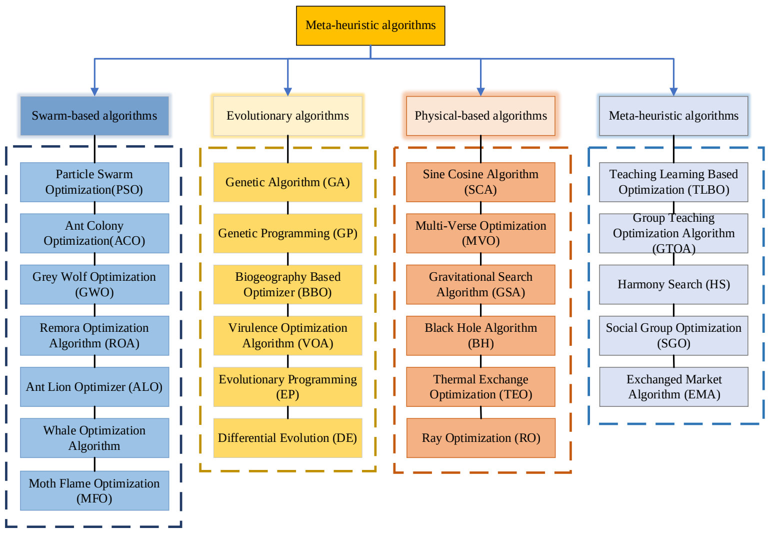

Meta-heuristic algorithms
Swarm-based algorithms
Evolutionary algorithms
Physical-based algorithms
Meta-heuristic algorithms
Particle Swarm Optimization(PSO)
Ant Colony Optimization(ACO)
Grey Wolf Optimization (GWO)
Remora Optimization Algorithm (ROA)
Ant Lion Optimizer (ALO)
Whale Optimization Algorithm
Moth Flame Optimization (MFO)
Genetic Algorithm (GA)
Genetic Programming (GP)
Biogeography Based Optimizer (BBO)
Virulence Optimization Algorithm (VOA)
Evolutionary Programming (EP)
Differential Evolution (DE)
Sine Cosine Algorithm (SCA)
Multi-Verse Optimization (MVO)
Gravitational Search Algorithm (GSA)
Black Hole Algorithm (BH)
Thermal Exchange Optimization (TEO)
Ray Optimization (RO)
Teaching Learning Based Optimization (TLBO)
Group Teaching Optimization Algorithm (GTOA)
Harmony Search (HS)
Social Group Optimization (SGO)
Exchanged Market Algorithm (EMA)

Figure 1. Classification of meta-heuristic optimization algorithms.

The Sand Cat Swarm Optimization (SCSO) [26] is a meta-heuristic optimization algorithm proposed in 2022. It is based on the idea of a swarm algorithm. The SCSO algorithm simulates the hunting behavior of the sand cat. Each sand cat is sensitive to sound frequency. According to the sound frequency of the prey, the sand cat will choose to attack or search for prey. In hunting, the sand cat will keep close to its prey. This will cause the sand cat to fall into the local optimum in the later stage, reducing the optimization performance of the algorithm. Li et al. proposed a sand cat swarm optimization algorithm based on stochastic variation and elite collaboration (SE-SCSO) [27]. The SE-SCSO algorithm adds a randomly changing elite cooperation strategy, which enables the algorithm to break away from the local extremum, and improves the algorithm's optimization seeking accuracy and convergence speed. Jovanovic et al. proposed feature selection by an improved sand cat swarm optimizer for intrusion detection [28]. They use extreme machine learning to test the improved sand cat optimization algorithm (HSCSO). The HSCSO algorithm has achieved good results in feature selection. The SCSO algorithm has insufficient convergence ability and quickly falls into the local optimum. This paper proposes a modified sand cat swarm optimization algorithm (MSCSO) for the above problem. Each sand cat has added different wandering strategies when hunting in order to find a better position. When attacking prey, the sand cat will walk according to the Levy flight walking (LFW) strategy. When searching for prey, the sand cat uses the triangle walking (TW) strategy to wander. Sand cats judge the distance between themselves and their prey and then use a Roulette Wheel selection algorithm to choose the direction of walking, and finally obtain a new position according to the trigonometric function calculation principle. Each sand cat searches for a better location through its walk strategy, which enhances the mobility of the algorithm and makes the MSCSO algorithm have a stronger global exploration ability. After that, the global exploration capability of the MSCSO algorithm is further enhanced by lens opposition-based learning (LOBL).

Through these two strategies, the global capability of the MSCSO algorithm is enhanced and the MSCSO algorithm can converge better. In the experimental part, we used 23 standard and CEC2014 benchmark functions to verify the optimization effect of the MSCSO algorithm, and the tables, convergence curves, box charts, and Wilcoxon rank sum tests of benchmark test functions were analyzed. Finally, in order to verify the engineering practicability of the MSCSO algorithm, we selected six engineering problems to test the optimization performance of the MSCSO algorithm. The results illustrate that the MSCSO algorithm also performs well in solving optimization problems.

The main contributions of this paper are as follows:

- The original SCSO algorithm is improved by the wandering strategy and the optimization performance of the original SCSO algorithm is enhanced.
- When searching for prey, the triangle walk (TW) strategy is added to expand the search scope of the SCSO algorithm and improve the global exploration ability of the algorithm.
- When attacking prey, the Levy flight walk (LFW) strategy is added to enable the sand cat to walk around the prey, so that the sand cat can find a better position and improve the optimization performance of the algorithm.
- Adding lens opposition-based learning (LOBL) to the MSCSO algorithm enhances the global exploration ability of the algorithm
- The MSCSO algorithm is tested and compared with the other eight algorithms, which proves that the MSCSO algorithm has a better optimization effect.

The structure of this article is as follows: The second part introduces the Restated Work. The third part briefly introduces the SCSO algorithm. The fourth part describes the improvement strategy of the MSCSO algorithm. The fifth and sixth parts give the experimental results of the MSCSO algorithm on benchmark functions and engineering problems. Finally, a summary is made in the seventh part.

2. Related Work

Meta-heuristic algorithms (MAs) are the improvement of the heuristic algorithm, which is the combination of a random algorithm and a local search algorithm. They mainly solve the optimal solution by simulating nature and human intelligence. The core of MAs is to balance the exploration and development capabilities of algorithms. MAs are widely used in various optimization fields because of their simplicity, easy implementation, and high accuracy of solution [29].

However, according to the NFL theorem [30], no MAs can solve all optimization problems. For this reason, improving the known MAs to better solve different optimization problems has become the research direction of many scholars. Many scholars have conceived many good methods [31]. For the defects of different algorithms, many scholars have proposed excellent solutions. For example, Mohammad H. Nadimi-Shahraki et al. proposed a multi-trial vector-based differential evolution algorithm (MTDE). The MTDE is distinguished by introducing an adaptive movement step designed based on a new multi-trial vector approach (MTV), which combines different search strategies in the form of trial vector producers (TVPs). The article uses the MTV method in the MTDE algorithm through three TVPs and verifies that the MTDE algorithm is more effective in dealing with different complex problems [32]. The Salp Swarm Algorithm (SSA) [33] simulated the foraging behavior of the salp swarm. Each salp will follow the best Salp for foraging. The algorithm has high convergence and coverage, and can approximate the optimal solution for the population. However, being too close to the optimal solution leads to the decline of the exploration ability of the SSA, which makes the algorithm difficult to converge in the later period. Hongliang Zhang et al. proposed the ensemble mutation-driven salp swarm algorithm with a restart mechanism (CMSRSSSA). The algorithm adds an ensemble mutation strategy. In this strategy, they adopt mutation schemes based on DE rand local mutation methods in Adaptive CoDE [34]. The exploration ability of the SSA was enhanced by strengthening the communication between different salps. Secondly, a restart mechanism is added, which enables individuals trapped in the local optimum to jump out of the local optimum to obtain a better position. These two mechanisms greatly improve the exploration ability of the SSA algorithm [35] The GWO algorithm lacks population diversity, and it is difficult to balance the exploitation and exploration, leading to premature convergence of the algorithm. Mohammad H. Nadimi Shahraki et al. proposed an improved Grey Wolf Optimizer (I-GWO). The I-GWO algorithm benefits from a new movement strategy named a dimension learning-based hanging (DLH) search strategy inherited from the individual hanging behavior of wolves in nature. The I-GWO algorithm uses the DLH strategy to build a domain for each gray wolf so that neighboring gray wolves can share information. This strategy balances the ability of the GWO algorithm exploration and exploitation and enhances the diversity of the population [36]. The idea of the Remora Optimization Algorithm (ROA) is that remora depends on powerful marine organisms to forage. Different organisms forage in different situations with novel content but lack autonomy. Zheng et al. proposed an autonomous foraging mechanism [37]. Remora not only depends on powerful marine organisms to find food but also can find food independently, which is more in line with biological characteristics and has achieved good optimization results. Mohammad H. Nadimi-Shahraki proposed a multi-trial vector-based moth-flame optimization (MTV-MFO) algorithm. In the algorithm, the MFO movement strategy is substituted by the multi-trial vector (MTV) approach to using a combination of different movement strategies, each of which is adjusted to accomplish a particular behavior. The MTV-MFO algorithm uses three different search strategies to improve the global search ability, maintain the balance between exploration and exploitation, and prevent the original MFO from premature convergence in the optimization process [38].

3. The Sand Cat Swarm Optimization Algorithm (SCSO)

3.1. Initialize Population

Each sand cat is a $1 \times dim$ array in the dim dimension optimization problem. It represents the solution to the problem, as shown in Figure 2. In a set of variable values (Pos_1, Pos_2, … , Pos_{dim}), each Pos must lie between the lower and upper boundary. In the initialization algorithm, an initialization matrix is created according to the size of the problem ($N \times dim$). In addition, the corresponding solution will be output in each iteration.

The current solution will be replaced if the next output value is better. If no better solution is found in the next iteration, the solution of this iteration will not be stored.

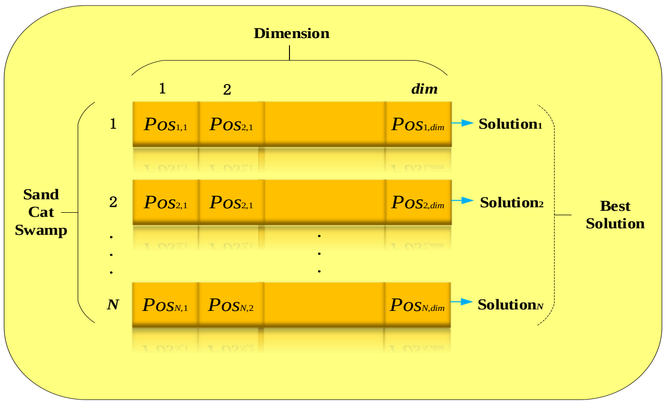


Figure 2. Population initialization diagram.

3.2. Search for Prey (Exploration Stage)

The position of each sand cat is expressed as Pos_i. The SCSO algorithm benefits from the hearing ability of sand cats in low-frequency detection. Each sand cat can sense the low frequency below 2 kHz. Therefore, in mathematical modeling, the sensitivity r_G is defined by Formula (1), so that the sensitivity range of the dune cat is 2 to 0 kHz. In addition, the parameter R is obtained according to Formula (2), and the algorithm exploration and exploitation ability is controlled.

$$r_G = S_M - \left(\frac{S_M \times t}{T}\right) \tag{1}$$

$$R = 2 \times r_G \times rand(0,1) - r_G \tag{2}$$

where S_M is 2, t is the current iteration number, and T is the maximum iteration number.

Each sand cat will randomly find a new location within the sensitivity range when searching for prey. This is more conducive to the exploration and exploitation of algorithms. To avoid falling into the local optimum, each sand cat's sensitivity range (r) is different. As shown in Formula (3).

$$r = r_G \times rand(0,1) \tag{3}$$

where r_G is used for the guidance parameter r.

Each sand cat will search for the position of prey according to the optimal candidate position (Pos_{bc}), current position ($Pos_c(t)$), and its sensitivity range (r). The specific Formula is shown in (4).

$$Pos(t+1) = r \times (Pos_{bc}(t) - rand(0,1) \times Pos_c(t)) \tag{4}$$

3.3. Attack Prey (Exploitation Stage)

The distance (Pos_{rnd}) between the sand cat and prey is shown by Formula (5) to simulate the process of the sand cat attacking prey. Assume that the sensitivity range of the sand cat is a circle, and the direction of movement uses the Roulette Wheel selection algorithm to select a random angle (α). Since the random angle selected is between 0° and 360°, its value is between −1 and 1. In this way, each sand cat can move in different circumferential directions in the search space, as shown in Figure 3. Then, the prey is attacked according to Formula (6). In this way, the dune cat can approach the hunting position faster.

$$pos_{rnd} = |rand(0,1) \times pos_b(t) - pos_c(t)| \tag{5}$$

$$pos(t+1) = Pos_b(t) - r \times Pos_{rnd} \times \cos(\alpha) \tag{6}$$

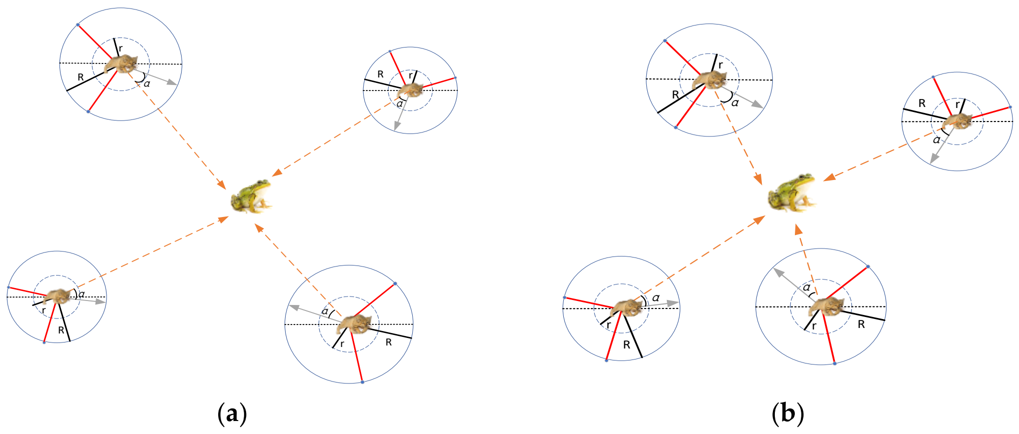


Figure 3. Location update mechanism of the SCSO algorithm. (**a**) The position of the sand cat group in *t* iteration; (**b**) The position of the sand cat group in *t* + 1 iteration.

3.4. Implementation of the SCSO Algorithm

The SCSO algorithm regulates the exploration and exploitation of the algorithm by controlling the adaptive parameters r_G and R. Formula (1), and shows that r_G decreases linearly from 2 to 0 during iteration. Therefore, the parameter R is a random value of [−4, 4]. The sand cat will attack prey when R is less than or equal to 1. Otherwise, the sand cat will search for prey, as shown in Formula (7).

$$Pos(t+1)\begin{cases} r \times (Pos_{bc}(t) - rand(0,1) \times Pos_c(t)) & |R| > 1;\ \text{exploration} \\ Pos_b(t) - Pos_{rnd} \times \cos(\alpha) \times r & |R| \le 1;\ \text{exploitation} \end{cases} \tag{7}$$

Formula (7) shows the location update of each sand cat during the exploration and exploitation stage. When $R \le 1$, the sand cat will attack its prey. Otherwise, the task of the sand cat is to find new prey in the global area. The pseudo-code is shown in Algorithm 1.

Algorithm 1. Sand Cat Swarm Optimization Algorithm Pseudo-Code

```
Initialize the population
Calculate the fitness function based on the objective function
Initialize the $r$, $r_G$, and $R$
While ($t \le$ maximum iteration)
   For each search agent
      Obtain a random angle based on the Roulette Wheel Selection ($0° \le \alpha \le 360°$)
      If (abs($R$) > 1)
         Update the search agent position based on Formula (4)
      Else
         Update the search agent position based on Formula (6)
   End
$T = t + 1$
End
```

4. The Modified Sand Cat Swarm Optimization Algorithm (MSCSO)

4.1. Wandering Strategy

4.1.1. Triangle Walk Strategy

The triangle walk strategy is for the sand cats to walk around as they approach their prey. First, obtain the distance L_1 between the sand cat and its prey. Then, obtain the step size range L_2 of the sand cat. Then, define the sand cat's walking direction (β) according to Formula (10). L_1 and L_2 are shown in Formulas (8) and (9). After that, calculate the distance

P between the position obtained by swimming and the prey by Formula (11). See Figure 4a for details. Finally, the position of the sand cat is obtained by Formula (12).

$$L_1 = pos_b(t) - pos_c(t) \tag{8}$$

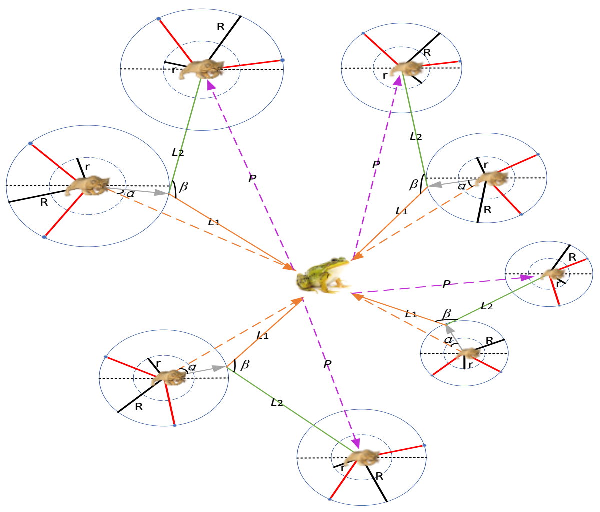
$$\overrightarrow{L_2} = rand() \times \overrightarrow{L_1} \tag{9}$$

$$\beta = 2 \times \pi \times rand() \tag{10}$$

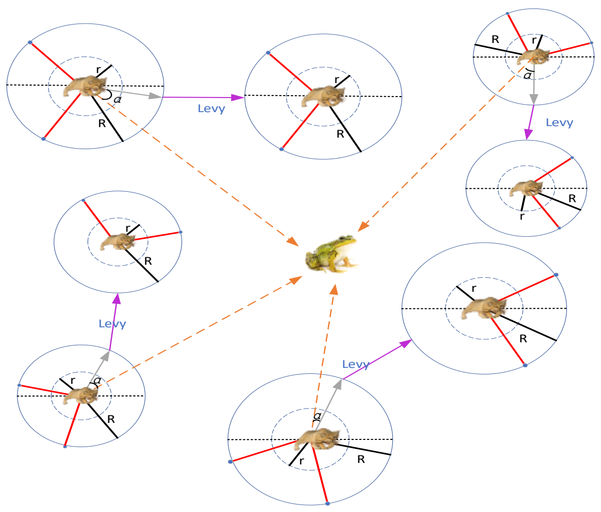
$$P = {L_1}^2 + {L_2}^2 - 2 \times L_1 \times L_2 \times \cos(\beta) \tag{11}$$

$$Pos_{new} = pos_b(t) + r \times P \tag{12}$$

Among them, Pos_{new} is the position obtained through the walking strategy.



(**a**) Triangle walk strategy



(**b**) Levy flight walk strategy

Figure 4. Schematic Diagram of the Wandering Strategy.

4.1.2. Levy Flight Walk Strategy

When attacking prey, the sand cat is very close to its prey. Levy flight is a very effective mathematical method for providing random factors. Levy flight can provide a walking method that conforms to Levy distribution. However, sometimes the step length of Levy's flight is too long. In order to better conform to the behavior of sand cats attacking prey, the constant $C = 0.35$ is multiplied in Levy flight. This allows the sand cat to walk as close to its prey as possible, as shown in Figure 4b. Levy's flight walking strategy is shown in Formula (13).

$$Pos_{new} = pos_b(t) + (pos_b(t) - pos_c(t)) \times C \times Levy \tag{13}$$

4.2. Lens Opposition-Based Learning

The main idea of lens opposition-based learning comes from the principle of convex lens imaging. The search range is expanded by generating a reverse position based on the current coordinates [39], which can be seen in Figure 5. In two-dimensional coordinates, the search range of the x-axis is (a, b) and the y-axis represents a convex lens. Suppose that the projection of object A on the x-axis is x and the height is h. Through lens imaging, the image on the other side is A^*, A^* is projected on the x-axis as x^*, and the height is h^*. Through the above analysis, we can calculate the reverse projection x^* of x.

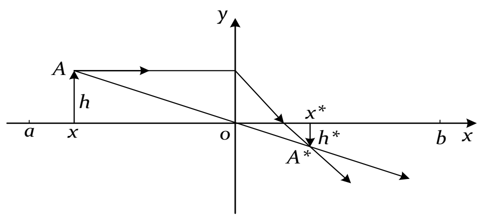


Figure 5. Lens opposition-based learning diagram.

In Figure 5, x takes o as the base point to obtain its corresponding reverse point x*, which can be obtained from the lens imaging principle.

$$\frac{(a+b)/2-x}{x^*-(a+b)/2}=\frac{h}{h^*} \tag{14}$$

Let $k = h/h^*$ to obtain the Formula (15) based on lens opposition-based learning.

$$x_j^*=\frac{a_j+b_j}{2}+\frac{a_j+b_j}{2k}-\frac{x_j}{k} \tag{15}$$

where x_j is the individual's position in the jth dimension and x_j* is the inverse solution of x_j. a_j and b_j are the maximum and minimum boundaries of dimension j in the search space.

4.3. Implementation of the MSCSO Algorithm

Initialization: In the initialization phase, initialize the population size N, dimension dim, iteration number T, and initialize the population as shown in Formula (16).

$$pos_{i,j}=(ub_j-lb_j)\times rand+lb_j \tag{16}$$

where ub_j is the upper bound of individual i in the j dimension, lb_j is the lower bound of individual i in the j dimension, and *rand* is a random number of [0, 1].

Search for prey: The hunting behavior of the sand cat is affected by the parameter R. When $|R|$ is greater than 1, it means that the prey is far away. At this time, the sand cat will search for prey according to the sensitivity range, as shown in Formula (4).

Triangle walk strategy (TW): While the sand cat is searching for its prey, it can not only search for its prey according to sensitivity range. Through the triangular walk strategy, the sand cat can choose the walking angle to randomly obtain new positions. The update is shown in Formula (12).

Attack prey: When the parameter $|R|$ is less than or equal to 1, this means that the sand cat is attacking its prey. Sand cats attack through the Roulette Wheel Selection algorithm by selecting angles and sensitivity range (r). As shown in Formula (6)

Levy flight walk strategy (LFW): In the stage of attacking prey, the sand cat is close to the optimal solution, which tends to lead to the population concentrating on the local optimal solution and being unable to find a better solution. Therefore, the levy flight can provide a walking method that conforms to levy distribution and make the sand cat more mobile. The specific implementation is shown in Formula (13).

Lens Opposition-Based Learning (LOBL): In order to further enhance the exploration ability of the MSCSO algorithm, lens opposition-based learning is added to further enhance the global exploration ability of the algorithm when updating the location. As shown in Formula (15).

Update population position: The location is updated by comparing fitness values. When the fitness value obtained from the update is better, the original individual will be replaced. On the contrary, the fitness value of the original individual is better than that of the newly acquired individual, and the original individual will be retained.

The pseudo-code of the MSCSO algorithm such as Algorithm 2.

Algorithm 2. The Modified Sand Cat Swarm Optimization Algorithm Pseudo-Code

```
Initialize the population according to Formula (16)
Calculate the fitness function based on the objective function
Initialize the $r$, $r_G$, and $R$
While ($t \leq$ maximum iteration)
  For each search agent
    Obtain a random angle based on the Roulette Wheel Selection ($0° \leq \alpha \leq 360°$).
    If (abs($R$) > 1)
      Update the search agent position based on Formula (4)
      Use Formula (12) for the triangle walk strategy to obtain a new position
    Else
      Update the search agent position based on Formula (6)
      Use Formula (13) to carry out the Levy flight walk strategy to obtain a new position
  End
Conduct the lens opposition-based learning strategy according to Formula (15)
$T = t + 1$
End
```

The flow chart of the MSCSO algorithm is shown in Figure 6:

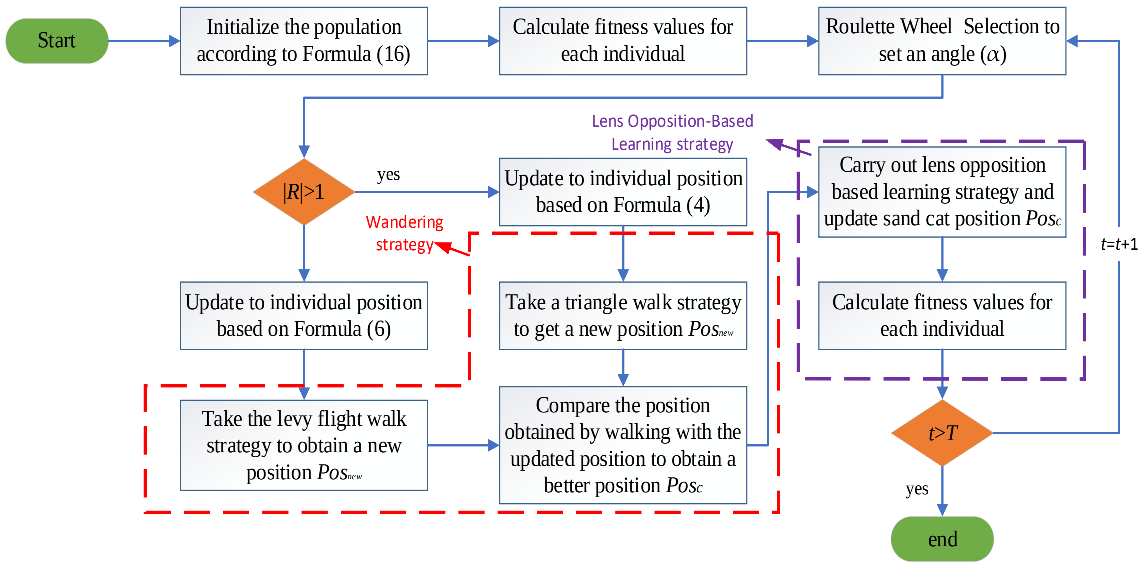

Start
Initialize the population according to Formula (16)
Calculate fitness values for each individual
Roulette Wheel Selection to set an angle (α)
Lens Opposition-Based Learning strategy
|R|>1
yes
Update to individual position based on Formula (4)
Carry out lens opposition based learning strategy and update sand cat position Pos_c
Wandering strategy
t=t+1
Update to individual position based on Formula (6)
Take a triangle walk strategy to get a new position Pos_new
Calculate fitness values for each individual
Take the levy flight walk strategy to obtain a new position Pos_new
Compare the position obtained by walking with the updated position to obtain a better position Pos_c
t>T
yes
end

Figure 6. Flowchart for the proposed MSCSO algorithm.

4.4. Complexity Analysis

The time complexity depends on the population size of the sand cat (N), the dimension of the given problem (dim), the number of iterations of the algorithm (T), and the evaluation cost required to solve the function (C). Therefore, the time complexity of the MSCSO algorithm is shown in Formula (17).

$$\begin{aligned} O(\text{MGTOA}) = & O(\text{define parameters}) + O(\text{population initialization}) \\ & + O(\text{dunction evaluation } \cos t) + O(\text{location update}) \end{aligned} \quad (17)$$

The specific definitions of each complexity are:

(1) The initialization parameter time is $O(1)$.
(2) Initialization of population position time $O(N \times dim)$.
(3) Time required for sand cats to prey $O(T \times N \times dim)$.
(4) Time required for position update of lens opposition-based learning $O(T \times N \times dim)$.
(5) The cost time of the calculation function includes the calculation time cost of the algorithm itself $O(T \times N \times C)$, the calculation time cost of walk strategy $O(T \times N \times C)$,

and the calculation time cost of lens opposition-based learning $O(T \times N \times C)$. Total $O(3 \times T \times N \times C)$.

Therefore, the time complexity of the MSCSO algorithm is.

$$O(\mathrm{MSCSO}) = O(1 + N \times dim + 3 \times T \times N \times C + 2 \times T \times N \times dim) \tag{18}$$

Because $1 << T \times N \times C$, $1 << T \times N \times dim$, $N \times dim << T \times N \times C$, and $N \times dim << T \times N \times dim$, Formula (18) can be simplified to Formula (19).

$$O(\mathrm{MSCSO}) \cong O(3 \times T \times N \times C + 2 \times T \times N \times dim) \tag{19}$$

5. Experimental Results and Discussion

All the experiments in this paper are completed on the computer with the 11th Gen Intel(R) Core(TM) i7-11700 processor with a primary frequency of 2.50 GHz, 16 GB memory, and an operating system of 64-bit Windows 11 using matlab2021a.

To verify the optimization effect of the MSCSO algorithm, this paper uses 23 standard benchmark functions and CEC2014 benchmark functions to verify the performance of the MSCSO algorithm. To better show the optimization effect, the MSCSO algorithm is compared with Sand Cat Swarm Optimization (SCSO) [26], the Arithmetic Optimization Algorithm (AOA) [40], Bald Eagle Search (BES) [41], the Whale Optimization Algorithm (WOA) [6], the Remora Optimization Algorithm (ROA) [4], the Sine Cosine Algorithm (SCA) [15], the Sooty Tern Optimization Algorithm (STOA) [42], and Genetic Algorithms (GA) [8]. The parameter settings of these algorithms are shown in Table 1.

Table 1. Parameter settings for the comparative algorithms.

Algorithm	Parameters	Value
GA	*Type*	Real coded
	Selection	Roulette Wheel (proportionate)
	Crossover	Whole aritharithmetic (Probability = 0.7)
	Mutation	Gaussian (Probability = 0.01)
STOA	*Sa*	[0, 2]
	b	1
SCA	α	2
ROA	*C*	0.1
WOA	Coefficient vectors $\vec{A}$	1
	Coefficient vectors $\vec{C}$	[−1, 1]
	Helical parameter *b*	0.75
	Helical parameter *l*	[−1, 1]
BES	α	[1.5, 2.0]
	r	[0, 1]
AOA	*MOP_Max*	1
	MOP_Min	0.2
	A	5
	Mu	0.499
SCSO	*SM*	2
	Roulette Wheel selection	[0, 360]
MSCSO	*C*	0.35
	SM	2
	β	[0, 2π]
	Roulette Wheel selection	[0, 360]

5.1. Experiments on the 23 Standard Benchmark Functions

The 23 standard benchmark functions are shown in Table 2. This benchmark contains seven unimodal, six multimodal, and ten fixed-dimension multimodal functions. Where F is the mathematical function, *dim* is the dimension, *Range* is the interval of the search space, and F_{min} is the optimal value the corresponding function can achieve, as seen in Figure 7. In this experiment, set the population size N = 30, the spatial dimension dim = 30/500, and the maximum number of iterations T = 500. The MSCSO algorithm and the eight comparison algorithms were independently run thirty times to obtain each algorithm's best fitness, average fitness, and standard deviation.

5.1.1. Result Statistics and Convergence Curve Analysis of the 23 Standard Reference Functions

Table 3 shows the statistical results of nine algorithms in the twenty-three standard benchmark functions. In the table, the MSCSO algorithm has obtained theoretical optimal values in F1–F4. The BES obtained the theoretical optimal value in F1. The ROA also has good convergence ability in F1. In the 30 dimensions, the AOA achieves the best in F2. In F5–F6, the MSCSO algorithm's best and mean are only next to the BES. In F7, the MSCSO algorithm obtains the optimal fitness value and is very stable. In F8, the MSCSO algorithm is inferior to the WOA, the ROA, and the BES algorithms, but superior to other comparison algorithms. The MSCSO algorithm achieves theoretical optimum in F9–F11. Compared with the SCSO algorithm, it has been dramatically improved. In F12–F13, the MSCSO algorithm did not obtain the best fitness value, but it achieved a better fitness value. In the 30 dimensions of F13, GA obtain better results, indicating that GA also have better optimization effects. The function of F14–F23 is relatively simple, and it is easy to find a better fitness value, but it also tests the optimization ability of the algorithm. The MSCSO algorithm obtains the optimal fitness value in the combination function's optimal fitness. The above analysis proves that the SCSO has a better optimization effect in the MSCSO algorithm with the TW, LFW, and LOBL strategies.

Table 2. Details of the 23 benchmark functions.

Type	F	dim	Range	F_{min}
Unimodal benchmark functions	$F_1(x) = \sum_{i=1}^{n} x_i^2$	30/500	$[-100, 100]$	0
	$F_2(x) = \sum_{i=1}^{n}\lvert x_i\rvert + \prod_{i=1}^{n}\lvert x_i\rvert$	30/500	$[-10, 10]$	0
	$F_3(x) = \sum_{i=1}^{n} (\sum_{j-1}^{i} x_j)^2$	30/500	$[-100, 100]$	0
	$F_4(x) = \max\{\lvert x_i\rvert, 1 \le i \le n\}$	30/500	$[-100, 100]$	0
	$F_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30/500	$[-30, 30]$	0
	$F_6(x) = \sum_{i=1}^{n} (x_i + 5)^2$	30/500	$[-100, 100]$	0
	$F_7(x) = \sum_{i=1}^{n} i \times x_i^4 + random[0, 1)$	30/500	$[-1.28, 1.28]$	0
Multimodal benchmark functions	$F_8(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i\rvert})$	30/500	$[-500, 500]$	$-418.9829 \times dim$
	$F_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30/500	$[-5.12, 5.12]$	0
	$F_{10}(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e)$	30/500	$[-32, 32]$	0
	$F_{11}(x) = \frac{1}{400}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$	30/500	$[-600, 600]$	0
	$F_{12}(x) = \frac{\pi}{n}\left\{10\sin(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\right\}$ $+\sum_{i=1}^{n} u(x_i, 10, 100, 4), \text{where } y_i = 1 + \frac{x_i+1}{4},$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	30/500	$[-50, 50]$	0
	$F_{13}(x) = 0.1(\sin^2(3\pi x_1) + \sum_{i=1}^{n} (x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)]$ $+(x_n - 1)^2[1 + \sin^2(2\pi x_n)]) + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30/500	$[-50, 50]$	0
Fixed-dimension multimodal benchmark functions	$F_{14}(x) = (\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j+\sum_{i=1}^{2} (x_i - a_{ij})^6})^{-1}$	2	$[-65, 65]$	1
	$F_{15}(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	4	$[-5, 5]$	0.00030
	$F_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1 x_2 - 4x_2^2 + x_2^4$	2	$[-5, 5]$	-1.0316
	$F_{17}(x) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos x_1 + 10$	2	$[-5, 5]$	0.398
	$F_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1 x_2 + 3_2^2)]$ $\times[30 + (2x_1 - 3x_2)^2 \times (18 - 32x_2 + 12x_1^2 + 48x_2 - 36x_1 x_2 + 27x_2^2)]$	5	$[-2, 2]$	3
	$F_{19}(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2)$	3	$[-1, 2]$	-3.86
	$F_{20}(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2)$	6	$[0, 1]$	-3.32
	$F_{21}(x) = -\sum_{i=1}^{5} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0, 10]$	-10.1532
	$F_{22}(x) = -\sum_{i=1}^{7} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0, 10]$	-10.4028
	$F_{23}(x) = -\sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0, 10]$	-10.5363

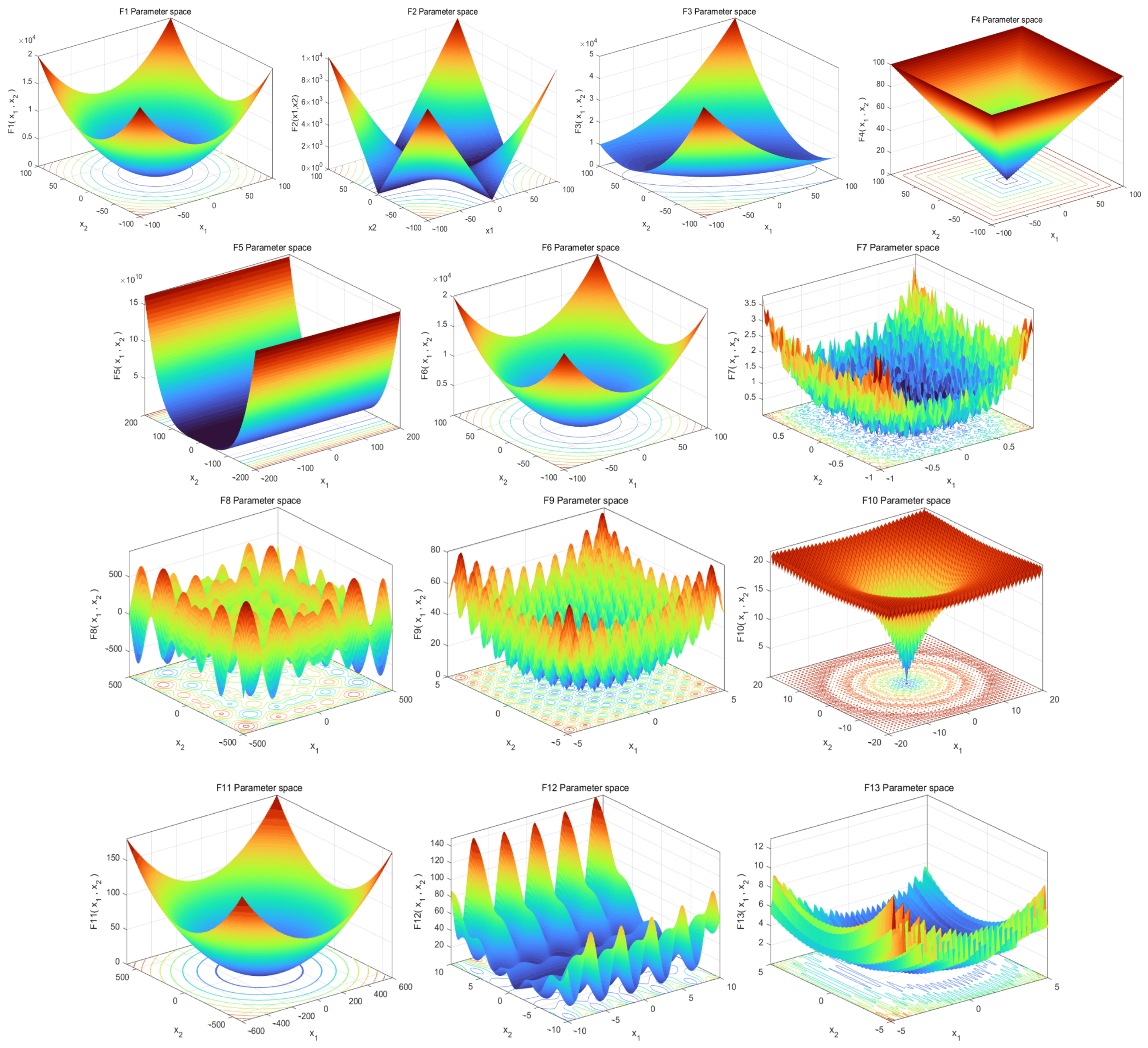


Figure 7. Schematic diagram of the 23 standard reference functions.

Table 3. Statistical results of the 23 standard reference functions (we bold the data with good algorithm).

F	*dim*	Metric	MSCSO	SCSO	AOA	BES	WOA	ROA	SCA	STOA	GA
F1	30	min	**0**	8.42×10^{-125}	3.07×10^{-169}	**0**	8.8×10^{-91}	**0**	1.46×10^{-2}	4.12×10^{-9}	6.97×10^{-3}
		mean	**0**	3.70×10^{-111}	9.15×10^{-20}	**0**	2.43×10^{-73}	1.43×10^{-322}	9.48	6.54×10^{-7}	2.29×10^{-2}
		std	**0**	2.01×10^{-110}	5.01×10^{-19}	**0**	1.01×10^{-72}	**0**	14.4	1.03×10^{-6}	7.21×10^{-3}
	500	min	**0**	1.64×10^{-110}	5.6×10^{-1}	**0**	6.15×10^{-83}	**0**	9.67×10^{4}	1.31	68.7
		mean	**0**	4.64×10^{-96}	6.49×10^{-1}	**0**	9.16×10^{-69}	1.16×10^{-315}	2×10^{5}	9.46	72
		std	**0**	2.54×10^{-95}	4.23×10^{-2}	**0**	3.76×10^{-68}	**0**	6.07×10^{4}	10.3	2.62
F2	30	min	**0**	3.02×10^{-65}	**0**	4.66×10^{-229}	9.23×10^{-57}	1.46×10^{-180}	2.06×10^{-4}	5.75×10^{-7}	3.60×10^{-1}
		mean	**0**	1.28×10^{-58}	**0**	4.45×10^{-159}	1.25×10^{-49}	1.81×10^{-156}	3.41×10^{-2}	7.79×10^{-6}	4.97×10^{-1}
		std	**0**	6.77×10^{-58}	**0**	2.44×10^{-158}	6.69×10^{-49}	9.86×10^{-156}	9.25×10^{-2}	8.89×10^{-6}	6.27×10^{-2}
	500	min	**0**	4.89×10^{-56}	5.74×10^{-10}	3.89×10^{-220}	1×10^{-54}	5.87×10^{-180}	31.9	1.07×10^{-2}	1.33×10^{2}
		mean	**0**	7.07×10^{-50}	1.44×10^{-3}	2.40×10^{-160}	1.24×10^{-48}	1.82×10^{-161}	1.21×10^{2}	1.26×10^{-1}	1.4×10^{2}
		std	**0**	3.81×10^{-49}	2.22×10^{-3}	1.32×10^{-159}	6.21×10^{-48}	9.02×10^{-161}	69.6	1.04×10^{-1}	2.59
F3	30	min	**0**	6.38×10^{-112}	8.44×10^{-116}	**0**	8.72×10^{3}	1.44×10^{-320}	1.76×10^{3}	2.22×10^{-3}	6.06×10^{3}
		mean	**0**	2.52×10^{-98}	4.35×10^{-3}	3.67×10^{-23}	4.15×10^{4}	6.51×10^{-280}	9.64×10^{3}	5.12×10^{-1}	2.28×10^{4}
		std	**0**	1.31×10^{-97}	9.38×10^{-3}	2.01×10^{-22}	1.32×10^{4}	**0**	6.10×10^{3}	2.26	8.15×10^{3}
	500	min	**0**	1.53×10^{-98}	15.5	**0**	1.23×10^{7}	4.22×10^{-297}	4.34×10^{6}	2.31×10^{5}	4.66×10^{5}
		mean	**0**	6.14×10^{-82}	34.4	15.1	2.93×10^{7}	1.51×10^{-260}	6.85×10^{6}	5.39×10^{5}	7.25×10^{5}
		std	**0**	3.36×10^{-81}	17.8	82.6	1.26×10^{7}	**0**	1.53×10^{6}	1.93×10^{5}	1.28×10^{5}
F4	30	min	**0**	1.99×10^{-54}	1.31×10^{-48}	8.59×10^{-232}	4.54	2.98×10^{-176}	20.9	1.39×10^{-2}	2.23×10^{-1}
		mean	**0**	5.92×10^{-49}	2.53×10^{-2}	1.96×10^{-158}	51.6	8.49×10^{-157}	32.1	5.94×10^{-2}	2.91×10^{-1}
		std	**0**	3.14×10^{-48}	2.01×10^{-2}	1.07×10^{-157}	28.7	4.31×10^{-156}	12.2	6.97×10^{-2}	4.6×10^{-2}
	500	min	**0**	5.55×10^{-51}	1.64×10^{-1}	6.92×10^{-227}	5.57	8.10×10^{-177}	98.5	97.4	9.44×10^{-1}
		mean	**0**	3.65×10^{-44}	1.78×10^{-1}	4.41×10^{-153}	73.7	4.30×10^{-156}	99	98.6	9.69×10^{-1}
		std	**0**	1.84×10^{-43}	1.52×10^{-2}	2.41×10^{-152}	28.9	2.33×10^{-155}	2.8×10^{-1}	6×10^{-1}	1.17×10^{-2}
F5	30	min	24.5	26.2	27.7	$\mathbf{5.99 \times 10^{-1}}$	27.2	26.1	88.5	27.3	17
		mean	27.1	27.9	28.4	**25.2**	27.9	27	2.62×10^{4}	28.1	67.5
		std	1.37	9.07×10^{-1}	$\mathbf{3.21 \times 10^{-1}}$	8.94	5.15×10^{-1}	5.78×10^{-1}	4.94×10^{4}	4.74×10^{-1}	30.6
	500	min	4.96×10^{2}	4.98×10^{2}	4.99×10^{2}	**1.01**	4.96×10^{2}	4.94×10^{2}	1.02×10^{9}	2.63×10^{3}	4.87×10^{3}
		mean	4.97×10^{2}	4.98×10^{2}	4.99×10^{2}	$\mathbf{4.66 \times 10^{2}}$	4.96×10^{2}	4.95×10^{2}	2.01×10^{9}	1.56×10^{4}	5.14×10^{3}
		std	4.8×10^{-1}	1.95×10^{-1}	$\mathbf{6.6 \times 10^{-2}}$	1.14×10^{2}	4.84×10^{-1}	2.98×10^{-1}	4.52×10^{8}	1.63×10^{4}	1.46×10^{2}
F6	30	min	$\mathbf{9.9 \times 10^{-6}}$	1.13	2.71	1.61×10^{-3}	1.19×10^{-1}	2.86×10^{-2}	5.14	2.02	7.75
		mean	7.21×10^{-1}	2.09	3.19	2.33	4.95×10^{-1}	$\mathbf{1.34 \times 10^{-1}}$	23.5	2.69	8.07
		std	3.38×10^{-1}	6.99×10^{-1}	3.4×10^{-1}	3.44	3.07×10^{-1}	$\mathbf{1.17 \times 10^{-1}}$	64.5	5.22×10^{-1}	1.18×10^{-1}
	500	min	60.6	99	1.14×10^{2}	$\mathbf{5.72 \times 10^{-5}}$	16.7	6.07	9.69×10^{4}	1.14×10^{2}	3.31×10^{2}
		mean	85.7	1.06×10^{2}	1.16×10^{2}	22	32.1	**16.2**	2.25×10^{5}	1.23×10^{2}	3.42×10^{2}
		std	7.37	3.21	**1.11**	46.9	10.3	6.15	5.95×10^{4}	8.4	4.69

Table 3. *Cont.*

F	*dim*	Metric	MSCSO	SCSO	AOA	BES	WOA	ROA	SCA	STOA	GA
F7	30	min	**1.11×10^{-6}**	3.21×10^{-6}	6.35×10^{-6}	9.28×10^{-4}	4.29×10^{-5}	5.79×10^{-6}	1.16×10^{-2}	2.03×10^{-3}	7.19×10^{-2}
		mean	**6.98×10^{-5}**	1.95×10^{-4}	8.25×10^{-5}	7.35×10^{-3}	3.88×10^{-3}	1.44×10^{-4}	1.11×10^{-1}	6.07×10^{-3}	1.73×10^{-1}
		std	**7.4×10^{-5}**	2.66×10^{-4}	7.76×10^{-5}	4.37×10^{-3}	3.49×10^{-3}	1.42×10^{-4}	1.17×10^{-1}	2.82×10^{-3}	5.63×10^{-2}
	500	min	**9.26×10^{-7}**	3.18×10^{-6}	5.76×10^{-6}	1.59×10^{-3}	2.45×10^{-4}	1.1×10^{-5}	8.79×10^{3}	2.15×10^{-1}	3.89×10^{3}
		mean	**6.66×10^{-5}**	3.09×10^{-4}	1.04×10^{-4}	5.15×10^{-3}	4.1×10^{-3}	1.73×10^{-4}	1.52×10^{4}	4.62×10^{-1}	4.46×10^{3}
		std	**5.99×10^{-5}**	4.23×10^{-4}	1.09×10^{-4}	3.19×10^{-3}	4.75×10^{-3}	1.7×10^{-4}	3.81×10^{3}	2.53×10^{-1}	2.77×10^{2}
F8	30	min	-9.12×10^{3}	-7.99×10^{3}	-6.03×10^{3}	-1.25×10^{4}	-1.26×10^{4}	**-1.26×10^{4}**	-4.49×10^{3}	-6.35×10^{3}	-5.71×10^{3}
		mean	-7.99×10^{3}	-6.56×10^{3}	-5.25×10^{3}	-9.66×10^{3}	-1×10^{4}	**-1.23×10^{4}**	-3.79×10^{3}	-5.37×10^{3}	-4.66×10^{3}
		std	5.05×10^{2}	9.09×10^{2}	4.7×10^{2}	2.05×10^{3}	1.76×10^{3}	4.17×10^{2}	**2.65×10^{2}**	5.59×10^{2}	6.4×10^{2}
	500	min	-8.43×10^{4}	-6.91×10^{4}	-2.6×10^{4}	-2.05×10^{5}	-2.09×10^{5}	**-2.09×10^{5}**	-1.84×10^{4}	-3.09×10^{4}	-3.67×10^{4}
		mean	-7.36×10^{4}	-5.93×10^{4}	-2.3×10^{4}	-1.6×10^{5}	-1.74×10^{5}	**-2.07×10^{5}**	-1.58×10^{4}	-2.53×10^{4}	-3.31×10^{4}
		std	4.2×10^{3}	6.73×10^{3}	1.5×10^{3}	2.65×10^{4}	2.64×10^{4}	6.23×10^{3}	1.71×10^{3}	3.86×10^{3}	**1.36×10^{3}**
F9	30	min	**0**	**0**	**0**	**0**	**0**	**0**	1.05×10^{-2}	3×10^{-8}	9.71×10^{-1}
		mean	**0**	**0**	**0**	**0**	1.89×10^{-15}	**0**	40.6	10.2	2.6
		std	**0**	**0**	**0**	**0**	1.04×10^{-14}	**0**	34.1	14.5	7.95×10^{-1}
	500	min	**0**	**0**	**0**	**0**	**0**	**0**	4.53×10^{2}	1.35×10^{-2}	2.25×10^{3}
		mean	**0**	**0**	5.45×10^{-6}	**0**	6.06×10^{-14}	**0**	1.29×10^{3}	25.9	2.39×10^{3}
		std	**0**	**0**	7.17×10^{-6}	**0**	2.31×10^{-13}	**0**	5.38×10^{2}	30.1	58.1
F10	30	min	**8.88×10^{-16}**	**8.88×10^{-16}**	**8.88×10^{-16}**	**8.88×10^{-16}**	**8.88×10^{-16}**	**8.88×10^{-16}**	2.65×10^{-2}	20	8.76×10^{-2}
		mean	**8.88×10^{-16}**	**8.88×10^{-16}**	**8.88×10^{-16}**	**8.88×10^{-16}**	5.15×10^{-15}	**8.88×10^{-16}**	14.1	20	1.36×10^{-1}
		std	**0**	**0**	**0**	**0**	2.36×10^{-15}	**0**	8.69	1.6×10^{-3}	3.13×10^{-2}
	500	min	**8.88×10^{-16}**	**8.88×10^{-16}**	7.33×10^{-3}	**8.88×10^{-16}**	**8.88×10^{-16}**	**8.88×10^{-16}**	10.2	20	2.86
		mean	**8.88×10^{-16}**	**8.88×10^{-16}**	8.08×10^{-3}	**8.88×10^{-16}**	4.91×10^{-15}	**8.88×10^{-16}**	18.9	20	2.9
		std	**0**	**0**	3.38×10^{-4}	**0**	2.59×10^{-15}	**0**	3.67	4.72×10^{-5}	2.74×10^{-2}
F11	30	min	**0**	**0**	3.81×10^{-2}	**0**	**0**	**0**	3.69×10^{-1}	3.47×10^{-8}	4.14×10^{-4}
		mean	**0**	**0**	2.6×10^{-1}	**0**	1.49×10^{-2}	**0**	9.24×10^{-1}	3.32×10^{-2}	2.11×10^{-2}
		std	**0**	**0**	1.67×10^{-1}	**0**	4.6×10^{-2}	**0**	3.27×10^{-1}	4.76×10^{-2}	1.07×10^{-1}
	500	min	**0**	**0**	6.39×10^{3}	**0**	**0**	**0**	8.3×10^{2}	1.61×10^{-1}	2.18×10^{-1}
		mean	**0**	**0**	1×10^{4}	**0**	**0**	**0**	1.65×10^{3}	7.19×10^{-1}	2.6×10^{-1}
		std	**0**	**0**	2.67×10^{3}	**0**	**0**	**0**	7.49×10^{2}	3.62×10^{-1}	9.74×10^{-2}
F12	30	min	**3.26×10^{-6}**	4.27×10^{-2}	4.35×10^{-1}	6.17×10^{-5}	6.98×10^{-3}	1.92×10^{-3}	1.92	8.13×10^{-2}	1.61
		mean	2.46×10^{-2}	1.19×10^{-1}	5.21×10^{-1}	1.52×10^{-1}	2.52×10^{-2}	**1.02×10^{-2}**	4.06×10^{5}	2.8×10^{-1}	1.73
		std	1.98×10^{-2}	6.52×10^{-2}	5.16×10^{-2}	3.83×10^{-1}	2.24×10^{-2}	**9.82×10^{-3}**	1.64×10^{6}	1.5×10^{-1}	3.77×10^{-2}
	500	min	3.4×10^{-1}	6.38×10^{-1}	1.06	**3.69×10^{-6}**	4.62×10^{-2}	8.46×10^{-3}	4.91×10^{9}	2.01	2.74
		mean	5.03×10^{-1}	7.74×10^{-1}	1.08	1.63×10^{-1}	1.01×10^{-1}	**4.4×10^{-2}**	5.79×10^{9}	4.89	2.81
		std	6.85×10^{-2}	6.54×10^{-2}	**8.88×10^{-3}**	4.16×10^{-1}	4.49×10^{-2}	2.8×10^{-2}	1.26×10^{9}	3.07	3.7×10^{-2}

Table 3. *Cont.*

F	*dim*	Metric	MSCSO	SCSO	AOA	BES	WOA	ROA	SCA	STOA	GA
F13	30	min	2.02×10^{-1}	2.03	2.58	**9×10^{-5}**	2.2×10^{-1}	2.39×10^{-2}	3.1	1.63	1.73×10^{-3}
		mean	1.52	2.38	2.83	1.42	5.16×10^{-1}	2.12×10^{-1}	1.01×10^{5}	1.94	**4.96×10^{-3}**
		std	7.43×10^{-1}	4.55×10^{-1}	1.19×10^{-1}	1.49	2.23×10^{-1}	1.36×10^{-1}	3.66×10^{5}	2.23×10^{-1}	**3.54×10^{-3}**
	500	min	48.9	49.71	50.1	**1.06×10^{-3}**	9.85	1.84	6.27×10^{9}	1.05×10^{2}	10.2
		mean	49.4	49.8	50.2	14.1	18.6	**7.45**	9.85×10^{9}	1.74×10^{2}	11
		std	1.7×10^{-1}	7.01×10^{-2}	**3.82×10^{-2}**	21.8	5.86	3.89	1.83×10^{9}	76.3	3.72×10^{-1}
F14	2	min	**9.98×10^{-1}**	**9.98×10^{-1}**	1.99	**9.98×10^{-1}**	**9.98×10^{-1}**	**9.98×10^{-1}**	**9.98×10^{-1}**	**9.98×10^{-1}**	1
		mean	4.13	5.76	9.13	3.26	2.9	4.45	**1.46**	1.98	9.68
		std	3.99	4.36	4.04	1.45	3.08	4.7	**8.53×10^{-1}**	1.91	3.61
F15	4	min	**3.07×10^{-4}**	3.07×10^{-4}	3.8×10^{-4}	5.61×10^{-4}	3.09×10^{-4}	3.09×10^{-4}	6.11×10^{-4}	3.2×10^{-4}	4.07×10^{-4}
		mean	**3.42×10^{-4}**	4.39×10^{-4}	1.62×10^{-2}	6.14×10^{-3}	7.34×10^{-4}	4.81×10^{-4}	1.11×10^{-3}	2.31×10^{-3}	1.78×10^{-2}
		std	**1.68×10^{-4}**	3.2×10^{-4}	2.63×10^{-2}	7.26×10^{-3}	5×10^{-4}	2.47×10^{-4}	3.61×10^{-4}	4.92×10^{-3}	2.5×10^{-2}
F16	2	min	**−1.03**	**−1.03**	**−1.03**	**−1.03**	**−1.03**	**−1.03**	**−1.03**	**−1.03**	**−1.03**
		mean	**−1.03**	**−1.03**	**−1.03**	-9.97×10^{-1}	**−1.03**	**−1.03**	**−1.03**	**−1.03**	−1
		std	**7.84×10^{-12}**	9.12×10^{-10}	1.57×10^{-7}	1.65×10^{-1}	1.72×10^{-9}	7.67×10^{-8}	4.86×10^{-5}	2.26×10^{-6}	1.43×10^{-2}
F17	2	min	**3.98×10^{-1}**	3.98×10^{-1}	3.98×10^{-1}	3.98×10^{-1}	3.98×10^{-1}	3.98×10^{-1}	3.98×10^{-1}	3.98×10^{-1}	3.99×10^{-1}
		mean	**3.98×10^{-1}**	3.98×10^{-1}	3.98×10^{-1}	5.31×10^{-1}	3.98×10^{-1}	3.98×10^{-1}	3.99×10^{-1}	3.98×10^{-1}	1.15
		std	**2.83×10^{-10}**	2.78×10^{-8}	8.39×10^{-8}	2.11×10^{-1}	1.08×10^{-5}	7.56×10^{-6}	1.62×10^{-3}	9.87×10^{-5}	6.38×10^{-1}
F18	5	min	**3**	**3**	**3**	3.01	**3**	**3**	**3**	**3**	3.1
		mean	**3**	**3**	8.4	6.41	**3**	**3**	**3**	**3**	24.2
		std	**2.59×10^{-7}**	1.1×10^{-5}	11	10.4	1.07×10^{-4}	1.25×10^{-4}	2.05×10^{-4}	2.27×10^{-4}	21.3
F19	3	min	**−3.86**	−3.86	−3.86	−3.85	−3.86	−3.86	−3.86	−3.86	−3.86
		mean	**−3.86**	−3.86	−3.85	−3.7	−3.86	−3.86	−3.85	−3.86	−3.71
		std	**1.82×10^{-8}**	4.3×10^{-3}	4.49×10^{-3}	2.17×10^{-1}	4.36×10^{-3}	2.5×10^{-3}	6.17×10^{-3}	7.68×10^{-3}	3.15×10^{-1}
F20	6	min	**−3.32**	−3.32	−3.16	−3.25	−3.32	−3.32	−3.12	−3.13	−3.32
		mean	**−3.29**	−3.2	−3.05	−2.87	−3.25	−3.24	−2.77	−2.93	−3.28
		std	**5.54×10^{-2}**	1.47×10^{-1}	9.24×10^{-2}	2.62×10^{-1}	9.14×10^{-2}	9.84×10^{-2}	5.08×10^{-1}	4.13×10^{-1}	5.8×10^{-2}
F21	4	min	**−10.2**	−10.2	−6.91	−10.2	−10.2	−10.2	−4.8	−10.1	−5.05
		mean	**−10.2**	−4.9	−3.78	−6.15	−7.52	−10.1	−1.92	−3.5	−1.1
		std	**2.08×10^{-6}**	1.94	1.4	2.66	2.92	2.75×10^{-2}	1.56	3.9	1.11
F22	4	min	**−10.4**	−10.4	−6.87	−10.3	−10.4	−10.4	−5.72	−10.3	−5.08
		mean	**−10.4**	−6.56	−3.43	−6.16	−7.11	−10.4	−3.43	−5.88	−1.23
		std	**4.87×10^{-6}**	2.6	1.41	2.22	3	3.04×10^{-2}	1.77	4.43	8.7×10^{-1}
F23	4	min	**−10.5**	−10.5	−7.27	−10.5	−10.5	−10.5	−5.15	−10.5	−5.13
		mean	**−10.5**	−7.11	−4	−6.4	−6.69	−10.5	−3.65	−8.08	−1.66
		std	**1.97×10^{-6}**	2.95	1.77	3.16	3.3	2.27×10^{-2}	2.02	3.96	1.09

The Table 3 analysis cannot fully prove the optimization effect of the MSCSO algorithm in the 23 standard benchmark functions. In order to better understand the optimization effect of MSCSO, Figures 8–10 show the convergence curves of each algorithm. It can be seen from the image that the MSCSO algorithm has a strong convergence ability in F1–F4, and the optimal value is found quickly. There is a small gap between algorithms in F5. In F6 and F12 of Figure 8, the MSCSO algorithm can jump out of the local optimum in the later stage so that the algorithm can converge better. Because the walking strategy is added, the sand cat group has stronger mobility, which makes the sand cat have a stronger walking ability. The exploration ability of the MSCSO algorithm is enhanced by lens opposition-based learning. It can be concluded that the MSCSO algorithm has a better optimization effect than the SCSO algorithm in these functions. In F7, the MSCSO algorithm can quickly find a very excellent fitness value. This shows that the exploration ability of the MSCSO algorithm has been enhanced and better solutions can be found. In F9–F11, the MSCSO algorithm can quickly find the optimal value compared with other comparison algorithms. In F14–F23, these algorithms can find a better fitness value. These algorithms have good optimization effects, but the MSCSO algorithm can also find very good fitness values. It can be seen from F14, F15, F21, F22, and F23 that the MSCSO algorithm is more excellent. According to the comprehensive analysis of tables and images, the MSCSO algorithm is more stable and can find better values.

5.1.2. Analysis of the Wilcoxon Rank Sum Test Results

The Wilcoxon rank sum test is a nonparametric statistical test that can find more complex data distribution. Table 3 gives the best fitness value, average value, and standard deviation of each algorithm but does not compare with the results of multiple algorithms. Therefore, the Wilcoxon rank sum test is required for further verification and testing. Table 4 shows the experimental results of the MSCSO algorithm and eight other different algorithms running thirty times in the twenty-three standard benchmark functions. The significance level is 5%. Less than 5% indicates a significant difference between the two algorithms. It can be seen from the table that most test results are less than 5%, but some results are more than 5%. There are many results equal to one in F9–F11. This is because many algorithms can find the optimal value in F9–F11, resulting in the consistency of the final optimal fitness value. The MSCSO and BES algorithms have many results greater than 5% in unimodal functions, which shows that these two algorithms have good convergence ability in unimodal functions. Many algorithms can find a better value in F14 because the function is relatively simple. In the rest of the functions, the MSCSO algorithm has a significant difference compared with other algorithms. The MSCSO algorithm has generally achieved good results in the Wilcoxon rank sum test.

The above experimental analysis shows that the MSCSO algorithm has a good optimization effect in the 23 standard benchmark functions. Compared with the SCSO algorithm, it has excellent improvement. Compared with other comparison algorithms, it also has more significant advantages.

5.2. Experiments on the CEC2014 Benchmark Function

The 23 standard benchmark functions are simple test functions, which are insufficient to prove the MSCSO algorithm's optimization performance fully. In order to thoroughly verify the optimization effect of the MSCSO algorithm, the CEC2014 benchmark function is used for testing in this section. Table 5 shows the specific introduction of the CEC2014 benchmark functions. Set the number of individuals of each algorithm $N = 30$, the maximum number of iterations $T = 500$, and the dimension $dim = 10$. Eight algorithms run thirty times independently to obtain each algorithm's best, average, and standard deviation.

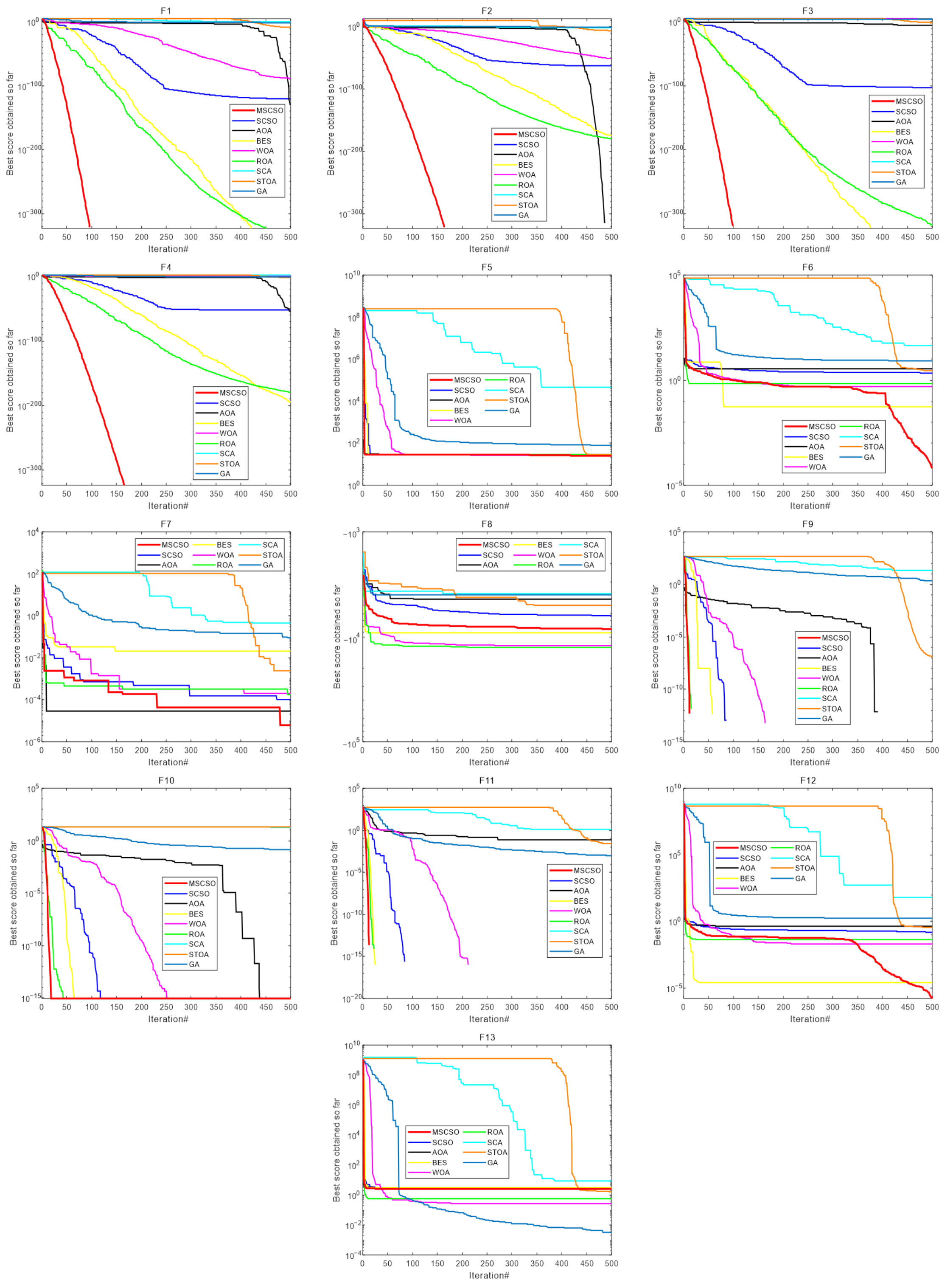


Figure 8. Convergence curves for the optimization algorithms for standard benchmark functions (F1–F13) with *dim* = 30.

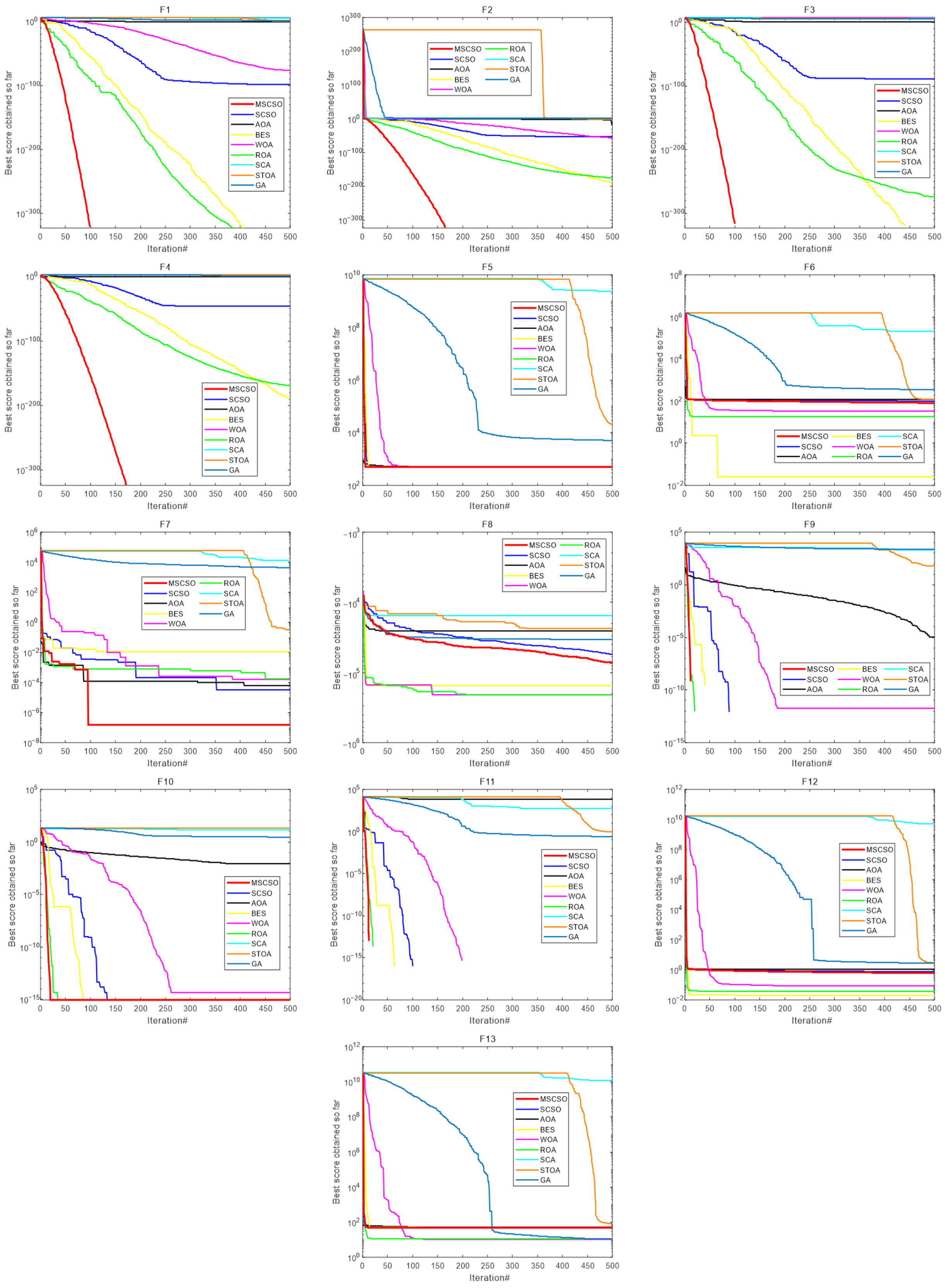


Figure 9. Convergence curves for the optimization algorithms for standard benchmark functions (F1–F13) with *dim* = 500.

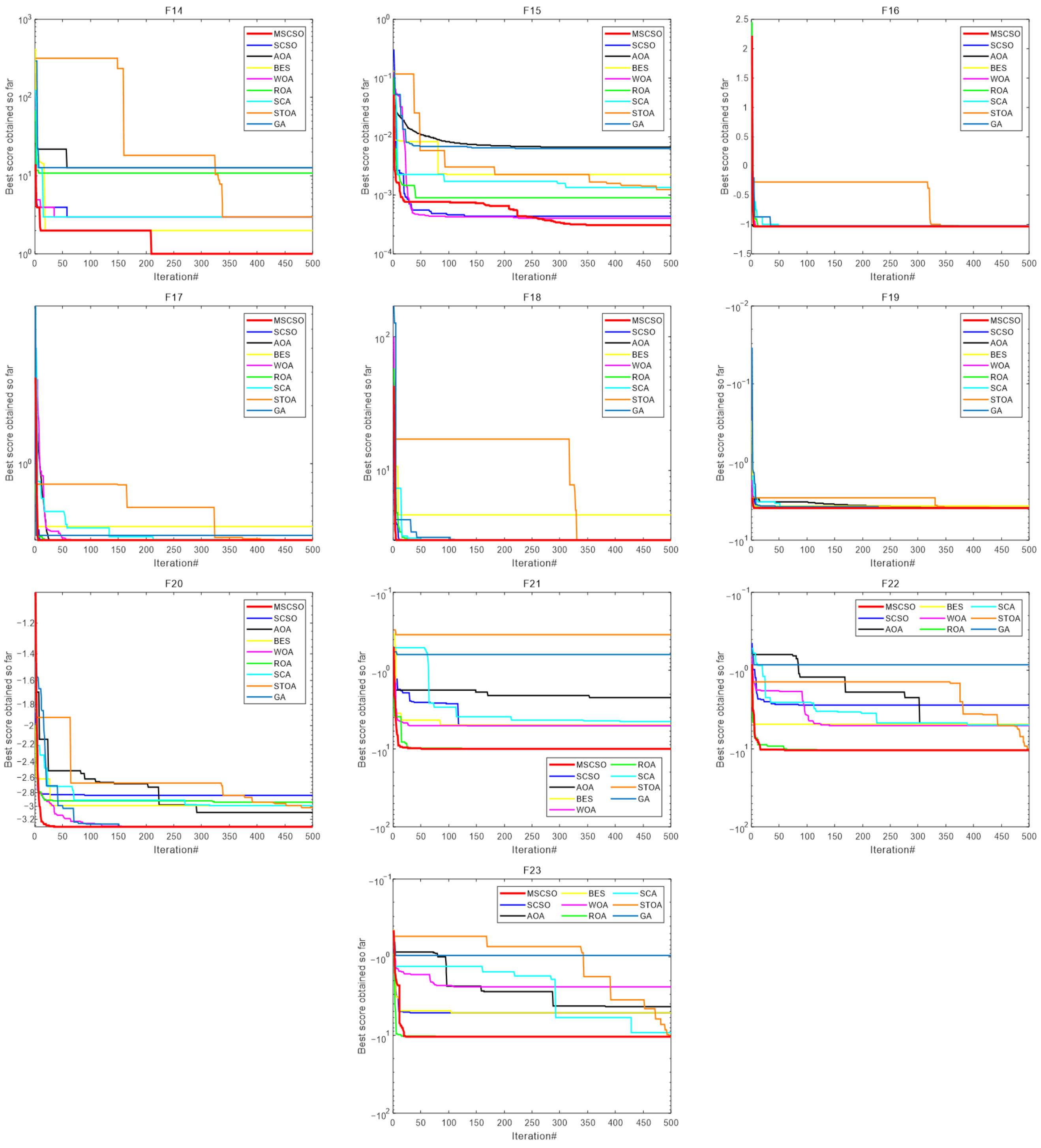


Figure 10. Convergence curves for the optimization algorithms for standard benchmark functions (F14–F23).

Table 4. Experimental results of the Wilcoxon rank−sum test on the 23 standard benchmark functions (We bold the good data).

F	*dim*	**SCSO vs. MSCSO**	**AOA vs. MSCSO**	**BES vs. MSCSO**	**WOA vs. MSCSO**	**ROA vs. MSCSO**	**SCA vs. MSCSO**	**STOA vs. MSCSO**	**GA vs. MSCSO**
F1	30	1.73×10^{-6}	1.73×10^{-6}	**1**	1.73×10^{-6}	$\mathbf{2.5 \times 10^{-1}}$	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
	500	1.73×10^{-6}	1.73×10^{-6}	**1**	1.73×10^{-6}	3.13×10^{-2}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F2	30	1.73×10^{-6}	**1**	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
	500	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F3	30	1.73×10^{-6}	1.73×10^{-6}	**1**	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
	500	1.73×10^{-6}	1.73×10^{-6}	$\mathbf{1.25 \times 10^{-1}}$	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F4	30	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
	500	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F5	30	4.49×10^{-2}	4.07×10^{-5}	2.96×10^{-3}	1.29×10^{-3}	$\mathbf{9.1 \times 10^{-1}}$	1.73×10^{-6}	2.6×10^{-5}	8.47×10^{-6}
	500	2.35×10^{-6}	1.73×10^{-6}	8.73×10^{-3}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F6	30	4.29×10^{-6}	1.73×10^{-6}	1.66×10^{-2}	7.51×10^{-5}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
	500	1.73×10^{-6}	1.73×10^{-6}	7.69×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F7	30	3.85×10^{-3}	$\mathbf{9.26 \times 10^{-1}}$	1.73×10^{-6}	2.35×10^{-6}	$\mathbf{1.06 \times 10^{-1}}$	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
	500	3.5×10^{-2}	$\mathbf{7.81 \times 10^{-1}}$	1.73×10^{-6}	1.73×10^{-6}	4.11×10^{-3}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F8	30	1.02×10^{-5}	1.73×10^{-6}	4.73×10^{-6}	1.92×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
	500	4.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F9	30	**1**	**1**	**1**	**1**	**1**	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
	500	**1**	4.38×10^{-4}	**1**	**1**	**1**	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F10	30	**1**	**1**	**1**	8.19×10^{-6}	**1**	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
	500	**1**	1.73×10^{-6}	$\mathbf{5 \times 10^{-1}}$	1.87×10^{-6}	**1**	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F11	30	**1**	1.73×10^{-6}	**1**	$\mathbf{5 \times 10^{-1}}$	**1**	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
	500	**1**	1.73×10^{-6}	**1**	**1**	**1**	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F12	30	1.73×10^{-6}	1.73×10^{-6}	$\mathbf{2.99 \times 10^{-1}}$	$\mathbf{8.97 \times 10^{-2}}$	5.29×10^{-4}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
	500	2.13×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F13	30	4.2×10^{-4}	1.73×10^{-6}	2.43×10^{-2}	5.75×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	$\mathbf{7.66 \times 10^{-1}}$	1.73×10^{-6}
	500	1.73×10^{-6}	1.73×10^{-6}	1.24×10^{-5}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F14	2	$\mathbf{8.61 \times 10^{-1}}$	1.36×10^{-5}	$\mathbf{9.92 \times 10^{-1}}$	$\mathbf{3.82 \times 10^{-1}}$	$\mathbf{9.1 \times 10^{-1}}$	$\mathbf{5.98 \times 10^{-2}}$	$\mathbf{1.53 \times 10^{-1}}$	1.64×10^{-5}
F15	4	4.86×10^{-5}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.36×10^{-5}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F16	2	1.92×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	3.72×10^{-5}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F17	2	3.88×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F18	5	1.92×10^{-6}	$\mathbf{5.17 \times 10^{-1}}$	1.73×10^{-6}	1.24×10^{-6}	9.32×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F19	6	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F20	3	6.04×10^{-3}	1.73×10^{-6}	1.73×10^{-6}	9.84×10^{-3}	8.22×10^{-3}	1.73×10^{-6}	4.29×10^{-6}	4.07×10^{-2}
F21	4	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F22	4	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F23	4	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}

Table 5. Details of 30 CEC2014 benchmark functions.

Name	NO.	Functions	F_{min}
Unimodal Functions	CEC 1	Rotated High Conditioned Elliptic Function	100
	CEC 2	Rotated Bent Cigar Function	200
	CEC 3	Rotated Discus Function	300
Simple Multimodal Functions	CEC 4	Shifted and Rotated Rosenbrock's Function	400
	CEC 5	Shifted and Rotated Ackley's Function	500
	CEC 6	Shifted and Rotated Weierstrass Function	600
	CEC 7	Shifted and Rotated Griewank's Function	700
	CEC 8	Shifted Rastrigin's Function	800
	CEC 9	Shifted and Rotated Rastrigin's Function	900
	CEC 10	Shifted Schwefel's Function	1000
	CEC 11	Shifted and Rotated Schwefel's Schwefel's Function	1100
	CEC 12	Shifted and Rotated Katsuura Function	1200
	CEC 13	Shifted and Rotated HappyCat Function	1300
	CEC 14	Shifted and Rotated HGBat Function	1400
	CEC 15	Shifted and Rotated Expanded Griewank'splus Rosenbrock's Function	1500
	CEC 16	Shifted and Rotated Expanded Scaffer's F6 Function	1600
Hybrid Function 1	CEC 17	Hybrid Function 1 (N = 3)	1700
	CEC 18	Hybrid Function 2 (N = 3)	1800
	CEC 19	Hybrid Function 3 (N = 4)	1900
	CEC 20	Hybrid Function 4 (N = 4)	2000
	CEC 21	Hybrid Function 5 (N = 5)	2100
	CEC 22	Hybrid Function 6 (N = 5)	2200
Composition Functions	CEC 23	Composition Function 1 (N = 5)	2300
	CEC 24	Composition Function 2 (N = 3)	2400
	CEC 25	Composition Function 3 (N = 3)	2500
	CEC 26	Composition Function 4 (N = 5)	2600
	CEC 27	Composition Function 5 (N = 5)	2700
	CEC 28	Composition Function 6 (N = 5)	2800
	CEC 29	Composition Function 7 (N = 3)	2900
	CEC 30	Composition Function 8 (N = 3)	3000
Search Range: $[-100, 100]^{dim}$			

5.2.1. The CEC2014 Benchmark Function Results Statistics and Image Analysis

Table 6 shows the statistical results of the benchmark functions of the MSCSO algorithm and the eight comparison algorithms in CEC2014. The data in the table refer to literature [43]. From the table data, it can be concluded that the MSCSO algorithm has achieved good results in the CEC2014 benchmark function. In CEC1–CEC3, the MSCSO algorithm can obtain a better fitness value compared with other comparison algorithms. Only in CEC2 is the stability inferior to the WOA. In CEC4–CEC8, the MSCSO algorithm can obtain a better fitness value, but its stability is not enough. Because the MSCSO algorithm may find a better solution through the walking strategy, but it is not necessarily able to find a better fitness value. However, the solution found is generally superior to other algorithms. In CEC9, the STOA algorithm can obtain a better fitness value. In CEC10–CEC16, the MSCSO algorithm obtains a better fitness value. Only part of the standard deviation of the function is insufficient. In CEC17–CEC30, the MSCSO algorithm has a very significant optimization effect. The standard deviation of CEC22 and CEC27 is lower than that of the SCA and ROA. The average fitness value and standard deviation of CEC24 are insufficient. Among other functions, the MSCSO algorithm achieves the optimal value. According to the analysis in Table 6, the addition of a walking strategy and a lens position-based learning improves the exploration ability of the algorithm, making the MSCSO algorithm have a stronger optimization ability.

Table 6. CEC2014 Algorithm Results of the Benchmark Function.

CEC	Metric	MSCSO	SCSO	AOA	BES	WOA	ROA	SCA	STOA	GA
	min	**1.1×10^{5}**	3.11×10^{5}	5.86×10^{6}	1.49×10^{7}	1.57×10^{6}	8.09×10^{5}	3.25×10^{6}	6.71×10^{5}	4.74×10^{6}
CEC 1	mean	**4.85×10^{6}**	8.49×10^{6}	7.06×10^{7}	9.19×10^{7}	1.33×10^{7}	1.84×10^{7}	1.18×10^{7}	5.27×10^{6}	7.96×10^{7}
	std	**4.17×10^{6}**	5.21×10^{6}	7.92×10^{7}	3.95×10^{7}	9.26×10^{6}	1.3×10^{7}	4.46×10^{6}	4.54×10^{6}	7.05×10^{7}
	min	**3.42×10^{2}**	4.8×10^{3}	2.91×10^{9}	3.55×10^{8}	1.09×10^{6}	1.03×10^{7}	5.93×10^{8}	1.89×10^{6}	1.94×10^{9}
CEC 2	mean	**1.98×10^{7}**	9.73×10^{7}	6.65×10^{9}	2.5×10^{9}	3.72×10^{7}	8.66×10^{8}	1.04×10^{9}	4.73×10^{8}	4.25×10^{9}
	std	6.8×10^{7}	3.3×10^{8}	2.1×10^{9}	1.69×10^{9}	**5.70×10^{7}**	8.71×10^{8}	3.64×10^{8}	4.5×10^{8}	1.37×10^{9}
	min	**6.28×10^{2}**	1.58×10^{3}	1.08×10^{4}	1.32×10^{4}	1.32×10^{4}	2.14×10^{3}	2.09×10^{3}	2.66×10^{3}	8.27×10^{3}
CEC 3	mean	**3.91×10^{3}**	6.51×10^{3}	1.82×10^{4}	6.91×10^{4}	5.52×10^{4}	7.73×10^{3}	1.15×10^{4}	1.41×10^{4}	3.91×10^{5}
	std	**3.26×10^{3}**	3.43×10^{3}	4.6×10^{3}	7.61×10^{4}	2.82×10^{4}	3.64×10^{3}	8.21×10^{3}	8.94×10^{3}	6.95×10^{5}
	min	**4×10^{2}**	4.02×10^{2}	5.4×10^{2}	4.96×10^{2}	4.05×10^{2}	4.2×10^{2}	4.47×10^{2}	4.19×10^{2}	5.19×10^{2}
CEC 4	mean	**4.3×10^{2}**	4.42×10^{2}	1.7×10^{3}	9.18×10^{2}	4.65×10^{2}	4.79×10^{2}	4.9×10^{2}	4.54×10^{2}	1.02×10^{3}
	std	**32.7**	**25.5**	7.75×10^{2}	3.03×10^{2}	44.4	55.5	32.8	32.4	4.24×10^{2}
	min	**5.2×10^{2}**	5.2×10^{2}	5.2×10^{2}	5.2×10^{2}	5.20×10^{2}	5.2×10^{2}	5.2×10^{2}	5.2×10^{2}	5.2×10^{2}
CEC 5	mean	**5.2×10^{2}**	5.2×10^{2}	5.2×10^{2}	5.2×10^{2}	5.20×10^{2}	5.2×10^{2}	5.2×10^{2}	5.2×10^{2}	5.2×10^{2}
	std	6.61×10^{-2}	1.11×10^{-1}	**5.24×10^{-2}**	1.32×10^{-1}	1.44×10^{-1}	1.25×10^{-1}	7.51×10^{-2}	9.08×10^{-2}	2.36×10^{-1}
	min	**6.01×10^{2}**	6.04×10^{2}	6.08×10^{2}	6.05×10^{2}	6.05×10^{2}	6.04×10^{2}	6.05×10^{2}	6.04×10^{2}	6.07×10^{2}
CEC 6	mean	**6.05×10^{2}**	6.06×10^{2}	6.1×10^{2}	6.09×10^{2}	6.09×10^{2}	6.07×10^{2}	6.08×10^{2}	6.08×10^{2}	6.09×10^{2}
	std	1.9	1.54	**9.85×10^{-1}**	1.89	1.83	1.6	1.25	1.47	1.3
	min	**7×10^{2}**	7×10^{2}	7.43×10^{2}	7.19×10^{2}	7.01×10^{2}	7.01×10^{2}	7.08×10^{2}	7.01×10^{2}	7.35×10^{2}
CEC 7	mean	**7.01×10^{2}**	7.02×10^{2}	8.44×10^{2}	7.57×10^{2}	7.02×10^{2}	7.05×10^{2}	7.14×10^{2}	7.05×10^{2}	7.79×10^{2}
	std	6.16×10^{-1}	2.4	51.8	32.9	**5.19×10^{-1}**	5.96	3.35	4.2	31.6
	min	**8.03×10^{2}**	8.09×10^{2}	8.24×10^{2}	8.38×10^{2}	8.13×10^{2}	8.16×10^{2}	8.29×10^{2}	8.12×10^{2}	8.57×10^{2}
CEC 8	mean	**8.18×10^{2}**	8.34×10^{2}	8.52×10^{2}	8.69×10^{2}	8.48×10^{2}	8.39×10^{2}	8.48×10^{2}	8.26×10^{2}	8.79×10^{2}
	std	7.87	12.4	14.3	16.6	19.3	11.7	**7.8**	9.67	14.9
	min	9.16×10^{2}	9.14×10^{2}	9.24×10^{2}	9.48×10^{2}	9.2×10^{2}	9.15×10^{2}	9.37×10^{2}	**9.12×10^{2}**	9.47×10^{2}
CEC 9	mean	9.34×10^{2}	9.37×10^{2}	9.45×10^{2}	9.66×10^{2}	9.52×10^{2}	9.44×10^{2}	9.49×10^{2}	**9.32×10^{2}**	9.74×10^{2}
	std	11.8	9.09	9.26	12.4	21	11	**8.29**	10.1	13.5
	min	**1.04×10^{3}**	1.47×10^{3}	1.14×10^{3}	1.75×10^{3}	1.07×10^{3}	1.11×10^{3}	1.77×10^{3}	1.46×10^{3}	1.44×10^{3}
CEC 10	mean	**1.22×10^{3}**	1.79×10^{3}	1.74×10^{3}	2.3×10^{3}	1.72×10^{3}	1.7×10^{3}	2.16×10^{3}	1.83×10^{3}	1.97×10^{3}
	std	1.89×10^{2}	**1.85×10^{2}**	2.34×10^{2}	2.75×10^{2}	2.49×10^{2}	2.64×10^{2}	1.87×10^{2}	2.21×10^{2}	2.59×10^{2}
	min	**1.15×10^{3}**	1.64×10^{3}	1.65×10^{3}	2.32×10^{3}	1.94×10^{3}	1.75×10^{3}	2.22×10^{3}	1.78×10^{3}	2.15×10^{3}
CEC 11	mean	**1.84×10^{3}**	2.04×10^{3}	2.08×10^{3}	2.77×10^{3}	2.25×10^{3}	2.18×10^{3}	2.58×10^{3}	2.26×10^{3}	2.89×10^{3}
	std	2.95×10^{2}	3.18×10^{2}	3.49×10^{2}	2.47×10^{2}	3.45×10^{2}	3.57×10^{2}	**2.25×10^{2}**	3.52×10^{2}	3.49×10^{2}
	min	**1.2×10^{3}**	1.2×10^{3}	1.2×10^{3}	1.2×10^{3}	1.2×10^{3}	1.2×10^{3}	1.2×10^{3}	1.2×10^{3}	1.2×10^{3}
CEC 12	mean	**1.2×10^{3}**	1.2×10^{3}	1.2×10^{3}	1.2×10^{3}	1.2×10^{3}	1.2×10^{3}	1.2×10^{3}	1.2×10^{3}	1.2×10^{3}
	std	**1.5×10^{-1}**	3×10^{-1}	2.71×10^{-1}	3.7×10^{-1}	4.86×10^{-1}	3.37×10^{-1}	3.12×10^{-1}	3.86×10^{-1}	6.82×10^{-1}

Table 6. *Cont.*

CEC	Metric	MSCSO	SCSO	AOA	BES	WOA	ROA	SCA	STOA	GA
	min	**1.3×10^{3}**	1.3×10^{3}	1.3×10^{3}	1.3×10^{3}	1.3×10^{3}	1.3×10^{3}	1.3×10^{3}	1.3×10^{3}	1.3×10^{3}
CEC 13	mean	**1.3×10^{3}**	1.3×10^{3}	1.3×10^{3}	1.3×10^{3}	1.3×10^{3}	1.3×10^{3}	1.3×10^{3}	1.3×10^{3}	1.3×10^{3}
	std	1.29×10^{-1}	3.81×10^{-1}	1.16	1.23	2.23×10^{-1}	7.01×10^{-1}	**1.23×10^{-1}**	2.19×10^{-1}	9.89×10^{-1}
	min	**1.4×10^{3}**	1.4×10^{3}	1.41×10^{3}	1.41×10^{3}	1.4×10^{3}	1.4×10^{3}	1.4×10^{3}	1.4×10^{3}	1.4×10^{3}
CEC 14	mean	1.4×10^{3}	1.4×10^{3}	1.43×10^{3}	1.42×10^{3}	**1.4×10^{3}**	1.4×10^{3}	1.4×10^{3}	1.4×10^{3}	1.41×10^{3}
	std	**2.31×10^{-1}**	1.11	11.1	9.56	3.18×10^{-1}	5	5.64×10^{-1}	1.05	6.66
	min	**1.5×10^{3}**	1.5×10^{3}	1.62×10^{3}	1.54×10^{3}	1.5×10^{3}	1.5×10^{3}	1.51×10^{3}	1.5×10^{3}	1.52×10^{3}
CEC 15	mean	**1.5×10^{3}**	1.52×10^{3}	5.06×10^{3}	3.09×10^{3}	1.51×10^{3}	1.68×10^{3}	1.52×10^{3}	1.52×10^{3}	4.88×10^{3}
	std	**1.2**	89.1	5.61×10^{3}	3.39×10^{3}	9.82	6.46×10^{2}	54.4	94	5.41×10^{3}
	min	**1.6×10^{3}**	1.6×10^{3}	1.6×10^{3}	1.6×10^{3}	1.6×10^{3}	1.6×10^{3}	1.6×10^{3}	1.6×10^{3}	1.6×10^{3}
CEC 16	mean	**1.6×10^{3}**	1.6×10^{3}	1.6×10^{3}	1.6×10^{3}	1.6×10^{3}	1.6×10^{3}	1.6×10^{3}	1.6×10^{3}	1.6×10^{3}
	std	3.67×10^{-1}	4.48×10^{-1}	3.03×10^{-1}	3.16×10^{-1}	4.77×10^{-1}	3.7×10^{-1}	**2.66×10^{-1}**	3.87×10^{-1}	2.91×10^{-1}
	min	**1.95×10^{3}**	3.16×10^{3}	6.99×10^{4}	2.27×10^{4}	1.05×10^{4}	3.88×10^{3}	1.87×10^{4}	7.62×10^{3}	5.18×10^{5}
CEC 17	mean	**6.66×10^{3}**	4.9×10^{4}	5.22×10^{5}	9.41×10^{5}	3.7×10^{5}	1.24×10^{5}	8.44×10^{4}	1.57×10^{5}	8.93×10^{6}
	std	**3.29×10^{3}**	1.34×10^{5}	3.95×10^{5}	1.63×10^{6}	6.73×10^{5}	1.89×10^{5}	1.41×10^{5}	2.04×10^{5}	1.64×10^{7}
	min	**1.88×10^{3}**	3×10^{3}	2.6×10^{3}	8.21×10^{3}	2.58×10^{3}	2.98×10^{3}	1.1×10^{4}	3.04×10^{3}	1.45×10^{4}
CEC 18	mean	**1×10^{4}**	1.52×10^{4}	1.4×10^{4}	1.6×10^{6}	1.67×10^{4}	1.22×10^{4}	6.31×10^{4}	1.92×10^{4}	3.29×10^{7}
	std	**5.84×10^{3}**	9.62×10^{3}	9.68×10^{3}	5.46×10^{6}	1.42×10^{4}	9.04×10^{3}	9.88×10^{4}	1.62×10^{4}	4.42×10^{7}
	min	**1.9×10^{3}**	1.9×10^{3}	1.91×10^{3}	1.91×10^{3}	1.9×10^{3}	1.9×10^{3}	1.91×10^{3}	1.9×10^{3}	1.91×10^{3}
CEC 19	mean	**1.9×10^{3}**	1.9×10^{3}	1.94×10^{3}	1.91×10^{3}	1.91×10^{3}	$1.91 \times 10^{+}$	1.91×10^{3}	1.9×10^{3}	1.93×10^{3}
	std	**6.93×10^{-1}**	1.41	28.6	10.4	2.23	12.7	1.35	1.24	22.8
	min	**2.04×10^{3}**	2.67×10^{3}	5.66×10^{3}	4.3×10^{3}	2.57×10^{3}	2.27×10^{3}	2.9×10^{3}	2.55×10^{3}	7.6×10^{3}
CEC 20	mean	**5.83×10^{3}**	8.12×10^{3}	1.38×10^{4}	1.12×10^{5}	1.5×10^{4}	1.04×10^{4}	9.66×10^{3}	1.4×10^{4}	1.53×10^{7}
	std	**3.17×10^{3}**	4.05×10^{3}	1.03×10^{4}	4.47×10^{5}	1.25×10^{4}	5.03×10^{3}	7.32×10^{3}	9.73×10^{3}	2.34×10^{7}
	min	**2.29×10^{3}**	3.19×10^{3}	7.03×10^{3}	4.92×10^{3}	1.25×10^{4}	3.16×10^{3}	$7. \times 10^{3}$	3.64×10^{3}	7.7×10^{4}
CEC 21	mean	**7.85×10^{3}**	1.08×10^{4}	1.57×10^{6}	3.77×10^{5}	1.05×10^{6}	5.54×10^{5}	2.05×10^{4}	1.42×10^{4}	3.13×10^{6}
	std	**4.57×10^{3}**	6.58×10^{3}	2.3×10^{6}	8.68×10^{5}	3.05×10^{6}	2.97×10^{6}	1.12×10^{4}	1.05×10^{4}	3.71×10^{6}
	min	**2.22×10^{3}**	2.24×10^{3}	2.28×10^{3}	2.26×10^{3}	2.23×10^{3}	2.23×10^{3}	2.26×10^{3}	2.24×10^{3}	2.32×10^{3}
CEC 22	mean	**2.25×10^{3}**	2.32×10^{3}	2.42×10^{3}	2.42×10^{3}	2.33×10^{3}	2.31×10^{3}	2.3×10^{3}	2.29×10^{3}	2.67×10^{3}
	std	47.9	66.5	1.12×10^{2}	1.11×10^{2}	97.1	79.7	**43**	62.4	1.74×10^{2}
	min	**2.5×10^{3}**	**2.5×10^{3}**	**2.5×10^{3}**	**2.5×10^{3}**	**2.5×10^{3}**	**2.5×10^{3}**	2.64×10^{3}	2.63×10^{3}	2.5×10^{3}
CEC 23	mean	**2.5×10^{3}**	**2.5×10^{3}**	**2.5×10^{3}**	2.6×10^{3}	2.64×10^{3}	**2.5×10^{3}**	2.65×10^{3}	2.65×10^{3}	2.7×10^{3}
	std	**0**	**0**	**0**	1.03×10^{2}	28.7	**0**	9.08	10.4	1.12×10^{2}
	min	**2.52×10^{3}**	2.56×10^{3}	2.55×10^{3}	2.57×10^{3}	2.54×10^{3}	2.56×10^{3}	2.55×10^{3}	2.53×10^{3}	2.56×10^{3}
CEC 24	mean	2.59×10^{3}	2.6×10^{3}	2.59×10^{3}	2.59×10^{3}	2.59×10^{3}	2.6×10^{3}	2.56×10^{3}	**2.55×10^{3}**	2.6×10^{3}
	std	23.9	**7.09**	20.3	14.3	28	7.1	10.1	20.6	17.8

Table 6. *Cont.*

CEC	Metric	MSCSO	SCSO	AOA	BES	WOA	ROA	SCA	STOA	GA
CEC 25	min	**2.64×10^3**	2.7×10^3	2.7×10^3	2.69×10^3	2.69×10^3	2.7×10^3	2.69×10^3	2.7×10^3	2.69×10^3
	mean	**2.7×10^3**	**2.7×10^3**	**2.7×10^3**	**2.7×10^3**	**2.7×10^3**	**2.7×10^3**	**2.7×10^3**	**2.7×10^3**	2.71×10^3
	std	**0**	**0**	1.89	7.13	5.95	8.37	8.8	1.34	4.6
CEC 26	min	**2.7×10^3**	2.7×10^3	2.7×10^3	2.7×10^3	2.7×10^3	2.7×10^3	2.7×10^3	2.7×10^3	2.7×10^3
	mean	**2.7×10^3**	2.7×10^3	2.72×10^3	2.7×10^3	2.7×10^3	2.7×10^3	2.7×10^3	2.7×10^3	2.71×10^3
	std	**1.01×10^{-1}**	3.82×10^{-1}	29.5	1.39	18.2	18.1	2.12×10^{-1}	1.34×10^{-1}	25
CEC 27	min	**2.7×10^3**	2.71×10^3	2.9×10^3	2.86×10^3	3.1×10^3	2.9×10^3	2.73×10^3	3.1×10^3	2.75×10^3
	mean	**2.89×10^3**	**2.89×10^3**	2.92×10^3	3.14×10^3	3.14×10^3	2.9×10^3	3.02×10^3	3.17×10^3	3.2×10^3
	std	35.1	35.4	89.2	1.76×10^2	1.37×10^2	**0**	1.63×10^2	65.1	1.47×10^2
CEC 28	min	**3×10^3**	**3×10^3**	**3×10^3**	**3×10^3**	3.23×10^3	**3×10^3**	3.24×10^3	3.17×10^3	3.54×10^3
	mean	**3×10^3**	**3×10^3**	3.06×10^3	3.34×10^3	3.45×10^3	**3×10^3**	3.3×10^3	3.19×10^3	3.88×10^3
	std	**0**	**0**	2.35×10^2	2.35×10^2	1.87×10^2	**0**	72.7	12	2.2×10^2
CEC 29	min	**3.1×10^3**	**3.1×10^3**	**3.1×10^3**	5.19×10^3	3.46×10^3	3.36×10^3	4.47×10^3	3.65×10^3	5.62×10^3
	mean	**3.62×10^3**	2.03×10^5	2.29×10^6	1.06×10^6	4.86×10^5	2.42×10^5	2.46×10^4	6.52×10^3	8.91×10^6
	std	**3.94×10^2**	6.07×10^5	7.64×10^6	1.65×10^6	1.16×10^6	6.18×10^5	2.69×10^4	4.52×10^3	1.42×10^7
CEC 30	min	**3.2×10^3**	3.94×10^3	3.2×10^3	5.33×10^3	4.19×10^3	3.94×10^3	4.41×10^3	3.72×10^3	1.01×10^4
	mean	**4.21×10^3**	5.11×10^3	5.72×10^4	3.49×10^4	7.96×10^3	5.21×10^3	5.59×10^3	4.32×10^3	6.06×10^4
	std	**4.93×10^2**	9.05×10^2	9.2×10^4	8.4×10^4	7.98×10^3	1.28×10^3	1.25×10^3	6.61×10^2	8.71×10^4

Figure 11 shows the convergence curve of the MSCSO algorithm and eight comparison algorithms in the CEC2014 benchmark function. It can be seen that the MSCSO algorithm has better convergence ability. In the unimodal functions of CEC1–CEC3, the MSCSO algorithm can find a better location and converge constantly. The SCSO algorithm is easy to fall into the local optimum. The convergence curve of other comparison algorithms is still inferior to the MSCSO algorithm. In simple multimodal functions, the MSCSO algorithm also has better global optimization capability. From the convergence curve of CEC4–CEC17, it can be seen that the MSCSO algorithm can find a better position in many functions and converge quickly. The MSCSO with the TW, LFW, and LOBL has a stronger exploration ability. They can jump out of the local optimum and obtain a better fitness value. In CEC17–CEC30, many algorithms are trapped in local optima, resulting in the algorithm not being able to converge better. However, the MSCSO algorithm can find a better location in CEC17, CEC18, CEC20, CEC21, CEC24, and CEC25, which makes the algorithm converge better to the best solution.

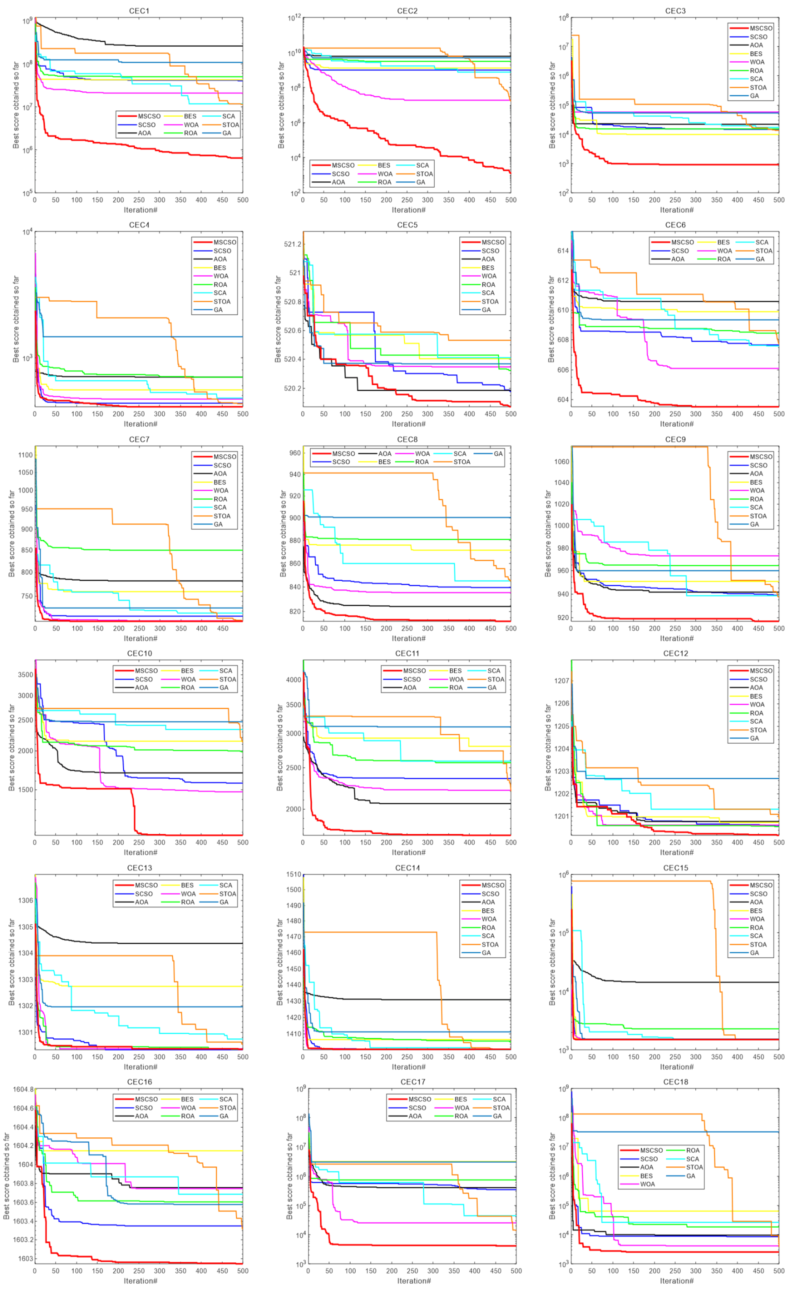


Figure 11. *Cont.*

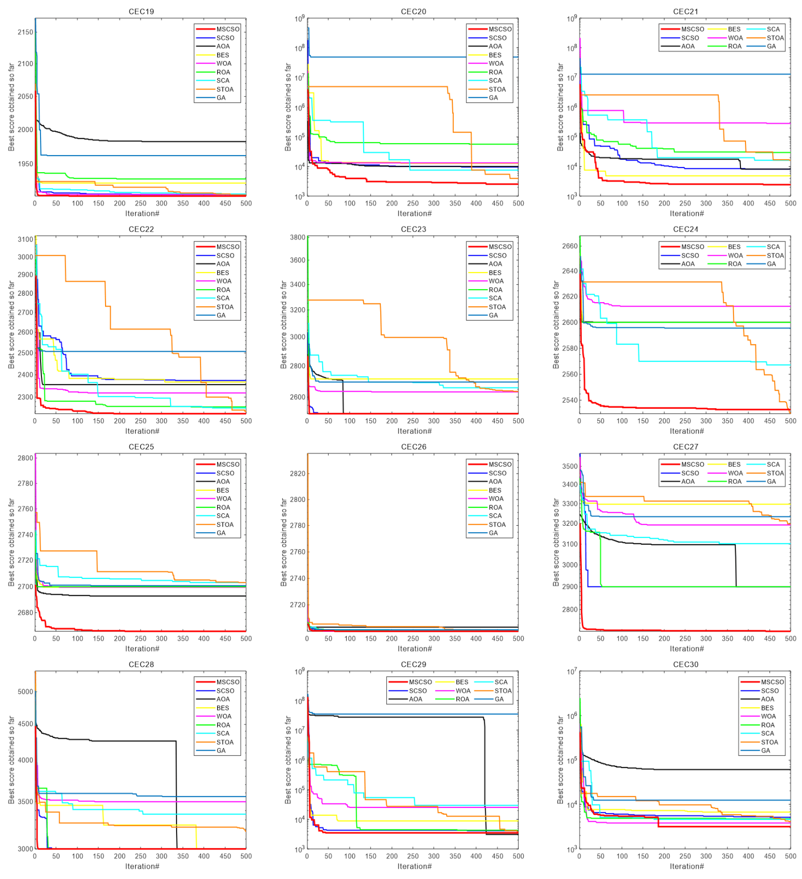


Figure 11. Convergence curve of the benchmark function optimization algorithm on CEC2014.

5.2.2. Analysis of Box Plot Results

A box chart is a statistical chart that uses five statistics in data: minimum, upper quartile, median, lower quartile, and maximum to describe data. The box chart's top- and bottom-line segments represent the data's maximum and minimum values, respectively. The upper and lower segments of the box chart represent the third quartile and the first quartile, respectively. The thick line in the middle of the box chart represents the median of the data. It can intuitively display the abnormal value of the data, the dispersion degree of distribution, and the symmetry of the data. Figure 12 is a block diagram obtained after thirty independent operations of nine algorithms. It can be seen that the MSCSO algorithm is very narrow and keeps the lowest point. Compared with the SCSO algorithm, the MSCSO algorithm can obtain low box graphs. Compared with GA, the MSCSO algorithm has a better optimization effect. Some of the box charts have little difference because it is easy to find a good value in the function, resulting in a small variance. In general, the box graph of the MSCSO algorithm has achieved better results.

5.2.3. Analysis of the Wilcoxon Rank Sum Test Results

The MSCSO algorithm has achieved good results in the CEC2014 benchmark function through the above analysis. Table 7 shows that the similarity between the MSCSO algorithm and the seven comparison algorithms is low, mostly less than 5%. However, Hybrid Function 1 and Composition Functions are partially greater than 5%. This means that the fitness values obtained by the eight comparison algorithms in these functions are not significantly different from those of the MSCSO algorithm. Many 1 of CEC23 and CEC28 occur, which means that the MSCSO algorithm achieves the same fitness values as these comparison algorithms. Some of the other functions are greater than 5%. This means that in these functions, the difference between the values obtained by the MSCSO algorithm and the comparison algorithm is not obvious, and the difference between the fitness values obtained by the MSCSO algorithm and the comparison algorithm is small. However, most of them are less than 5%, which indicates that the MSCSO algorithm differs significantly from the comparison algorithm in most functions.

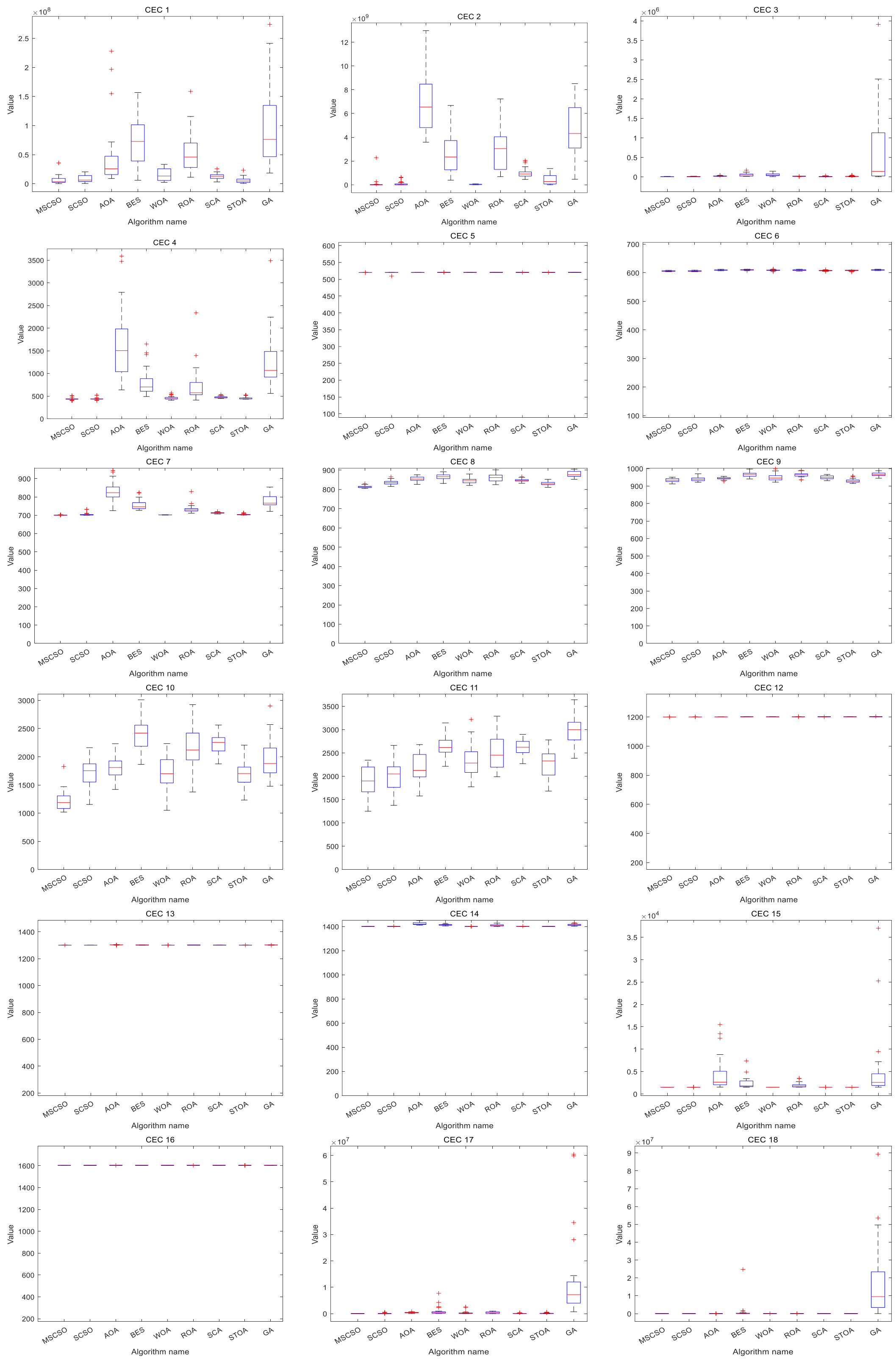


Figure 12. *Cont.*

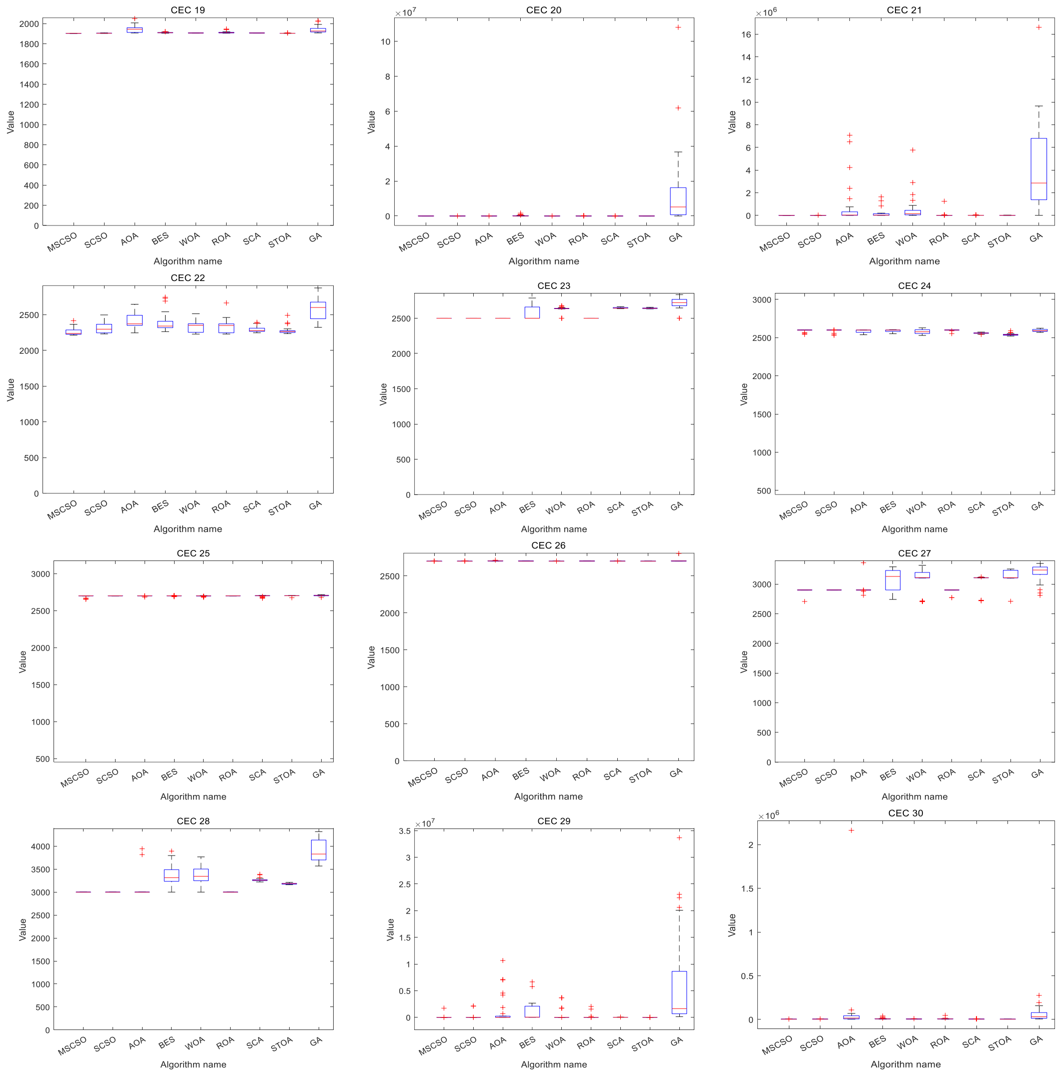


Figure 12. Convergence curve of the benchmark function optimization algorithm on CEC2014.

Table 7. CEC2014 Experimental Results of the Wilcoxon Rank Sum Test on Benchmark Functions.

CEC	SCSO vs. MSCSO	AOA vs. MSCSO	BES vs. MSCSO	WOA vs. MSCSO	ROA vs. MSCSO	SCA vs. MSCSO	STOA vs. MSCSO	GA vs. MSCSO
CEC 1	2.84×10^{-5}	1.73×10^{-6}	1.73×10^{-6}	7.51×10^{-5}	1.8×10^{-5}	6.89×10^{-5}	4.28×10^{-2}	1.73×10^{-6}
CEC 2	9.63×10^{-4}	1.73×10^{-6}	1.92×10^{-6}	1.25×10^{-4}	1.73×10^{-6}	2.6×10^{-6}	2.6×10^{-5}	1.92×10^{-6}
CEC 3	4.45×10^{-5}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	2.16×10^{-5}	2.35×10^{-6}	2.88×10^{-6}	1.73×10^{-6}
CEC 4	$\mathbf{8.61 \times 10^{-1}}$	1.73×10^{-6}	1.73×10^{-6}	4.2×10^{-4}	6.34×10^{-6}	2.6×10^{-6}	1.49×10^{-5}	1.73×10^{-6}
CEC 5	1.71×10^{-3}	5.22×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.92×10^{-6}
CEC 6	1.66×10^{-2}	1.92×10^{-6}	1.73×10^{-6}	4.29×10^{-6}	9.71×10^{-5}	6.34×10^{-6}	1.13×10^{-5}	1.73×10^{-6}
CEC 7	8.19×10^{-5}	1.73×10^{-6}	1.73×10^{-6}	1.13×10^{-5}	2.6×10^{-6}	1.73×10^{-6}	1.92×10^{-6}	1.73×10^{-6}
CEC 8	9.32×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	2.13×10^{-6}	1.73×10^{-6}	2.41×10^{-4}	1.73×10^{-6}
CEC 9	2.18×10^{-2}	2.58×10^{-3}	1.73×10^{-6}	9.63×10^{-4}	7.71×10^{-4}	2.35×10^{-6}	9.27×10^{-3}	1.92×10^{-6}
CEC 10	2.35×10^{-6}	1.92×10^{-6}	1.73×10^{-6}	3.18×10^{-6}	3.88×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
CEC 11	4.49×10^{-2}	1.6×10^{-4}	3.18×10^{-6}	2.37×10^{-5}	9.71×10^{-5}	1.73×10^{-6}	2.84×10^{-5}	1.73×10^{-6}
CEC 12	3.88×10^{-4}	1.73×10^{-6}	1.92×10^{-6}	1.73×10^{-6}	1.92×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
CEC 13	3.87×10^{-2}	1.73×10^{-6}	1.92×10^{-6}	$\mathbf{5.45 \times 10^{-2}}$	3.68×10^{-2}	1.64×10^{-5}	8.31×10^{-4}	1.73×10^{-6}
CEC 14	$\mathbf{9.59 \times 10^{-1}}$	1.73×10^{-6}	1.73×10^{-6}	2.7×10^{-2}	2.07×10^{-2}	5.22×10^{-6}	$\mathbf{1.53 \times 10^{-1}}$	1.73×10^{-6}
CEC 15	1.04×10^{-3}	1.73×10^{-6}	1.73×10^{-6}	5.22×10^{-6}	5.75×10^{-6}	1.73×10^{-6}	7.51×10^{-5}	1.73×10^{-6}

Table 7. *Cont.*

CEC	SCSO vs. MSCSO	AOA vs. MSCSO	BES vs. MSCSO	WOA vs. MSCSO	ROA vs. MSCSO	SCA vs. MSCSO	STOA vs. MSCSO	GA vs. MSCSO
CEC 16	2.58×10^{-3}	2.88×10^{-6}	2.16×10^{-5}	7.69×10^{-6}	4.07×10^{-2}	1.92×10^{-6}	1.02×10^{-5}	1.73×10^{-6}
CEC 17	$\mathbf{7.66 \times 10^{-1}}$	3.52×10^{-6}	3.41×10^{-5}	1.36×10^{-5}	4.72×10^{-2}	3.11×10^{-5}	2.84×10^{-5}	1.73×10^{-6}
CEC 18	$\mathbf{3.82 \times 10^{-1}}$	$\mathbf{5.04 \times 10^{-1}}$	1.73×10^{-6}	$\mathbf{1.06 \times 10^{-1}}$	$\mathbf{1.31 \times 10^{-1}}$	8.92×10^{-5}	$\mathbf{1.02 \times 10^{-1}}$	1.92×10^{-6}
CEC 19	2.58×10^{-3}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	5.22×10^{-6}	1.73×10^{-6}	3.59×10^{-4}	1.73×10^{-6}
CEC 20	7.27×10^{-3}	1.29×10^{-3}	2.41×10^{-3}	7.51×10^{-5}	3.38×10^{-3}	6.42×10^{-3}	1.96×10^{-2}	1.73×10^{-6}
CEC 21	7.27×10^{-3}	1.02×10^{-5}	3.18×10^{-6}	1.92×10^{-6}	$\mathbf{6.87 \times 10^{-2}}$	1.85×10^{-2}	1.85×10^{-2}	1.73×10^{-6}
CEC 22	2.58×10^{-3}	3.88×10^{-6}	1.24×10^{-5}	6.16×10^{-4}	$\mathbf{5.45 \times 10^{-2}}$	2.18×10^{-2}	3.16×10^{-2}	1.73×10^{-6}
CEC 23	**1**	**1**	4.38×10^{-4}	8.3×10^{-6}	**1**	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
CEC 24	3.34×10^{-4}	1.81×10^{-2}	$\mathbf{5.2 \times 10^{-1}}$	$\mathbf{3.09 \times 10^{-1}}$	4.69×10^{-2}	4.45×10^{-5}	3.52×10^{-6}	$\mathbf{8.29 \times 10^{-1}}$
CEC 25	$\mathbf{5 \times 10^{-1}}$	$\mathbf{6.25 \times 10^{-1}}$	$\mathbf{1.34 \times 10^{-1}}$	$\mathbf{8.2 \times 10^{-1}}$	1.56×10^{-2}	2.35×10^{-6}	1.73×10^{-6}	3.72×10^{-5}
CEC 26	$\mathbf{9.37 \times 10^{-2}}$	1.73×10^{-6}	2.13×10^{-6}	1.96×10^{-2}	4.53×10^{-4}	1.92×10^{-6}	4.49×10^{-2}	1.73×10^{-6}
CEC 27	$\mathbf{8.75 \times 10^{-1}}$	1.56×10^{-2}	9.15×10^{-5}	1.36×10^{-5}	$\mathbf{8.75 \times 10^{-1}}$	5.31×10^{-5}	1.73×10^{-6}	1.73×10^{-6}
CEC 28	**1**	**1**	3.79×10^{-6}	1.73×10^{-6}	**1**	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
CEC 29	7.73×10^{-3}	$\mathbf{8.94 \times 10^{-1}}$	2.13×10^{-6}	4.72×10^{-2}	4.68×10^{-3}	3.11×10^{-5}	1.36×10^{-4}	1.73×10^{-6}
CEC 30	1.83×10^{-3}	2.56×10^{-6}	2.88×10^{-6}	5.22×10^{-6}	2.22×10^{-4}	3.06×10^{-4}	$\mathbf{9.43 \times 10^{-1}}$	1.73×10^{-6}

6. Constrained Engineering Design Problems

In the fifth part, the optimization performance of the MSCSO algorithm is verified to verify the practical effect of the MSCSO algorithm in engineering problems. In this paper, seven engineering problems are selected for testing. The specific experimental results are as follows.

6.1. Pressure Vessel Design Problem

The purpose of pressure vessel design is to minimize the total cost of a cylinder-shaped pressure vessel. The schematic diagram of the pressure vessel is shown in Figure 13. The variables in question are shell thickness T_s, head thickness T_h, inner radius R, and vessel length L. The minimum cost of the pressure vessel is obtained through constraints.

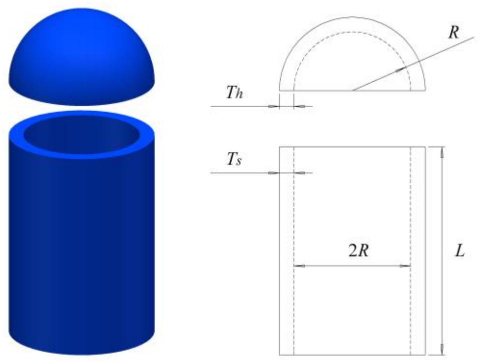


Figure 13. Model of the pressure vessel design.

Consider:

$$\vec{x} = [x_1 \quad x_2 \quad x_3 \quad x_4] = [T_s \quad T_h \quad R \quad L] \tag{20}$$

Objective function:

$$f\left(\vec{x}\right) = 0.6224x_1x_2x_3 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \tag{21}$$

Subject to:

$$g_1(\vec{x}) = -x_1 + 0.0193x_3 \le 0 \tag{22}$$

$$g_2(\vec{x}) = -x_3 + 0.00954x_3 \le 0 \tag{23}$$

$$g_3(\vec{x}) = -\pi x_3^2 x_4 + \frac{4}{3}\pi x_3^3 + 1{,}296{,}000 \leq 0 \tag{24}$$

$$g_4(\vec{x}) = -x_4 - 240 \leq 0 \tag{25}$$

Variable range:

$$0 \leq x_1 \leq 99, 0 \leq x_2 \leq 99, 10 \leq x_3 \leq 200, 10 \leq x_4 \leq 200 \tag{26}$$

The results of pressure vessel design problems are shown in Table 8. It shows that the MSCSO algorithm has a good effect in solving the engineering problem. As can be seen in the table, the MSCSO algorithm obtains T_s = 0.742406, T_h = 0.370292, R = 40.31962, and L = 200, resulting in the minimum cost of 5734.915. Among the other comparison algorithms, eight have achieved cost values greater than 6000, and four have less than 6000. The resulting costs are greater than those of the MSCSO algorithm.

Table 8. Experimental results of the pressure vessel design.

Algorithm	T_s	T_h	R	L	Best Cost
MSCSO	**0.742406**	**0.370292**	**40.31962**	**200**	**5734.915**
MGTOA [43]	0.754364	0.366375	40.42809	198.5652	5752.402458
CPSO [44]	0.8125	0.4375	42.0913	176.7465	6061.0777
HPSO [45]	0.8125	0.4375	42.0984	176.6366	6059.7143
GWO [3]	0.8125	0.4345	42.08918	176.7587	6059.5639
CS [46]	0.8125	0.4375	42.09845	176.6366	6059.714335
AO [47]	1.054	0.182806	59.6219	39.805	5949.2258
EROA [48]	0.84343	0.400762	44.786	145.9578	5935.7301
WOA [6]	0.8125	0.4375	42.09827	176.639	6059.741
GA [8]	0.8125	0.4375	42.0974	176.6541	6059.94634
MVO [16]	0.8125	0.4375	42.09074	176.7387	6060.8066
ACO [2]	0.8125	0.4375	42.10362	176.5727	6059.0888

6.2. Speed Reducer Design Problem

The goal of the speed reducer design is to find the minimum mass of the reducer to meet four design constraints: bending stress of gear teeth, covering stress, lateral deflection of shaft, and stress in the shaft. This problem has seven variables, namely the width of the tooth surface x_1, the gear module x_2, the number of teeth on the pinion x_3, the length of the first shaft between bearings x_4, the length of the second shaft between bearings x_5 the diameter of the first shaft x_6, and the diameter of the second shaft x_7. The schematic diagram of variables is shown in Figure 14.

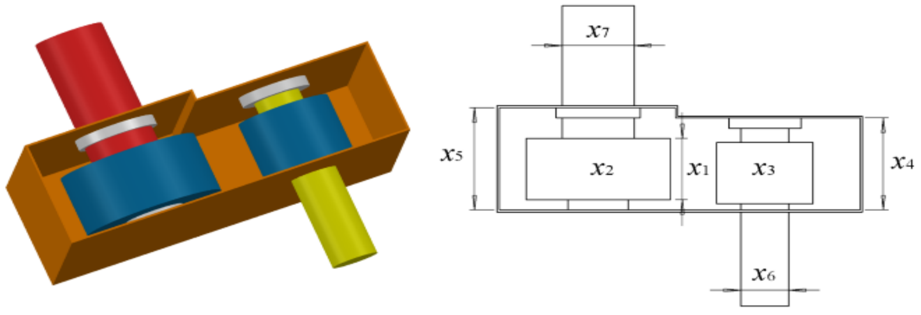


Figure 14. Model of the speed reducer design.

The mathematical formulation of this problem is shown below:
Consider:

$$x = [x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7] \tag{27}$$

Objective function:

$$f(\vec{x}) = 07854 \times x_1 \times {x_2}^2 \times (3.3333 \times {x_3}^2 + 14.9934 \times x_3 - 43.0934) - 1.508 \times x_1 \times ({x_6}^2 + {x_7}^2) + 7.4777 \times {x_6}^3 + {x_7}^3 + 0.7854 \times x_4 \times {x_6}^2 + x_5 \times {x_7}^2 \tag{28}$$

Subject to:

$$g_1(\vec{x}) = \frac{27}{x_1 \times {x_2}^2 \times x_3} - 1 \le 0 \tag{29}$$

$$g_2(\vec{x}) = \frac{397.5}{x_1 \times {x_2}^2 \times {x_3}^2} - 1 \le 0 \tag{30}$$

$$g_3(\vec{x}) = \frac{1.93 \times {x_4}^3}{x_2 \times x_3 \times {x_6}^4} - 1 \le 0 \tag{31}$$

$$g_4(\vec{x}) = \frac{1.93 \times {x_5}^3}{x_2 \times x_3 \times {x_7}^4} - 1 \le 0 \tag{32}$$

$$g_5(\vec{x}) = \frac{1}{110 \times {x_6}^3} \times \sqrt{(\frac{745 \times x_4}{x_2 \times x_3})^2 + 16.9 \times 10^6} - 1 \le 0 \tag{33}$$

$$g_6(\vec{x}) = \frac{1}{85 \times {x_7}^3} \times \sqrt{(\frac{745 \times x_5}{x_2 \times x_3})^2 + 16.9 \times 10^6} - 1 \le 0 \tag{34}$$

$$g_7(\vec{x}) = \frac{x_2 \times x_3}{40} - 1 \le 0 \tag{35}$$

$$g_8(\vec{x}) = \frac{5 \times x_2}{x_1} - 1 \le 0 \tag{36}$$

$$g_9(\vec{x}) = \frac{x_1}{12 \times x_2} - 1 \le 0 \tag{37}$$

$$g_{10}(\vec{x}) = \frac{1.5 \times x_6 + 1.9}{x_4} - 1 \le 0 \tag{38}$$

$$g_{11}(\vec{x}) = \frac{1.1 \times x_7 + 1.9}{x_5} - 1 \le 0 \tag{39}$$

Boundaries:

$$2.6 \le x_1 \le 3.6, 0.7 \le x_2 \le 0.8, 17 \le x_3 \le 28, 7.3 \le x_4 \le 8.3, 7.3 \le x_5 \le 8.3, 2.9 \le x_6 \le 3.9, 5 \le x_7 \le 5.5 \tag{40}$$

In Table 9, the MSCSO algorithm finally obtained a weight of 2995.438. The first one is obtained in the comparison algorithm. Compared with other algorithms, it has particular improvement.

Table 9. Experimental results of the speed reducer design.

Algorithm	Optimal Values for Variables							Optimal Weight
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	
MSCSO	**3.497592**	**0.7**	**17**	**7.3**	**7.8**	**3.350043**	**5.285504**	**2995.438**
AOA [40]	3.50384	0.7	17	7.3	7.72933	3.35649	5.2867	2997.9157
MFO [7]	3.497455	0.7	17	7.82775	7.712457	3.351787	5.286352	2998.94083
CS [46]	3.5015	0.7	17	7.605	7.8181	3.352	5.2875	3000.981
RSA [49]	3.50279	0.7	17	7.30812	7.74715	3.35067	5.28675	2996.5157
HS [23]	3.520124	0.7	17	8.37	7.8	3.36697	5.288719	3029.002

6.3. Welded Beam Design Problem

The design problem of the welded beam is to minimize the cost of the welded beam under four decision variables and seven constraints. This problem has four variables: weld width h, connecting beam length l, beam height t, and connecting beam thickness b. See Figure 15 for details.

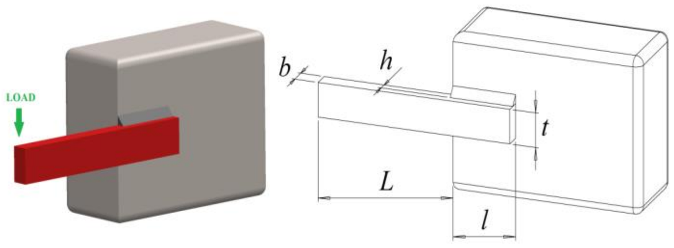


Figure 15. Model of the welded beam design.

The mathematical formulation of this problem is shown below:
Consider:

$$x = [x_1\ x_2\ x_3\ x_4] = [h\ l\ t\ b] \tag{41}$$

Objective function:

$$f(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2) \tag{42}$$

Subject to:

$$g_1\left(\vec{x}\right) = \tau\left(\vec{x}\right) - \tau_{\max} \leq 0 \tag{43}$$

$$g_2\left(\vec{x}\right) = \sigma\left(\vec{x}\right) - \sigma_{\max} \leq 0 \tag{44}$$

$$g_3\left(\vec{x}\right) = \delta\left(\vec{x}\right) - \delta_{\max} \leq 0 \tag{45}$$

$$g_4\left(\vec{x}\right) = x_1 - x_4 \leq 0 \tag{46}$$

$$g_5\left(\vec{x}\right) = P - P_c\left(\vec{x}\right) \leq 0 \tag{47}$$

$$g_6\left(\vec{x}\right) = 0.125 - x_1 \leq 0 \tag{48}$$

$$g_7\left(\vec{x}\right) = 1.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 0.5 \leq 0 \tag{49}$$

where:

$$\tau\left(\vec{x}\right) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')}, \tau' = \frac{P}{\sqrt{2x_1x_2}}, \tau'' = \frac{MR}{J}, \tag{50}$$

$$M = P\left(L + \frac{x_2}{2}\right), R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}, \sigma\left(\vec{x}\right) = \frac{6PL}{x_4x_3^2}, \tag{51}$$

$$J = 2\left\{\sqrt{2x_1x_2}\left[\frac{x_x^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\}, \delta\left(\vec{x}\right) = \frac{6PL^3}{Ex_4x_3^2}, \tag{52}$$

$$P_c\left(\vec{x}\right) = \frac{\sqrt[4.013E]{\frac{x_3^2x_4^6}{0}}}{L^2}, \left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right), \left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right), \tag{53}$$

$$P = 6000lb, L = 14\ \ in, \delta_{\max} = 0.25\text{in}, E = 30 \times 10^6\ \ psi, \tag{54}$$

$$\tau_{\max} = 13,600\ \ psi, and\ \ \sigma_{\max} = 30,000\ \ \ psi \tag{55}$$

Boundaries:

$$0.1 \leq x_i \leq 2, i = 1, 4; 0.1 \leq x_i \leq 10, i = 2.3 \quad (56)$$

The results of the welded beam design problems are shown in Table 10. The weld width *h* = 0.205723, connecting beam length *l* = 3.253494, beam height *t* = 9.036686, and connecting beam thickness *b* = 0.205731 obtained by the MSCSO algorithm. Compared with other algorithms, the MSCSO algorithm obtains the minimum weight. The final weight is 1.695309.

Table 10. Experimental results of the welded beam design.

Algorithm	*h*	*l*	*t*	*b*	Best Weight
MSCSO	**0.205723**	**3.253494**	**9.036686**	**0.205731**	**1.695309**
TSA [50]	0.244157	6.223066	8.29555	0.244405	2.38241101
WOA [6]	0.20536	3.48293	9.03746	0.206276	1.730499
ROA [4]	0.200077	3.365754	9.011182	0.206893	1.706447
GWO [3]	0.205676	3.478377	9.03681	0.205778	1.72624
GA [8]	0.1829	4.0483	9.3666	0.2059	1.8242
MFO [7]	0.2057	3.4703	9.0364	0.2057	1.72452
MVO [16]	0.205463	3.473193	9.044502	0.205695	1.72645
GSA [17]	0.182129	3.856979	10	0.202376	1.879952
RO [20]	0.203687	3.528467	9.004233	0.207241	1.735344
MROA [51]	0.2062185	3.254893	9.020003	0.206489	1.699058

6.4. Tension/Compression Spring Design Problem

The tension/compression spring design's purpose is to reduce the spring's mass through three variables and four constraints. Constraints include minimum deviation (g_1), shear stress (g_2), impact frequency (g_3), and outer diameter limit (g_4). The corresponding variables include wire diameter *d*, average coil diameter *D*, and effective coil number *N*. f(x) is the minimum spring mass. See Figure 16 for details.

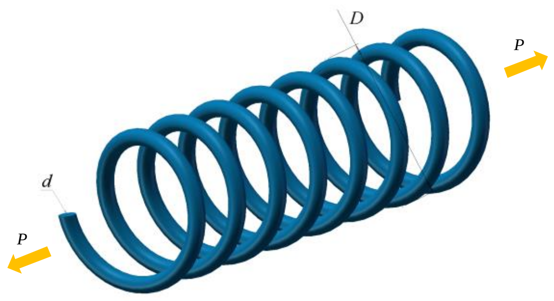


Figure 16. Model of the tension/compression spring design.

The mathematical formulation of this problem is shown below:
Consider:

$$x = [x_1\ x_2\ x_3] = [d\ D\ N] \quad (57)$$

Objective function:

$$f(x) = (x_3 + 2) \times x_2 \times x_1^2 \quad (58)$$

Subject to:

$$g_1(x) = 1 - \frac{x_3 \times x_2^3}{71,785 \times x_1^4} \leq 0 \quad (59)$$

$$g_2(x) = \frac{4 \times x_2^2 - x_1 \times x_2}{12,566 \times x_1^4} + \frac{1}{5108 \times x_1^2} - 1 \leq 0 \quad (60)$$

$$g_3(x) = 1 - \frac{140.45 \times x_1}{x_2^2 \times x_3} \leq 0 \quad (61)$$

$$g_4(x) = \frac{x_1 + x_2}{1.5} - 1 \leq 0 \tag{62}$$

Boundaries:

$$\begin{aligned} &0.05 \leq x_1 \leq 2.0; 0.25 \leq x_2 \leq 1.3; \\ &2.0 \leq x_3 \leq 15.0 \end{aligned} \tag{63}$$

As can be seen in Table 11, the weight obtained by each algorithm is relatively small. This extensively tests the accuracy of the algorithm in solving engineering problems. The MSCSO algorithm achieves a minimum weight of 0.009872 among these algorithms. It shows that the MSCSO algorithm is more accurate in solving the engineering problem.

Table 11. Experimental results of the tension/compression spring design.

Algorithm	*d*	*D*	*V*	Best Weight
MSCSO	**0.05**	**0.374433**	**8.546579**	**0.009872**
MFO [7]	0.051994	0.364109	10.86842	0.012667
SSA [33]	0.051207	0.345215	12.00403	0.012676
ES [52]	0.051989	0.363965	10.89052	0.012681
PSO [1]	0.051728	0.357644	11.24454	0.012675
EROA [48]	0.053799	0.46951	5.811	0.010614
HHO [53]	0.051796	0.359305	11.13886	0.012665
HS [23]	0.051154	0.349871	12.07643	0.012671
MVO [16]	0.05251	0.37602	10.33513	0.01279
GA [8]	0.05148	0.351661	11.6322	0.012705
GWO [3]	0.05169	0.356737	11.28885	0.012666
DE [13]	0.051609	0.354714	11.41083	0.01267

6.5. Cantilever Beam Design Problem

The optimization purpose of the cantilever beam design is to minimize the weight of the cantilever, given the following decision variables: the height or width of five hollow square blocks with constant thickness. The model of the cantilever beam is shown in Figure 17.

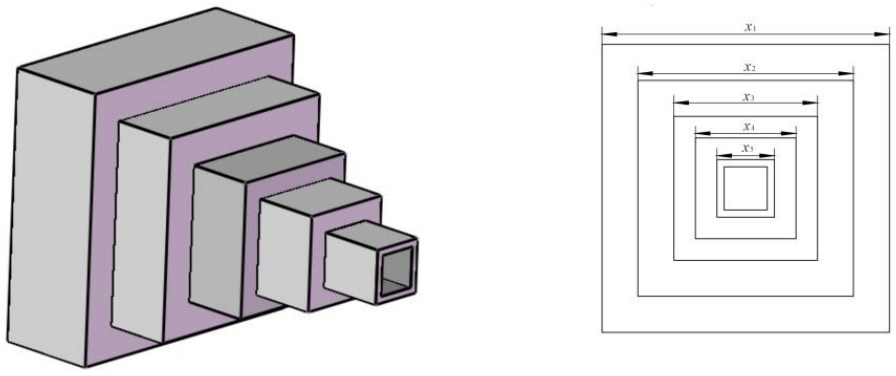


Figure 17. Model of the cantilever beam design.

The mathematical formulation of this problem is shown below:
Consider:

$$x = [x_1\ x_2\ x_3\ x_4\ x_5] \tag{64}$$

Objective function:

$$f(x) = 0.0624(x_1 + x_2 + x_3 + x_4 + x_5) \tag{65}$$

Subject to:

$$g(x) = \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \leq 0 \tag{66}$$

Boundaries:

$$0.01 \leq x_i \leq 100(i = 1, 2, \cdots 5) \tag{67}$$

The statistical table of the cantilever beam design is shown in Table 12. The $x_i(i = 1, 2, \cdots, 5)$ obtained by the MSCSO algorithm decreases gradually, which conforms to the design of the cantilever beam and, finally, a minimum weight of 1.33995853466334 is obtained. Compared with the data of other algorithms, the data obtained by the MSCSO algorithm are more consistent with the characteristics of the engineering problem.

Table 12. Experimental results of the cantilever beam design.

Algorithm	Optimal Values for Variables					Optimum Weight
	x_1	x_2	x_3	x_4	x_5	
MSCSO	**6.01265**	**5.315452**	**4.492016**	**3.501096**	**2.152481**	**1.33995853466334**
WOA [6]	5.1261	5.6188	5.0952	3.9329	2.3219	1.37873150673956
BWO [54]	6.2094	6.2094	6.2094	6.2094	6.2094	1.93736251728534
PSO [1]	6.0040	5.2950	4.4915	3.5125	2.1710	1.33998298081255
GSA [17]	5.6052	4.9553	5.6619	3.1959	3.2026	1.41155753917296
ERHHO [55]	6.0509	5.2639	4.514	3.4605	2.1878	1.3402

6.6. Multiple Disc Clutch Brake Problem

The purpose of the multiple disc clutch brake problem is to find five related variable values of the minimum mass multi-plate brake under eight constraints. The five variables are inner radius r_i, outer radius r_o, brake disc thickness t, driving force F, and surface friction number Z. The specific model is shown in Figure 18.

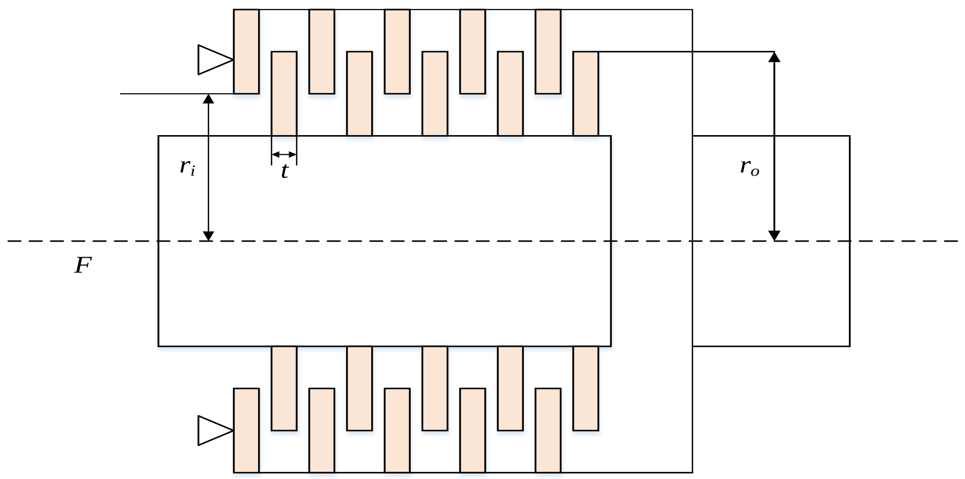


Figure 18. Model of the multiple disc clutch brake.

The mathematical formulation of this problem is shown below:
Consider:

$$x = [x_1 \quad x_2 \quad x_3 \quad x_4 \ \ x_5] = [r_i \quad r_o \quad t \ \ F \ \ Z] \tag{68}$$

Objective function:

$$f(x) = \Pi\left(r_o^2 - r_i^2\right)t(Z+1)\rho \ (\rho = 0.0000078) \tag{69}$$

Subject to:

$$g_1(x) = r_o - r_i - \Delta r \geq 0 \tag{70}$$

$$g_2(x) = l_{max} - (Z+1)(t+\delta) \geq 0 \tag{71}$$

$$g_3(x) = P_{max} - P_{rz} \geq 0 \tag{72}$$

$$g_4(x) = P_{max} v_{sr\ max} - P_{rz} v_{sr} \geq 0 \tag{73}$$

$$g_5(x) = v_{sr\ max} - v_{sr} \geq 0 \tag{74}$$

$$g_6(x) = T_{max} - T \geq 0 \tag{75}$$

$$g_7(x) = M_h - sM_s \geq 0 \tag{76}$$

$$g_8(x) = T \geq 0 \tag{77}$$

Variable range:

$$\begin{gathered} 60 \leq x_1 \leq 80, 90 \leq x_2 \leq 110, 1 \leq x_3 \leq 3, \\ 600 \leq x_4 \leq 1000, 2 \leq x_5 \leq 9 \end{gathered} \tag{78}$$

Other parameters:

$$M_h = \frac{2}{3}\mu F Z \frac{r_o^3 - r_i^2}{r_o^2 - r_i^3}, P_{rz} = \frac{F}{\text{II}\left(r_o^2 - r_i^2\right)}, \tag{79}$$

$$v_{rz} = \frac{2\text{II}\left(r_o^3 - r_i^3\right)}{90\left(r_o^2 - r_i^2\right)}, T = \frac{I_z \text{II} \quad n}{30\left(M_h + M_f\right)} \tag{80}$$

$$\Delta r = 20 \text{ mm},\ I_z = 55 \text{ kgmm}^2,\ P_{max} = 1 \text{ MPa},\ F_{max} = 1000 \text{ N}, \tag{81}$$

$$T_{max} = 15 \text{ s},\ \mu = 0.5,\ s = 1.5,\ M_s = 40 \text{ Nm},\ M_f = 3 \text{ Nm}, \tag{82}$$

$$n = 250 \text{ rpm},\ v_{sr\ max} = 10 \text{ m/s}, l_{max} = 30 \text{ mm} \tag{83}$$

In Table 13, the weight obtained by the MSCSO algorithm is 0.235242. Compared with other algorithms, the first algorithm is obtained. Other algorithms also have some effect, but the weight obtained is more excellent. It is proved that the MSCSO algorithm has a good effect on this problem.

Table 13. Experimental results of the multiple disc clutch brake.

Algorithm	Optimal Values for Variables					Optimum Weight
	x_1	x_2	x_3	x_4	x_5	
MSCSO	**70**	**90**	**1**	**637.791**	**2**	**0.235242**
TLBO [21]	70	90	1	810	3	0.313656611
WCA [56]	70	90	1	910	3	0.313656
MVO [16]	70	90	1	910	3	0.313656
CMVO [57]	70	90	1	910	3	0.313656
MFO [7]	70	90	1	910	3	0.313656
RSA [49]	70.0347	90.0349	1	801.7285	2.974	0.31176

6.7. Car Crashworthiness Design Problem

This problem also belongs to a minimal problem with eleven variables, subject to ten constraints. Figure 19 shows the finite element model of this problem. The decision variables are, respectively, the internal thickness of the B-pillar, the thickness of B-pillar reinforcement, the internal thickness of the floor, the thickness of the cross beam, the thickness of the door beam, the thickness of the door belt line reinforcement, the thickness of the roof longitudinal beam, the internal material of the B-pillar, the internal material of the floor, the height of the obstacle, and the impact position of the obstacle. The constraints are, respectively, the abdominal load, the upper viscosity standard, the middle viscosity standard, the lower viscosity standard, the upper rib deflection, the middle rib deflection, the lower rib deflection pubic symphysis force, B-pillar midpoint speed, and B-pillar front door speed.

Figure 19. The car crashworthiness design.

The mathematical formulation of this problem is shown below:
Minimize:

$$f(\vec{x}) = \text{Weight}, \tag{84}$$

Subject to:

$$g_1(\vec{x}) = F_a(\text{load in abdomen}) \le 1 \text{ kN}, \tag{85}$$

$$g_2(\vec{x}) = V \times Cu(\text{dummy upper chest}) \le 0.32 \text{ m/s}, \tag{86}$$

$$g_3(\vec{x}) = V \times Cm(\text{dummy middle chest}) \le 0.32 \text{ m/s}, \tag{87}$$

$$g_4(\vec{x}) = V \times Cl(\text{dummy lower chest}) \le 0.32 \text{ m/s}, \tag{88}$$

$$g_5(\vec{x}) = \Delta_{\text{ur}}(\text{upper rib deflection}) \le 32 \text{ mm}, \tag{89}$$

$$g_6(\vec{x}) = \Delta_{\text{mr}}(\text{middle rib deflection}) \le 32 \text{ mm}, \tag{90}$$

$$g_7(\vec{x}) = \Delta_{\text{lr}}(\text{lower rib deflection}) \le 32 \text{ mm}, \tag{91}$$

$$g_8(\vec{x}) = F(\text{Public force})_p \le 4 \text{ kN}, \tag{92}$$

$$\begin{aligned} &g_9(\vec{x}) = V_{\text{MBP}}(\text{Velocity of V}- \\ &\text{Pillar at middle point}) \le 9.9 \text{ mm/ms}, \end{aligned} \tag{93}$$

$$\begin{aligned} &g_{10}(\vec{x}) = V_{\text{FD}}(\text{Velocity of front door at V}- \\ &\text{Pillar}) \le 15.7 \text{ mm/ms}, \end{aligned} \tag{94}$$

Variable Range:

$$0.5 \le x_1 - x_7 \le 1.5, x_8, x_9 \in (0.192, 0.345), -30 \le x_{10}, x_{11} \le 30, \tag{95}$$

Table 14 shows the statistical results of the car crash worthiness design problem. From the table data, it can be concluded that the MSCSO algorithm can obtain a better solution to this problem. The MSCSO algorithm can obtain more precise unknowns in variable solving.

Table 14. Experimental results of the car crashworthiness design.

Algorithm	MSCSO	ROA [4]	MPA [58]	ROLGWO [59]	HHOCM [60]	MALO [61]
x_1	**0.500111598**	0.5	0.5	0.501255	0.500164	0.5
x_2	**1.228268972**	1.22942	1.22823	1.245551	1.248612	1.2281
x_3	**0.500012764**	0.5	0.5	0.500046	0.659558	0.5
x_4	**1.202547678**	1.21197	1.2049	1.180254	1.098515	1.2126
x_5	**0.500193341**	0.5	0.5	0.500035	0.757989	0.5
x_6	**1.052807602**	1.37798	1.2393	1.16588	0.767268	1.308
x_7	**0.500029525**	0.50005	0.5	0.500088	0.500055	0.5
x_8	**0.34499308**	0.34489	0.34498	0.344895	0.343105	0.3449
x_9	**0.335951909**	0.19263	0.192	0.299583	0.192032	0.2804
x_{10}	**0.461176886**	0.62239	0.44035	3.59508	2.898805	0.4242
x_{11}	**1.050120991**	-	1.78504	2.29018	-	4.6565
Best Weight	**23.19085116**	23.23544	23.19982	23.22243	24.48358	23.2294

7. Conclusions

The sand cat swarm optimization algorithm (SCSO) is a recently proposed population intelligence optimization algorithm. The SCSO algorithm simulates the hunting process of sand cats. Each sand cat will gradually move close to its prey, but the SCSO algorithm has insufficient exploration ability in the later stage, and it is easy to fall into local optimization, leading to difficulties in the convergence of the algorithm. To solve this problem, this paper proposes a modified sand cat swarm optimization algorithm (MSCSO). The core of the MSCSO algorithm is to use the wandering strategy when sand cats are hunting. When searching for prey, in order to increase the search range of the sand cat group, the triangle walking (TW) strategy is used to further search for a better position in the search range. The TW strategy first calculates the distance from the prey, then selects the walking direction through the Roulette Wheel Selection, and finally obtains the walking step length. This method increases the exploration ability of the SCSO algorithm and makes the MSCSO algorithm more global. In order to find a better position when the sand cat attacks its prey, it walks through the Levy flight walking (LFW) strategy. After adding the wandering strategy, the global exploration ability of the SCSO algorithm is enhanced. Then, the lens alternative-based learning (LBOL) strategy is added to enhance the optimization effect of the SCSO algorithm. The following conclusions can be drawn from the results of experimental performance evaluation and statistical analysis:

- According to the experimental image analysis, the proposed TW, FLW, and LBOL enhance the global exploration ability of the MSCSO algorithm.
- According to the experimental statistics, the proposed TW, FLW, and LBOL enhance the optimization performance of the MSCSO algorithm and can find better solutions in most functions.
- In engineering problems, the MSCSO algorithm has obtained better solutions than many other algorithms. It is proved that MSCSO has a good effect in solving engineering problems.

The MSCSO algorithm has strong exploration ability and can jump out of local optimization to prevent premature convergence of the algorithm. However, the exploitation ability of the MSCSO algorithm is relatively reduced, and the algorithm is difficult to converge faster when finding a better location. However, compared with SCSO, it is greatly improved. In a future work, we will strengthen the exploitation capability of the MSCSO algorithm and apply it to UAV 3D path planning, text clustering, feature selection, scheduling in cloud computing, parameter estimation, image segmentation, intrusion detection, etc.

Author Contributions: Conceptualization, D.W. and H.R.; methodology, D.W.; software, H.J., H.R. and C.W.; validation, H.J. and D.W.; formal analysis, D.W., H.R. and C.W.; investigation, D.W. and H.J.; resources, Q.L. and D.W.; data curation, H.R. and C.W.; writing—original draft preparation, H.R. and D.W.; writing—review and editing, H.J. and L.A.; visualization, D.W. and H.J.; supervision, H.J.

and D.W.; funding acquisition, D.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by National Education Science Planning Key Topics of the Ministry of Education—"Research on the core quality of applied undergraduate teachers in the intelligent age" (DIA220374).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to thank the anonymous reviewers for helping us improve this paper's quality.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Fearn, T. Particle swarm optimization. *NIR News* **2014**, *25*, 27.
2. Dorigo, M.; Birattari, M.; Stutzle, T. Ant colony optimization. *IEEE Comput. Intell. Mag.* **2006**, *1*, 28–39. [CrossRef]
3. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
4. Jia, H.; Peng, X.; Lang, C. Remora optimization algorithm. *Expert Syst. Appl.* **2021**, *185*, 115665. [CrossRef]
5. Assiri, A.S.; Hussien, A.G.; Amin, M. Ant lion optimization: Variants, hybrids, and applications. *IEEE Access* **2020**, *8*, 77746–77764. [CrossRef]
6. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
7. Hussien, A.G.; Amin, M.; Abd El Aziz, M. A comprehensive review of moth-flame optimisation: Variants, hybrids, and applications. *J. Exp. Theor. Artif. Intell.* **2020**, *32*, 705–725. [CrossRef]
8. Holland, J.H. Genetic algorithms. *Sci. Am.* **1992**, *267*, 66–73. [CrossRef]
9. Banzhaf, W.; Koza, J.R. Genetic programming. *IEEE Intell. Syst.* **2000**, *15*, 74–84. [CrossRef]
10. Simon, D. Biogeography-based optimization. *IEEE Trans. Evol. Comput.* **2008**, *12*, 702–713. [CrossRef]
11. Jaderyan, M.; Khotanlou, H. Virulence Optimization Algorithm. *Appl. Soft. Comput.* **2016**, *43*, 596–618. [CrossRef]
12. Sinha, N.; Chakrabarti, R.; Chattopadhyay, P. Evolutionary programming techniques for economic load dispatch. *IEEE Trans. Evol. Comput.* **2003**, *7*, 83–94. [CrossRef]
13. Storn, R.; Price, K. Differential Evolution-A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
14. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by Simulated Annealing. *Science* **1983**, *220*, 671–680. [CrossRef]
15. Mirjalili, S. SCA: A Sine Cosine Algorithm for Solving Optimization Problems. *Knowl.-Based Syst.* **2016**, *96*, 120–133. [CrossRef]
16. Mirjalili, S.; Mirjalili, S.M.; Hatamlou, A. Multi-verse optimizer: A nature-inspired algorithm for global optimization. *Neural Comput. Appl.* **2015**, *27*, 495–513. [CrossRef]
17. Rashedi, E.; Nezamabadi-Pour, H.S. GSA: A Gravitational Search Algorithm. *Inform. Sci.* **2009**, *179*, 2232–2248. [CrossRef]
18. Hatamlou, A. Black hole: A new heuristic optimization approach for data clustering. *Inform. Sci.* **2013**, *222*, 175–184. [CrossRef]
19. Kaveh, A.; Dadras, A. A novel meta-heuristic optimization algorithm: Thermal exchange optimization. *Adv. Eng. Softw.* **2017**, *110*, 69–84. [CrossRef]
20. Kaveh, A.; Khayatazad, M. A new meta-heuristic method: Ray optimization. *Comput. Struct.* **2012**, *112*, 283–294. [CrossRef]
21. Rao, R.V.; Savsani, V.J.; Vakharia, D.P. Teaching-Learning-Based Optimization: An optimization method for continuous non-linear large scale problems. *Inform. Sci.* **2012**, *183*, 1–15. [CrossRef]
22. Zhang, Y.; Jin, Z. Group teaching optimization algorithm: A novel metaheuristic method for solving global optimization problems. *Expert Syst. Appl.* **2020**, *148*, 113246. [CrossRef]
23. Geem, Z.W.; Kim, J.H.; Loganathan, G.V. A New Heuristic Optimization Algorithm: Harmony Search. *Simulation* **2001**, *2*, 60–68. [CrossRef]
24. Satapathy, S.; Naik, A. Social group optimization (SGO): A new population evolutionary optimization technique. *Complex Intell. Syst.* **2016**, *2*, 173–203. [CrossRef]
25. Naser, G.; Ebrahim, B. Exchange market algorith. *Appl. Soft. Comput.* **2014**, *19*, 177–187.
26. Seyyedabbasi, A.; Kiani, F. Sand Cat swarm optimization: A nature-inspired algorithm to solve global optimization problems. *Eng. Comput.* **2022**, 1–25. [CrossRef]
27. Li, Y.; Wang, G. Sand Cat Swarm Optimization Based on Stochastic Variation With Elite Collaboration. *IEEE Access* **2022**, *10*, 89989–90003. [CrossRef]
28. Jovanovic, D.; Marjanovic, M.; Antonijevic, M.; Zivkovic, M.; Budimirovic, N.; Bacanin, N. Feature Selection by Improved Sand Cat Swarm Optimizer for Intrusion Detection. In Proceedings of the 2022 International Conference on Artificial Intelligence in Everything (AIE), Lefkosa, Cyprus, 2–4 August 2022; pp. 685–690.

29. Roman, R.C.; Precup, R.E.; Petriu, E.M. Hybrid data-driven fuzzy active disturbance rejection control for tower crane systems. *Eur. J. Control* **2021**, *58*, 373–387. [CrossRef]
30. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
31. Chi, R.; Li, H.; Shen, D.; Hou, Z.; Huang, B. Enhanced P-type Control: Indirect Adaptive Learning from Set-point Updates. *IEEE Trans. Autom. Control* **2022**. [CrossRef]
32. Nadimi-Shahraki, M.H.; Taghian, S.; Mirjalili, S.; Faris, H. MTDE: An effective multi-trial vector-based differential evolution algorithm and its applications for engineering design problems. *Appl. Soft Comput.* **2020**, *97*, 106761. [CrossRef]
33. Hussien, A.G. An enhanced opposition-based salp swarm algorithm for global optimization and engineering problems. *J. Ambient. Intell. Humaniz. Comput.* **2022**, *13*, 129–150. [CrossRef]
34. Wang, Y.; Cai, Z.; Zhang, Q. Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Trans. Evol. Comput.* **2011**, *15*, 55–66. [CrossRef]
35. Zhang, H.; Wang, Z.; Chen, W.; Heidari, A.A.; Wang, M.; Zhao, X.; Liang, G.; Chen, H.; Zhang, X. Ensemble mutation-driven salp swarm algorithm with restart mechanism: Framework and fundamental analysis. *Expert Syst. Appl.* **2021**, *165*, 113897. [CrossRef]
36. Nadimi-Shahraki, M.H.; Taghian, S.; Mirjalili, S. An improved grey wolf optimizer for solving engineering problems. *Expert Syst. Appl.* **2021**, *166*, 113917. [CrossRef]
37. Zheng, R.; Jia, H.; Abualigah, L.; Wang, S.; Wu, D. An improved remora optimization algorithm with autonomous foraging mechanism for global optimization problems. *Math. Biosci. Eng.* **2022**, *19*, 3994–4037. [CrossRef]
38. Nadimi-Shahraki, M.H.; Taghian, S.; Mirjalili, S.; Ewees, A.A.; Abualigah, L.; Abd Elaziz, M. MTV-MFO: Multi-Trial Vector-Based Moth-Flame Optimization Algorithm. *Symmetry* **2021**, *13*, 2388. [CrossRef]
39. Liu, Q.; Li, N.; Jia, H.; Qi, Q.; Abualigah, L. Modified remora optimization algorithm for global optimization and multilevel thresholding image segmentation. *Mathematics* **2022**, *10*, 1014. [CrossRef]
40. Abualigah, L.; Diabat, A.; Mirjalili, S.; Elaziz, M.A.; Gandomi, A.H. The arithmetic optimization algorithm. *Comput. Methods Appl. Mech. Eng.* **2021**, *376*, 113609. [CrossRef]
41. Alsattar, H.A.; Zaidan, A.A.; Zaidan, B.B. Novel meta-heuristic bald eagle search optimisation algorithm. *Artif. Intell. Rev.* **2020**, *53*, 2237–2264. [CrossRef]
42. Dhiman, G.; Kaur, A. STOA: A bio-inspired based optimization algorithm for industrial engineering problems. *Eng. Appl. Artif. Intell.* **2019**, *82*, 148–174. [CrossRef]
43. Rao, H.; Jia, H.; Wu, D.; Wen, C.; Liu, Q.; Abualigah, L. A Modified Group Teaching Optimization Algorithm for Solving Constrained Engineering Optimization Problems. *Mathematics* **2022**, *10*, 3765. [CrossRef]
44. He, Q.; Wang, L. An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Eng. Appl. Artif. Intell.* **2007**, *20*, 89–99. [CrossRef]
45. He, Q.; Wang, L. A hybrid particle swarm optimization with a feasibilitybased rule for constrained optimization. *Appl. Math. Comput.* **2007**, *186*, 1407–1422.
46. Gandomi, A.H.; Yang, X.S.; Alavi, A.H. Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. *Eng. Comput.* **2013**, *29*, 17–35. [CrossRef]
47. Laith, A.; Dalia, Y.; Mohamed, A.E.; Ahmed, A.E.; Mohammed, A.A.A.; Amir, H.G. Aquila Optimizer: A novel meta-heuristic optimization algorithm. *Comput. Ind. Eng.* **2021**, *157*, 107250.
48. Wang, S.; Hussien, A.G.; Jia, H.; Abualigah, L.; Zheng, R. Enhanced Remora Optimization Algorithm for Solving Constrained Engineering Optimization Problems. *Mathematics* **2022**, *10*, 1696. [CrossRef]
49. Abualigah, L.; Elaziz, M.A.; Sumari, P.; Zong, W.G.; Gandomi, A.H. Reptile search algorithm (RSA): A nature-inspired meta-heuristic optimizer. *Expert Syst. Appl.* **2021**, *191*, 116158. [CrossRef]
50. Babalik, A.; Cinar, A.C.; Kiran, M.S. A modification of tree-seed algorithm using Deb's rules for constrained optimization. *Appl. Soft. Comput.* **2018**, *63*, 289–305. [CrossRef]
51. Wen, C.; Jia, H.; Wu, D.; Rao, H.; Li, S.; Liu, Q.; Abualigah, L. Modified Remora Optimization Algorithm with Multistrategies for Global Optimization Problem. *Mathematics* **2022**, *10*, 3604. [CrossRef]
52. Beyer, H.G.; Schwefel, H.P. Evolution strategies–A comprehensive introduction. *Nat. Comput.* **2002**, *1*, 3–52. [CrossRef]
53. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H. Harris hawks optimization: Algorithm and applications. *Future Gener. Comput. Syst.* **2019**, *97*, 849–872. [CrossRef]
54. Hayyolalam, V.; Kazem, A.A.P. Black widow optimization algorithm: A novel meta-heuristic approach for solving engineering optimization problems. *Eng. Appl. Artif. Intell.* **2020**, *87*, 103249. [CrossRef]
55. Song, M.; Jia, H.; Abualigah, L.; Liu, Q.; Lin, Z.; Wu, D.; Altalhi, M. Modified Harris Hawks Optimization Algorithm with Exploration Factor and Random Walk Strategy. *Comput. Intell. Neurosci.* **2022**, *2022*, 23. [CrossRef]
56. Eskandar, H.; Sadollah, A.; Bahreininejad, A.; Hamdi, M. Water cycle algorithm–a novel metaheuristic optimization method for solving constrained engineering optimization problems. *Comput. Struct.* **2012**, *110*, 151–166. [CrossRef]
57. Sayed, G.I.; Darwish, A.; Hassanien, A.E. A new chaotic multi-verse optimization algorithm for solving engineering optimization problems. *J. Exp. Theor. Artif. Intell.* **2018**, *30*, 293–317. [CrossRef]
58. Faramarzi, A.; Heidarinejad, M.; Mirjalili, S.; Gandomi, A.H. Marine predators algorithm: A nature-inspired metaheuristic. *Expert Syst. Appl.* **2020**, *152*, 113377. [CrossRef]

59. Long, W.; Jiao, J.; Liang, X.; Cai, S.; Xu, M. A random opposition-based learning grey wolf optimizer. *IEEE Access* **2019**, *7*, 113810–113825. [CrossRef]
60. Houssein, E.H.; Neggaz, N.; Hosney, M.E.; Mohamed, W.M.; Hassaballah, M. Enhanced Harris hawks optimization with genetic operators for selection chemical descriptors and compounds activities. *Neural Comput. Appl.* **2021**, *33*, 13601–13618. [CrossRef]
61. Wang, S.; Sun, K.; Zhang, W.; Jia, H. Multilevel thresholding using a modified ant lion optimizer with opposition-based learning for color image segmentation. *Math. Biosci. Eng.* **2021**, *18*, 3092–3143. [CrossRef]