mathematics

Article

Fault Prediction of Control Clusters Based on an Improved Arithmetic Optimization Algorithm and BP Neural Network

Tao Xu [1,2], Zeng Gao [1] and Yi Zhuang [1,*]

[1] The College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China; xutao16@nuaa.edu.cn (T.X.); gaozeng@nuaa.edu.cn (Z.G.)
[2] Jiangsu Automation Research Institute, Lianyungang 222061, China
* Correspondence: zy16@nuaa.edu.cn

Abstract: Higher accuracy in cluster failure prediction can ensure the long-term stable operation of cluster systems and effectively alleviate energy losses caused by system failures. Previous works have mostly employed BP neural networks (BPNNs) to predict system faults, but this approach suffers from reduced prediction accuracy due to the inappropriate initialization of weights and thresholds. To address these issues, this paper proposes an improved arithmetic optimization algorithm (AOA) to optimize the initial weights and thresholds in BPNNs. Specifically, we first introduced an improved AOA via multi-subpopulation and comprehensive learning strategies, called MCLAOA. This approach employed multi-subpopulations to effectively alleviate the poor global exploration performance caused by a single elite, and the comprehensive learning strategy enhanced the exploitation performance via information exchange among individuals. More importantly, a nonlinear strategy with a tangent function was designed to ensure a smooth balance and transition between exploration and exploitation. Secondly, the proposed MCLAOA was utilized to optimize the initial weights and thresholds of BPNNs in cluster fault prediction, which could enhance the accuracy of fault prediction models. Finally, the experimental results for 23 benchmark functions, CEC2020 benchmark problems, and two engineering examples demonstrated that the proposed MCLAOA outperformed other swarm intelligence algorithms. For the 23 benchmark functions, it improved the optimal solutions in 16 functions compared to the basic AOA. The proposed fault prediction model achieved comparable performance to other swarm-intelligence-based BPNN models. Compared to basic BPNNs and AOA-BPNNs, the MCLAOA-BPNN showed improvements of 2.0538 and 0.8762 in terms of mean absolute percentage error, respectively.

Keywords: arithmetic optimization algorithm; multi-subpopulation; comprehensive learning; BP neural network; failure prediction

MSC: 68T20; 90C26

check for updates

Citation: Xu, T.; Gao, Z.; Zhuang, Y. Fault Prediction of Control Clusters Based on an Improved Arithmetic Optimization Algorithm and BP Neural Network. *Mathematics* **2023**, *11*, 2891. https://doi.org/10.3390/math11132891

Academic Editors: Adrian Deaconu, Petru Adrian Cotfas and Daniel Tudor Cotfas

Received: 16 May 2023
Revised: 17 June 2023
Accepted: 22 June 2023
Published: 27 June 2023

1. Introduction

In recent years, due to the rapid development of computer technology, computer systems have been widely applied in various industries within the national economy, greatly promoting socioeconomic development. At the same time, higher requirements have been placed on the sustainable and stable operation of computer systems, and people are increasingly concerned about the availability of computer systems [1–5]. Currently, providing continuous and stable services in control cluster systems is an urgent issue in computer cluster technology.

Previous work has mostly focused on load balancing [6,7], resource scheduling [4,8], fault tolerance [9,10], and other aspects of cluster systems. However, recently, limited research has been conducted on improving system stability through fault prediction. Control cluster fault prediction aims to predict node failures at an early stage, enabling proactive resource scheduling and enhancing the availability of the cluster system. Pinto et al. [11] designed a distributed computing system for Hadoop clusters and used SVM for cluster fault prediction. Mukwevho et al. [12] analyzed three fault-tolerant techniques and achieved fault prediction through proactive methods. Das et al. [13] employed log analysis for fault prediction. However, most of these fault prediction approaches required building custom models according to specific requirements and were not widely applicable, and while neural-network-based fault prediction can be widely applied, it suffers from issues such as slow convergence and susceptibility to local optima due to sensitivity to initial weights and thresholds. Fortunately, swarm intelligence approaches can effectively adjust these parameters. The novel research field successfully combines machine learning and swarm intelligence approaches and proved to be able to obtain outstanding results in different areas [14,15]. Therefore, in this work, we design a novel intelligent algorithm and employ it to find the optimal parameters in the neural network prediction model.

The arithmetic optimization algorithm (AOA) is a new metaheuristic algorithm proposed by Abualigah et al. in 2021, which primarily utilizes basic arithmetic operators to perform exploration (multiplication and division) and exploitation (subtraction and addition) [16]. The algorithm's main advantages lie in its simplicity, ease of programming, and fewer parameters [17], which have led researchers to apply it in various fields, including engineering design [18–20], cloud computing [21], and image processing [22], to name a few [23–25]. However, the AOA faces challenges in dealing with complex optimization problems, particularly regarding issues of local optima and slow convergence. Recently, improved versions of the AOA have emerged as a trend. For example, Li et al. [18] employed 10 chaotic maps to modify the control parameters and improve the exploration and exploitation stages during the iteration process. However, this approach did not modify the mathematical model, which implies that it may still encounter local optima when faced with complex optimization problems. Çelik [19] employed information exchange [26,27], Gaussian distribution [26], and quasi-opposition [28,29] to propose an information-exchanged Gaussian AOA with quasi-opposition learning (IEGQO-AOA), which improved the convergence performance without significantly increasing the computational complexity of the algorithm. Gölcük et al. [23] employed highly disruptive polynomial mutation and local escaping operators to propose an improved AOA for training feedforward neural networks. These methods [19,23] employed mutation factors to improve the exploration performance of the AOA, enabling it to escape local optima. However, mutation factors may generate solutions that deviate from the optimal solution, thus reducing the convergence speed. Kaveh et al. [25] improved population diversity and convergence performance by modifying the mathematical model in the exploration and exploitation stage of the AOA and applied the improved AOA to structural optimization. Some research works have improved the performance of the AOA by combining it with other meta-heuristic algorithms, such as the sine cosine algorithm (SCA) [20], the salp swarm algorithm (SSA) [21], and the aquila optimizer (AO) [30]. It is worth noting that the aforementioned algorithms improved the global optimization performance by introducing mutation factors, modifying the control parameters, or incorporating other algorithms.

Swarm intelligence algorithms consist of two main phases, namely exploration and exploitation [31–34]. The purpose of exploration is to search the regions where the global optimum may exist. Exploitation aims to further refine and precisely search the promising regions identified during exploration. It is well known that the key to optimizing performance in swarm intelligence lies in the search capabilities of exploration and exploitation, as well as the balance and transition between these two phases. However, the AOA utilizes multiplication and division operators in the exploration phase and addition and subtraction operators in the exploitation phase, and it revolves around a single elite individual without involving information sharing among individuals. These limitations greatly restrict the exploration and exploitation performance of the algorithm. In addition, a linear mechanism does not accurately reflect the complex optimization process; therefore, it fails to facilitate a smooth transition from the exploration phase to the exploitation phase.

Motivated by the aforementioned analysis, we proposed a novel improved AOA via multi-subpopulation (MS) and comprehensive learning (CL) strategies for global optimization (MCLAOA). Subsequently, the MCLAOA was combined with a BP neural network (BPNN) to form the MCLAOA-BPNN model for cluster fault prediction. Firstly, we proposed the novel MCLAOA. Specifically, the MS was applied in the exploration phase, where we divided the population into several subpopulations, and each subpopulation revolved around its own sub-elite. This strategy alleviated the weakness of a single elite in terms of exploration performance and enhanced population diversity. The CL was used in the exploitation phase to increase the information sharing among individuals and sped up the convergence of the algorithm. In addition, to ensure a smooth transition from the exploration phase to the exploitation phase, a nonlinear math optimizer accelerated (MOA) with a tangent function was employed instead of the standard MOA. After obtaining the high-performance MCLAOA, we combined it with the BPNN to form the MCLAOA-BPNN cluster fault prediction model. The model utilized MCLAOA to obtain the best initial weights and thresholds for BPNN, thereby improving prediction accuracy and providing the foundation for resource scheduling and sustainable operation of cluster systems.

In this work, the main contributions are summarized as follows:

(1) To enhance the accuracy of cluster fault prediction, we attempted to design a new optimization algorithm and combined it with BPNN to form the MCLAOA-BPNN cluster fault prediction model. The model utilized MCLAOA to optimize the initial weights and thresholds of BPNN.
(2) To address the lack of individual information sharing in both the exploration and exploitation phases, we proposed the MCLAOA. This approach employed the MS and CL strategies to modify the mathematical models of the exploration and exploitation phases, thereby improving the optimization performance.
(3) To ensure a smooth transition from the exploration phase to the exploitation phase for the MCLAOA, we designed a nonlinear MOA with tangent functions to replace the linear mechanism in the standard AOA.
(4) Experimental results over 23 benchmark functions, CEC2020 benchmark problems, and two engineering examples showed that the proposed MCLAOA has much stronger merit. In addition, the MCLAOA-BPNN had better prediction accuracy compared to other algorithms.

The remainder of this paper is structured as follows: The standard AOA is introduced in Section 2, and the proposed MCLAOA is presented in Section 3. In Section 4, results and analysis of the proposed algorithm are presented using 23 benchmark functions, CEC2020 benchmark problems, and two engineering design problems. Section 5 presents the MCLAOA-BPNN model for cluster fault prediction and compares it with other models. At last, the conclusion of this paper is provided in Section 6.

2. Arithmetic Optimization Algorithm (AOA)

Inspired by the arithmetic operators, the AOA is proposed as a new intelligent algorithm. The basic principle of AOA is shown in Figure 1, which is mainly divided into the exploration phase and the exploitation phase [16].

2.1. Math Optimizer Accelerated (MOA)

Before optimization, the MOA is designed to determine whether the population is performing the exploration phase or the exploitation phase. Here, given a random number r_1 between 0 and 1, the global exploration phase is executed if $r_1 < MOA$, otherwise, the local exploitation phase is executed. The MOA can be formulated as:

$$MOA(t) = Min + t \times \left(\frac{Max - Min}{T}\right), \tag{1}$$

where t and T are the current iteration and the maximum number of iterations, respectively. Max and Min represent the maximum value and minimum value of the accelerated function.

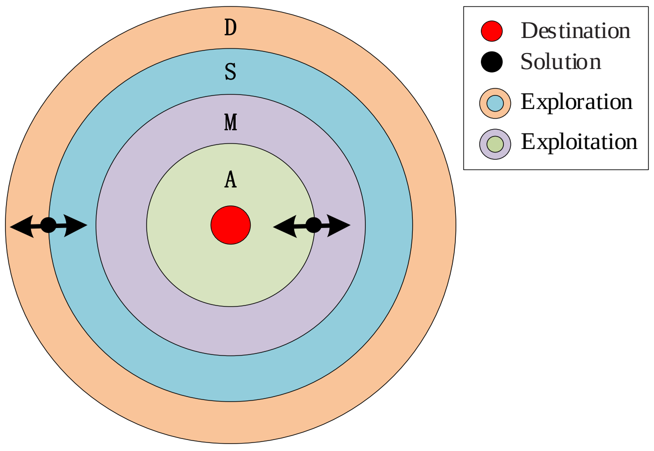

Figure 1. Updating toward or away from destination.

2.2. Exploration Phase

In this phase, the AOA mainly adopts division and multiplication search strategy to find better candidate solutions, and the mathematical model is as follows:

$$X_{i,j} = \begin{cases} best\left(X_j\right) \div (MOP + \partial) \times S_j, \ r_2 < 0.5, \\ best\left(X_j\right) \times MOP \times S_j, \ otherwise, \end{cases} \tag{2}$$

$$MOP(t) = 1 - \frac{t^{1/\alpha}}{T^{1/\alpha}}, \tag{3}$$

$$S_j = \left(UB_j - LB_j\right) \times \eta + LB_j, \tag{4}$$

where $X_{i,j}$ represents the position of the jth dimension of the ith individual, and $best\left(X_j\right)$ is the jth dimension in the best solution of all individuals. ∂ is a small integer number that prevents the denominator from being 0. $MOP(t)$ is a parameter representing the step size factor of the current iteration, S_j denotes the step size of the jth dimension, and r_2 is a random number in [0,1]. UB_j and LB_j represent the upper and lower bound value of the jth dimension, respectively. η is the control parameter that regulates the search process, and α is a sensitive parameter. η and α are set to 0.5 and 5, respectively, according to the literature [16].

2.3. Exploitation Phase

This phase performs the local exploitation. Additive and subtractive operators are adopted to search for the optimal solution. Specifically, given a random number r_3 between 0 and 1, if $r_3 < 0.5$, the subtractive operator is employed to search for the optimal solution, otherwise, the additive operator is employed to update the population position, which can be expressed as follows:

$$X_{i,j} = \begin{cases} best\left(X_j\right) - MOP \times S_j, \ r_3 < 0.5, \\ best\left(X_j\right) + MOP \times S_j, \ otherwise. \end{cases} \tag{5}$$

3. Proposed Method

From the mathematical model of AOA, it can be seen that all individuals perform expansion or reduction operations around the elite ($best$) in the exploration phase, which affects the search ability of the AOA. The exploitation phase employs a fixed step factor (S_j) without memory retention, which can lead to a lack of information exchange between individuals and reduce convergence effectiveness. In addition, the MOA with a linear mechanism cannot solve complex optimization problems. To deal with the above shortcomings, an improved AOA was proposed to solve the global optimization problem by MS strategy and CL strategy. Compared with the standard AOA, three operators, MS, CL and nonlinear MOA with tangent function, were introduced in this paper. The specific mathematical model is as follows.

3.1. Multi-Subpopulation (MS) Strategy

Global search refers to identifying the optimal region for the target within a larger search space to prevent the algorithm from getting trapped in local optima [35,36]. However, according to Equation (2), it is known that all individuals explore the search space based on the $best(X_j)$ and a fixed step size factor (S_j), which reduces population diversity and exploration performance. Therefore, we propose the MS strategy to improve the exploration performance of AOA, as shown in Figure 2.

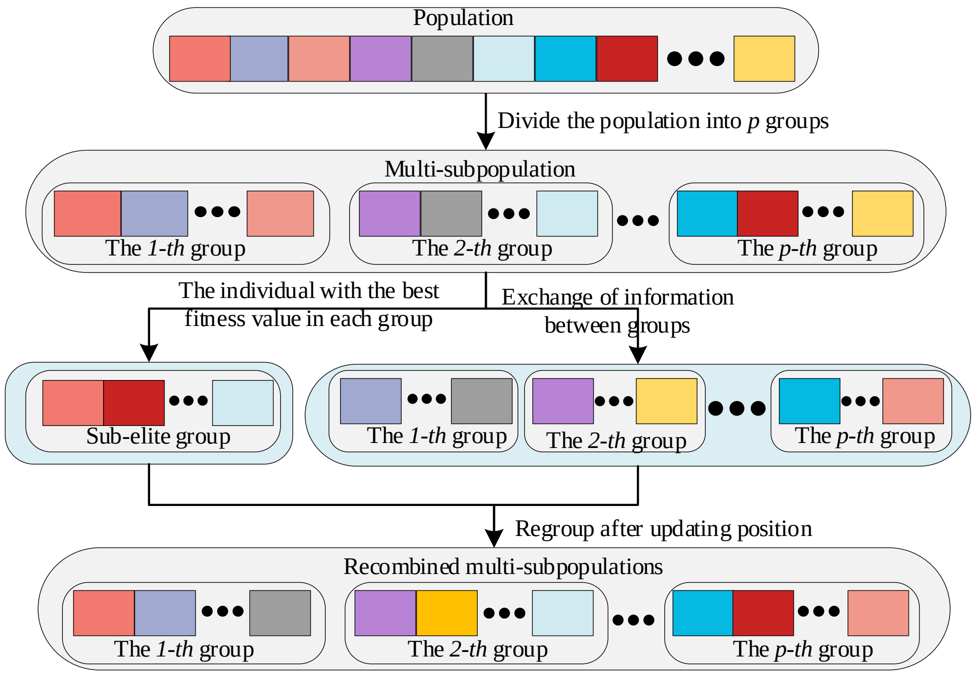
Population
Divide the population into p groups
Multi-subpopulation
The 1-th group
The 2-th group
The p-th group
The individual with the best fitness value in each group
Exchange of information between groups
Sub-elite group
The 1-th group
The 2-th group
The p-th group
Regroup after updating position
Recombined multi-subpopulations
The 1-th group
The 2-th group
The p-th group

Figure 2. The flowchart of the MS strategy.

To be specific, first a population size Np is given, which is divided into p subpopulations. Second, all individuals are evaluated for fitness, and the individual with the minimum fitness value in each group is selected as the sub-elite, forming a sub-elite group. Then, all individuals except for the sub-elite group are rearranged into p groups to explore the search space. Finally, each sub-elite is randomly assigned to different groups. Here, the sub-elite group $gbest$ can be expressed as Equation (6),

$$gbest = \begin{bmatrix} gbest^1 \\ gbest^2 \\ \vdots \\ gbest^p \end{bmatrix}_{sub-elite} = \begin{bmatrix} \arg f_{best}([x_{1,1}, x_{1,2}, \cdots, x_{1,q}]) \\ \arg f_{best}([x_{2,1}, x_{2,2}, \cdots, x_{2,q}]) \\ \vdots \\ \arg f_{best}([x_{p,1}, x_{p,2}, \cdots, x_{p,q}]) \end{bmatrix}, \tag{6}$$

where p is the number of sub-elite groups, q represents the number of individuals contained in each sub-population, $\arg f_{best}$ denotes the inverse function of fitness evaluation, and $gbest^k$ $(k = 1, 2, \ldots, p)$ represents the position of the kth sub-elite individual in the sub-elite group. By introducing the MS strategy, the diversity of the population can be ensured, the exploration ability can be increased in the search space, and the algorithm can be prevented from falling into local optima.

3.2. Comprehensive Learning (CL) Strategy

After finding some promising solutions during the exploration phase, local exploitation means attempting to delve deeper into these solutions to find better ones [35,36]. However, according to Equations (4) and (5), it can be seen that the position update rule during the exploration phase involves upper and lower bounds without any information exchange between individuals. That is, all individuals only affect the convergence rate by increasing or decreasing a fixed step factor. Inspired by the CL particle swarm optimizer (CLPSO) [37], the CL strategy is used during the exploitation phase. On the one hand, the CL strategy can preserve individual historical information and facilitate information sharing. On the other hand, the population does not learn from all dimensions of a single individual, but rather from different dimensions of the entire population. The learning probability for each dimension is determined based on a probability value [37]. The learning probability for the *ith* individual can be described as follows:

$$P_i^{cl} = 0.05 + 0.45\frac{\exp(10(i-1)/(Np-1)) - 1}{\exp(10) - 1}, \tag{7}$$

where Np is population size. The pseudo-code of the MCLAOA is shown in Algorithm 1.

Algorithm 1 Pseudo-code of the CL strategy

```
1: for $i$ = 1: $Np$
2:     Generate random number $r$
3:     Compute the learning probability value ($P_i^{cl}$) using Equation (7).
4:     Give the index of two random individuals, $f'_{i,j}$ and $f''_{i,j}$ .
5:     if $r < P_i^{cl}$
6:         if $f\left(X_{f'_{i,j}}\right) < f\left(X_{f''_{i,j}}\right)$
7:             $X_{fi,j} = X_{f'_{i,j}}$
8:         Else
9:             $X_{fi,j} = X_{f''_{i,j}}$
10:        End if
11:    Else
12:        $X_{fi,j} = X_{i,j}$
13:    End if
14: End for
```

3.3. Improved AOA with MS and CL (MCLAOA)

Based on the above analysis, the MS and CL strategies were introduced into the standard AOA to increase population diversity and improve the convergence performance. For the the MCLAOA, the MS strategy was only applied in the exploration phase, while the CL strategy was only applied in the exploitation phase. The specific details of these improvements will be introduced in the following subsection.

Exploration phase: In this phase, considering that expanding or shrinking by a certain proportion always limits the exploration performance of AOA, inspired by the teaching-learning-based optimization (TLBO) [38], we introduced the teaching phase of TLBO. Specifically, half of the subpopulations adopted the exploration phase of AOA, and the rest adopted the teaching phase.

For the first half of the subpopulation specifically, the MS strategy was introduced in Section 3.1. We employed multiple sub-elites to replace a single elite, increasing the diversity of solutions generated and preventing premature convergence due to local optima issues. We applied Equation (6) to Equation (2), and the modified Equation (2) can be described as:

$$X_{i,j}^k = \begin{cases} gbest^k(X_j) \div (MOP + \partial) \times S_j, \ r_2 < 0.5, \\ gbest^k(X_j) \times MOP \times S_j, \ otherwise, \end{cases} \tag{8}$$

where $X_{i,j}^k$ represents the position of the *jth* dimension of the *ith* individual in the *kth* group.

For the second half of the population,

$$X_{i,j} = X_{i,j} + rand \times \left(best\left(X_j\right) - T_F \times X_{ave}\right), \tag{9}$$

where X_{ave} is the average value of all individuals, and T_F denotes the teacher factor $(T_F = round[1 + rand(0,1)])$.

Exploitation phase: The standard AOA cannot retain memory during the exploitation phase, which lead to slow convergence. The CL strategy was introduced into Equation (5), individuals can exchange information, speeding up the algorithm's convergence to the global optimum. Therefore, the specific modification is shown in Equation (10):

$$X_{i,j} = \begin{cases} best\left(X_j\right) - MOP \times \left(X_{fi,j} - X_{i,j}\right), \ r_3 < 0.5, \\ best\left(X_j\right) + MOP \times \left(X_{fi,j} - X_{i,j}\right), \ otherwise, \end{cases} \tag{10}$$

where fi, j defines the value of the ith individual in the jth dimension, which determines whether the individual $X_{fi,j}$ learns in its own or other individuals' different dimensions. For choosing its own or other individual's dimension, it depends on the learning probability P_i^{cl} in Equation (7). If the random number r is greater than P_i^{cl}, it is learned from its own dimension $X_{fi,j}$, otherwise it occurs from another individual's dimension $X_{fi,j}$.

Modified MOA: The parameter MOA, which varies linearly with the number of iterations, cannot reflect the real optimization problem. Therefore, this paper modifies MOA using non-linear parameters with a tangent function, as shown in Figure 3. The tangent function is introduced into Equation (1), and its modified MOA can be expressed as:

$$MOA'(t) = Min + (Max - Min) \times \tan\left(0.25\frac{t}{T}\pi\right). \tag{11}$$

As can be seen from Figure 3, compared with the original MOA, MOA' can better balance and transition the exploration and exploitation.

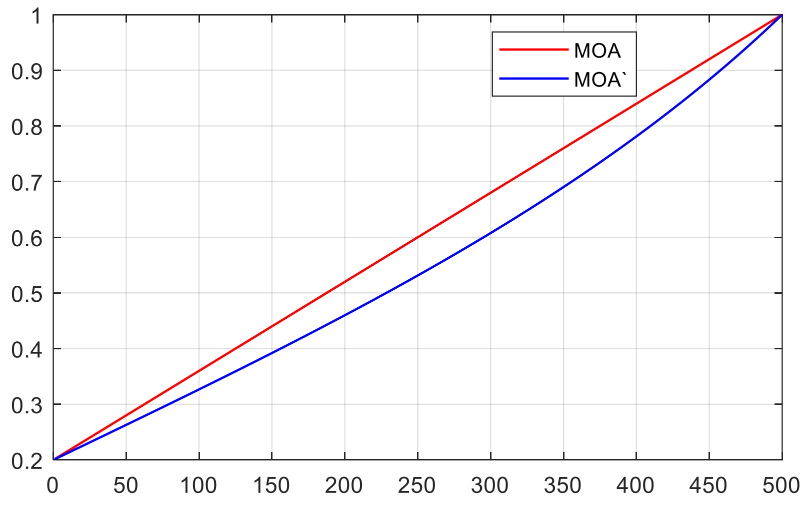

Figure 3. The MOA with the increase of iterations.

In summary, the MS and teaching strategies were applied in the exploration phase. The MS improved population diversity and enabled faster global search. Furthermore, the search method in the exploration phase is limited by a step factor determined by the upper and lower bounds, which constrains the algorithm's search capability. Therefore, we introduced the teaching phase of TLBO [38], where collective information from all individuals was used to mitigate this limitation. The CL strategy was applied in the exploitation phase of the algorithm, accelerating the convergence to the global optimum through information exchange among individuals. This strategy addressed the slow convergence issue caused by the addition and subtraction operators. In addition, we designed Equation (11) to effectively balance and smoothly transition between exploration and exploitation. We denote the improved AOA as MCLAOA. The detailed flowchart of the proposed MCLAOA is described in Figure 4. For clarity, the main contributions of this paper are highlighted, including MOA', the exploration phase, and the exploitation phase. In addition, the pseudo-code of MCLAOA is shown in Algorithm 2.

Algorithm 2 Pseudo-code of the proposed MCLAOA

```
1: Initialize: population size $Np$, position $X_{i,j}$, parameters $\eta$ and $\alpha$, the maximum number of iterations $T$, $p$ group (sub-population).
2: Update:
3: While $t < T$
4:     Calculate the Fitness Function for the given solutions.
5:     Find the best solution (Determined best so far).
6:     Update the MOA' value and MOP value using Equations (11) and (3).
7:     for $i$ = 1: $Np$
8:         for $j$ = 1 to Positions do
9:             Generate a random value between [0, 1] ($r_1$, $r_2$, and $r_3$)
10:            if $r_1 > MOA$ then
11:                %%%Exploration phase%%%
12:                For the first half of the subpopulation,
13:                if $r_2 > 0.5$ then
14:                    (1) Apply the Division math operator (D "÷")
15:                    Update the $ith$ solutions' positions using the first rule in Equation (8).
16:                Else
17:                    (2) Apply the Division math operator (M " ×")
18:                    Update the $ith$ solutions' positions using the first rule in Equation (8).
19:                End if
20:                For the second half of the subpopulation,
21:                Update the $ith$ solutions' positions using Equation (9).
22:            Else
23:                %%%Exploitation phase%%%
24:                Generate random number $r$
25:                Compute the learning probability value ($P_i^{cl}$) using Equation (7).
26:                Decide whether $X_{fi,j}$ is its own or another individual.
27:                if $r_3 > 0.5$ then
28:                    (1) Apply the Subtraction math operator (S "-").
29:                    Update the $ith$ solutions' positions using the first rule in Equation (10).
30:                Else
31:                    (2) Apply the Addition math operator (A "+").
32:                    Update the $ith$ solutions' positions using the second rule in Equation (10).
33:                End if
34:            End if
35:        End for $j$
36:    End for $i$
37:    $t = t + 1$
38: End While
39: Output: Return the best solution $best$.
```

3.4. Computational Complexity

Compared to the standard AOA, the proposed MCLAOA mainly introduces the MS strategy, CL strategy, and the modified MOA. Considering that the MS strategy involves sub-elite groups, it is necessary to conduct a fitness evaluation ranking, denoted as $O(Np \cdot \log(Np))$. The computational complexity of the CL strategy is $O(Np \cdot D)$, where D is the dimension. The computational complexity of the modified MOA is almost unchanged compared to the original MOA. Therefore, the computational complexity of the proposed MCLAOA is $O(MCLAOA) = O(Np \cdot D) + O(T \cdot (Np \cdot D + Np \cdot \log(Np)))$. If T is much greater than 1, then $O(MCLAOA) \approx O(T \cdot Np \cdot (D + \log(Np)))$.

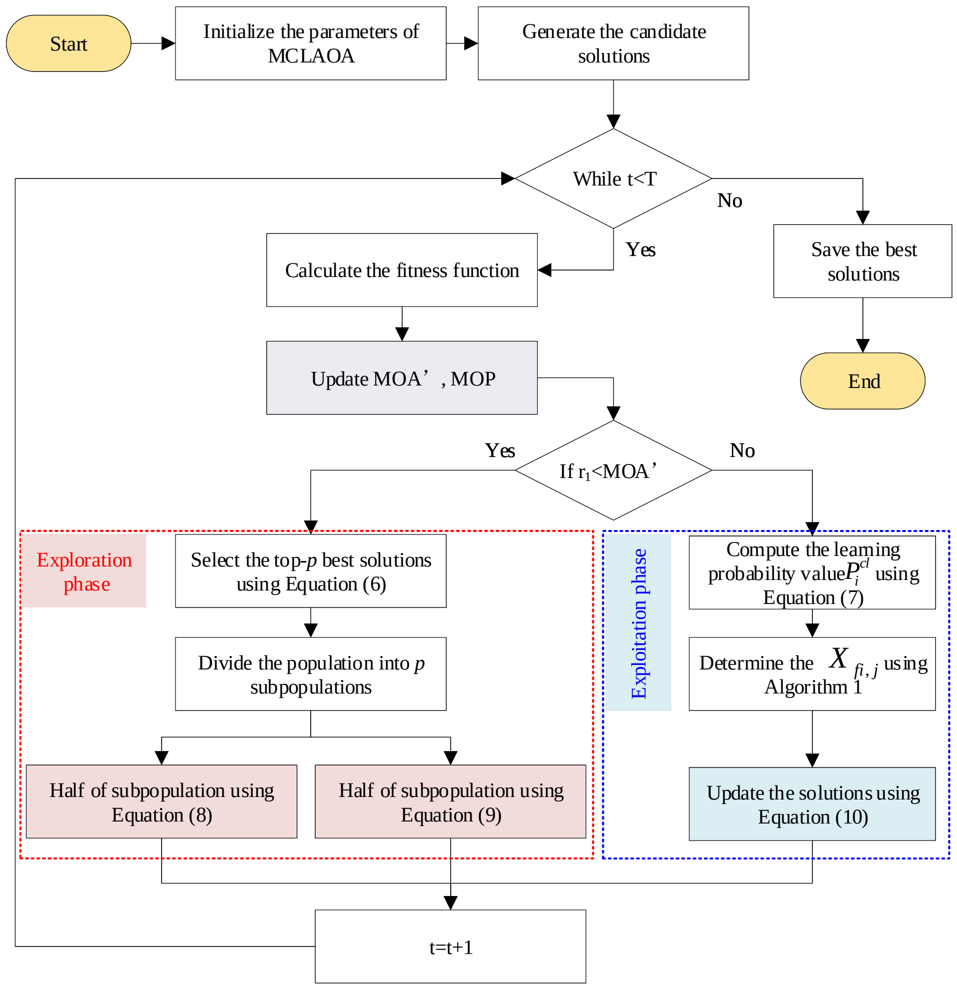
Start
Initialize the parameters of MCLAOA
Generate the candidate solutions
While t<T
No
Yes
Save the best solutions
End
Calculate the fitness function
Update MOA', MOP
If r1<MOA'
Yes
No
Exploration phase
Select the top-p best solutions using Equation (6)
Divide the population into p subpopulations
Half of subpopulation using Equation (8)
Half of subpopulation using Equation (9)
Exploitation phase
Compute the learning probability value P_i^{cl} using Equation (7)
Determine the $X_{fi,j}$ using Algorithm 1
Update the solutions using Equation (10)
t=t+1

Figure 4. Flowchart of the proposed MCLAOA.

4. Results and Analysis

The experiments were conducted using MATLAB2017b, and they were run on a PC with Intel Core i7-10700 2.90GHz and 16GB RAM. To examine the performance of the proposed MCLAOA, the 23 benchmark functions and 2 engineering design problems were employed. Among them, 23 benchmark functions [16] are shown in Table 1.

To verify the advanced performance of the proposed MCLAOA, comparisons were made with several algorithms, including (1) some versions of AOA, such as the AOA [16], the chaotic AOA (CAOA) [18], (2) advanced metaheuristic algorithms, such as the reptile search algorithm (RSA) [39], the whale optimization algorithm (WOA) [40], and the grasshopper optimization algorithm (GOA) [41], (3) the classical metaheuristic algorithm and particle swarm optimization (PSO) [42], and (4) the winner of CEC competition, the L-SHADE [43]. For some versions of AOA, the AOA was employed to validate the effectiveness of MCLAOA, while the CAOA was employed to test the strong competitiveness of MCLAOA compared to the versions of AOA. It is worth noting that the CAOA [18] has been proven to outperform some advanced algorithms, including the HHO [44], the EO [45], and the WHO [46], in certain optimization problems. For advanced and classical metaheuristic algorithms, these comparative algorithms can confirm that the MCLAOA achieves state-of-the-art performance and outperforms classical algorithms of the same type. For the winner of CEC competition, once it is confirmed that the MCLAOA outperforms the LSHADE, it can be classified as a high-performance optimizer. All algorithms were set with parameters as shown in Table 2. For the sake of fairness in comparison, the maximum function evaluation with a population size of $Np = 50$ and $FES = 100{,}000$ was selected for 23 benchmark functions. All algorithms were independently run 30 times on each test function. To compare the superiority and inferiority of these algorithms, the evaluation indicators used were the average (*ave*) and standard deviation (*std*) as well as the best optimal value (*best*), and convergence curves were used to indicate the convergence performance of the algorithms. Box plots were adopted to verify the stability of the algorithms. The Wilcoxon rank-sum test and Friedman rank test were employed to reflect the statistical significance of the algorithms [47]. Next, experimental analysis was conducted on 23 benchmark functions.

Table 1. 23 benchmark functions.

Type	No.	Description	R	Dim	f_{min}
Unimodal functions	F_1	Sphere	$[-100, 100]$	30	0
	F_2	Schwefel 2.22	$[-10, 10]$	30	0
	F_3	Schwefel 1.2	$[-100, 100]$	30	0
	F_4	Schwefel 2.21	$[-100, 100]$	30	0
	F_5	Rosenbrock	$[-30, 30]$	30	0
	F_6	Step	$[-100, 100]$	30	0
	F_7	Quartic	$[-1.28, 1.28]$	30	0
Multimodal functions	F_8	Schwefel	$[-500, 500]$	30	$-418.98 \times$D
	F_9	Rastrigin	$[-5.12, 5.12]$	30	0
	F_{10}	Ackley	$[-32, 32]$	30	0
	F_{11}	Griewank	$[-600, 600]$	30	0
	F_{12}	Penalized	$[-50, 50]$	30	0
	F_{13}	Penalize 2	$[-50, 50]$	30	0
Fixed-dimension multimodal functions	F_{14}	Foxholes	$[-65.536, 65.536]$	2	0.9980
	F_{15}	Kowalik	$[-5, 5]$	4	0.0003
	F_{16}	Six-hump Camel-Back	$[-5, 5]$	2	−1.0316
	F_{17}	Branin	$[-5, 5]$	2	0.398
	F_{18}	Goldstein-Price	$[-2, 2]$	2	3
	F_{19}	Hartman 3	$[1, 3]$	3	−3.863
	F_{20}	Hartman 6	$[0, 1]$	6	−3.322
	F_{21}	Shekel5	$[0, 10]$	4	−10.153
	F_{22}	Shekel7	$[0, 10]$	4	−10.403
	F_{23}	Shekel10	$[0, 10]$	4	−10.536

Table 2. The parameter setting of the algorithms.

Algorithms	Parameters	Values
MCLAOA	$\alpha, \eta, Max, Min, p$	5, 0.5, 1, 0.2, 5
AOA [16]	α, η, Max, Min	5, 0.5, 1, 0.2
CAOA [18]	α, η	5, 0.5
RSA [39]	α, β	0.1, 0.1
WOA [40]	b, l, a, p	1, [−1, 1], 2 to 0, [0,1]
GOA [41]	l, f, c	1.5, 0.5, [0,1]
PSO [42]	W, C_1, C_2, V_{max}	0.9 to 0.4, 2, 2, 4
L-SHADE [43]	$H, Pbestrate, Arcrate$	5, 0.11, 1.4

4.1. Results Comparisons Using 23 Benchmark Functions

The 23 benchmark functions [16] are a classic function benchmark for evaluating optimization algorithms which can be divided into three types: unimodal functions (F_1–F_7), multimodal functions (F_8–F_{13}), and fixed-dimension multimodal functions (F_{14}–F_{23}). For details on the 23 benchmark functions, please refer to Table 1. The experimental results of all algorithms with *ave*, *std*, and *best* are shown in Table 3, and the best results of each function are marked in bold type.

Table 3. Numerical results of MCLAOA with other algorithms using 23 benchmark functions.

Function	Criteria	MCLAOA	AOA	CAOA	RSA	WOA	GOA	PSO	LSHADE
	ave	**0.00E+00**	6.42E-08	**0.00E+00**	**0.00E+00**	4.94E-324	2.81E-06	8.77E-03	3.96E-22
F_1	*std*	0.00E+00	6.66E-08	0.00E+00	0.00E+00	0.00E+00	1.88E-06	4.74E-03	9.97E-22
	best	0.00E+00	1.93E-13	0.00E+00	0.00E+00	0.00E+00	7.02E-07	2.04E-03	2.67E-24
	ave	**0.00E+00**	3.23E-05	**0.00E+00**	**0.00E+00**	3.91E-222	6.14E+00	1.32E+00	6.53E-08
F_2	*std*	0.00E+00	9.34E-05	0.00E+00	0.00E+00	0.00E+00	1.05E+01	8.00E-01	2.31E-07
	best	0.00E+00	8.55E-17	0.00E+00	0.00E+00	1.61E-234	4.51E-03	2.90E-01	4.49E-12
	ave	**0.00E+00**	6.71E-06	**0.00E+00**	**0.00E+00**	2.85E+03	3.91E+02	9.53E-02	1.12E-01
F_3	*std*	0.00E+00	1.44E-05	0.00E+00	0.00E+00	2.97E+03	1.27E+03	4.78E-02	1.29E-01
	best	0.00E+00	6.75E-14	0.00E+00	0.00E+00	1.37E+02	1.81E+01	1.44E-02	6.81E-03
	ave	**0.00E+00**	1.96E-03	**0.00E+00**	**0.00E+00**	2.48E+01	6.73E-01	5.61E-01	6.07E+00
F_4	*std*	0.00E+00	3.63E-03	0.00E+00	0.00E+00	2.63E+01	6.27E-01	3.46E-01	1.82E+00
	best	0.00E+00	1.13E-05	0.00E+00	0.00E+00	8.14E-04	1.47E-01	1.22E-01	2.61E+00
	ave	2.88E+01	2.56E+01	2.74E+01	**8.64E+00**	2.58E+01	1.09E+02	4.23E+01	3.72E+01
F_5	*std*	3.45E-01	3.87E-01	3.26E-01	1.34E+01	2.80E-01	2.40E+02	3.33E+01	2.32E+01
	best	2.81E+01	2.45E+01	2.62E+01	5.85E-29	2.50E+01	1.88E+01	2.28E+01	1.26E+01
	ave	9.67E+00	1.42E-07	9.92E-05	6.58E+00	2.87E-04	2.53E-06	7.66E-03	**7.18E-22**
F_6	*std*	2.74E-01	3.68E-08	4.17E-05	7.59E-01	9.95E-05	1.80E-06	3.64E-03	2.02E-21
	best	5.80E-01	7.47E-08	4.63E-05	5.06E+00	1.37E-04	6.00E-07	2.41E-03	3.60E-25
	ave	**9.55E-07**	8.93E-06	9.93E-06	1.84E-05	6.32E-04	1.68E-02	6.69E-02	3.70E-02
F_7	*std*	1.08E-06	8.50E-06	8.78E-06	1.76E-05	6.85E-04	5.28E-03	2.47E-02	1.55E-02
	best	4.20E-09	2.90E-07	1.47E-07	7.54E-07	5.45E-05	8.52E-03	2.87E-02	1.18E-02
	ave	**−1.02E+06**	−5.66E+03	−8.16E+03	−5.51E+03	−1.22E+04	−7.26E+03	−4.23E+03	−2.09E+04
F_8	*std*	4.98E+05	4.31E+02	4.46E+02	1.61E+02	5.97E+02	6.82E+02	1.68E+03	8.66E+01
	best	−2.16E+06	−6.51E+03	−8.98E+03	−5.74E+03	−1.26E+04	−8.40E+03	−7.70E+03	−2.09E+04
	ave	**0.00E+00**	2.14E-09	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.68E+02	4.00E+01	4.28E-08
F_9	*std*	0.00E+00	1.12E-08	0.00e+00	0.00E+00	0.00E+00	5.62E+01	1.13E+01	2.05E-08
	best	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	6.17E+01	2.39E+01	4.65E-09
	ave	**8.88E-16**	1.29E-05	**8.88E-16**	**8.88E-16**	4.44E-15	1.47E+00	3.55E+00	2.56E+00
F_{10}	*std*	0.00E+00	2.49E-05	0.00e+00	0.00E+00	2.29E-15	9.42E-01	6.02E-01	5.37E-01
	best	8.88E-16	1.51E-12	8.88E-16	8.88E-16	8.88E-16	1.97E-04	2.53E+00	1.56E+00
	ave	**0.00E+00**	7.01E-07	7.53E-04	**0.00E+00**	9.44E-04	2.34E-02	1.08E-02	6.22E-03
F_{11}	*std*	0.00E+00	2.31E-07	4.04E-03	0.00E+00	5.17E-03	1.81E-02	1.49E-02	1.34E-02
	best	0.00E+00	4.02E-07	3.64E-06	0.00E+00	0.00E+00	1.05E-03	2.32E-04	1.11E-16
	ave	5.29E-02	2.27E-01	**1.27E-04**	1.18E+00	9.80E-04	2.42E+00	1.45E-01	1.29E-01
F_{12}	*std*	2.87E-02	2.60E-02	1.50E-04	3.66E-01	4.24E-03	1.44E+00	1.84E-01	2.59E-01
	best	1.35E-02	1.84E-01	1.33E-05	5.04E-01	1.90E-05	1.29E-01	3.42E-05	1.85E-22
	ave	2.47E+00	2.96E+00	2.85E+00	2.53E-01	6.79E-03	7.09E-03	5.67E-01	**2.47E-01**
F_{13}	*std*	1.06E-01	1.70E-02	1.58E-01	7.40E-01	1.92E-02	9.50E-03	1.28E+00	7.92E-01
	best	2.25E+00	2.91E+00	2.22E+00	5.29E-32	3.96E-04	9.16E-06	6.10E-04	4.97E-24
	ave	7.05E+00	9.60E+00	7.69E+00	2.54E+00	1.13E+00	**9.98E-01**	1.26E+00	**9.98E-01**
F_{14}	*std*	5.12E+00	4.79E+00	3.45E+00	1.92E+00	5.03E-01	3.44E-16	9.32E-01	0.00E+00
	best	9.98E-01	9.98E-01	9.98E-01	1.01E+00	9.98E-01	9.98E-01	9.98E-01	9.98E-01
	ave	5.68E-03	2.12E-03	7.14E-04	9.71E-04	5.00E-04	8.09E-03	1.05E-03	**3.38E-04**
F_{15}	*std*	1.04E-02	4.95E-03	1.13E-03	3.41E-04	2.73E-04	1.26E-02	3.66E-03	1.67E-04
	best	4.51E-04	3.08E-04	3.12E-04	4.68E-04	3.09E-04	3.08E-04	3.07E-04	3.07E-04
	ave	**−1.03E+00**	**−1.03E+00**	**−1.03E+00**	**−1.03E+00**	**−1.03E+00**	**−1.03E+00**	**−1.03E+00**	**−1.03E+00**
F_{16}	*std*	1.13E-02	1.34E-12	7.85E-12	2.66E-04	1.83E-13	2.45E-15	6.78E-16	6.78E-16
	best	−1.03E+00	−1.03E+00	−1.03E+00	−1.03E+00	−1.03E+00	−1.03E+00	−1.03E+00	−1.03E+00
	ave	**3.98E-01**	**3.98E-01**	**3.98E-01**	4.01E-01	**3.98E-01**	**3.98E-01**	**3.98E-01**	**3.98E-01**
F_{17}	*std*	1.25E-09	1.54E-07	5.77E-07	6.34E-03	8.31E-09	3.66E-08	0.00E+00	0.00E+00
	best	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01

Table 3. *Cont.*

Function	Criteria	MCLAOA	AOA	CAOA	RSA	WOA	GOA	PSO	LSHADE
F_{18}	*ave*	**3.00E+00**	8.40E+00	1.11E+01	**3.00E+00**	**3.00E+00**	**3.00E+00**	**3.00E+00**	**3.00E+00**
	std	1.22E-14	1.65E+01	1.26E+01	3.96E-05	8.75E-07	3.17E-14	1.26E-15	1.68E-15
	best	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00
F_{19}	*ave*	−**3.86E+00**	−**3.86E+00**	−**3.86E+00**	−3.85E+00	−**3.86E+00**	−3.76E+00	−**3.86E+00**	−**3.86E+00**
	std	2.32E-15	3.40E-07	7.21E-04	1.38E-02	2.09E-03	2.39E-01	2.40E-03	2.71E-15
	best	−3.86E+00	−3.86E+00	−3.86E+00	−3.86E+00	−3.86E+00	−3.86E+00	−3.86E+00	−3.86E+00
F_{20}	*ave*	−3.27E+00	−3.30E+00	−3.28E+00	−2.89E+00	−3.23E+00	−3.28E+00	−3.24E+00	−**3.31E+00**
	std	1.06E-01	4.51E-02	5.85E-02	2.51E-01	8.78E-02	5.83E-02	8.22E-02	3.63E-02
	best	−3.32E+00	−3.32E+00	−3.32E+00	−3.16E+00	−3.32E+00	−3.32E+00	−3.32E+00	−3.32E+00
F_{21}	*ave*	−**10.1513**	−9.39E+00	−9.98E+00	−5.06E+00	−9.98E+00	−5.64E+00	−6.65E+00	−9.82E+00
	std	2.24E-03	2.01E+00	9.31E-01	3.03E-07	9.30E-01	3.37E+00	3.45E+00	1.28E+00
	best	−10.1532	−10.1532	−10.1532	−5.06E+00	−10.1532	−10.1532	−10.1532	−10.1532
F_{22}	*ave*	−**10.4029**	−9.35E+00	−9.97E+00	−5.09E+00	−9.83E+00	−5.80E+00	−6.93E+00	−1.02E+01
	std	8.73E-16	2.42E+00	1.67E+00	2.89E-02	1.77E+00	3.64E+00	3.61E+00	1.22E+00
	best	−10.4029	−10.4029	−10.4029	−5.25E+00	−10.4029	−10.4029	−10.4029	−10.4029
F_{23}	*ave*	−**10.5364**	−7.95E+00	−1.03E+01	−5.13E+00	−1.05E+01	−6.02E+00	−7.09E+00	−10.5364
	std	2.86E-15	3.73E+00	1.41E+00	1.40E-06	8.00E-05	4.03E+00	3.79E+00	1.75E-15
	best	−10.5364	−10.5364	−10.5364	−5.13E+00	−10.5364	−10.5364	−10.5364	−10.5364
Friedman rank test		2.1034	3.9655	3.3103	4.7241	4.2759	6.0345	6.3448	5.2414
rank		1	3	2	5	4	7	8	6

Note: The best results of each function were marked in **bold** type in terms of *ave*.

4.1.1. Unimodal Functions and Exploitation

The unimodal functions (F_1–F_7) have only one global solution and no local solution, which is used to test the exploitation ability of the algorithm. It can be seen from Table 3 that the proposed MCLAOA has stronger advantages in terms of *ave* value compared to other comparison algorithms, except for F_5 and F_6. For F_1–F_4, both the MCLAOA and the CAOA achieve convergence with an *ave* value of 0, while the RSA also converges to 0. However, the AOA fails to converge to 0 in terms of the *best* metric. These results indicate that the exploitation performance of MCLAOA has been significantly improved. This is attributed to the introduction of the CL strategy which modifies the mathematical model of the exploitation phase and enhances the convergence to the optimal solution by sharing information among individuals.

4.1.2. Multimodal Functions and Exploration

The multimodal function contains multiple local optimal solutions, which are used to test the algorithm's ability to escape from poor local optima and obtain the near-global optimum. For multimodal functions (F_8–F_{13}) with a dimension of 30, the *ave* value of the proposed MCLAOA ranks first except for F_{12} and F_{13}. Compared with high-dimension multimodal functions (F_8–F_{13}), fixed-dimension multimodal functions (F_{14}–F_{23}) have only a few local minima, and the dimension of the function is small and fixed. It is worth noting that the *best* values of all algorithms converge to the global optimum for F_{16}–F_{19}, and for F_{20}–F_{23}, all algorithms except RSA also converge to the global optimum. These results demonstrate their ability to converge to the global optimum. However, when considering *ave* and *std* values together, the proposed MCLAOA exhibits superior performance, indicating its ability to converge more stably to the global optimum. Therefore, it can be seen from the experimental results of multimodal functions that the proposed MCLAOA has good global exploration performance. The MS strategy divides the population Np into p subpopulations by introducing p sub-elites instead of a single elite. This strategy enhances the global search capability. Additionally, we introduce a teaching phase to half of the subpopulations, which alleviates the limitations of expansion or shrinkage step size factor in the exploration phase of AOA. As a result, the proposed MCLAOA outperforms other algorithms, especially in F_8, F_{21}–F_{23}.

4.1.3. Convergence Behavior Analysis

To observe the convergence of the proposed MCLAOA and comparison algorithms, we record and save the fitness of the best solution for each iteration to draw the convergence curve. The convergence results of all algorithms on 23 benchmark functions are shown in Figure 5. It can be seen from Figure 5 that only the CAOA, the RSA, and the MCLAOA show a clear downward trend on F_1–F_4, and the proposed MCLAOA shows a more obvious decrease for F_3 and F_4 compared to the CAOA and the RSA. For multimodal functions F_8–F_{11}, the proposed MCLAOA achieves a significantly faster convergence to the global optimum compared to the other seven comparison algorithms. All algorithms can converge to the global optimum for F_{16} and F_{17}, but the convergence curve of the proposed MCLAOA drops faster than the AOA and the CAOA. Moreover, the proposed MCLAOA ranks first in convergence performance on F_{21}–F_{23} among all comparison algorithms, indicating its good exploitation performance. In summary, the proposed MCLAOA has achieved advanced performance. Compared with the basic AOA and the improved version CAOA, the proposed MCLAOA has improved convergence performance. In addition, since the comparison algorithms are meta-heuristic algorithms with randomness, in this paper, we employ Box plots to analyze the stability of the results. Box plots of the result of a global minimum of the MCLAOA, the AOA, the CAOA, the RSA, the WOA, the GOA, the PSO and the L-SHADE for F_1, F_7, F_{11}, and F_{23} are shown in Figure 6. It can be observed from Figure 6 that the proposed MCLAOA outperforms other algorithms in the stability of the results during the running of the algorithm.

Based on the above analysis, the proposed MCLAOA shows strong advantages in terms of convergence accuracy, convergence speed and robustness.

4.2. Statistical Analysis

It is worth mentioning that statistical analysis is very important for the statistical authenticity of results in the field of optimization algorithms. In this paper, the Wilcoxon rank-sum test and the Friedman test are employed.

The statistical results of 23 benchmark functions are shown in Table 4. From the data results in Table 4, it can be observed that the proposed MCLAOA performs better than other comparison algorithms in most functions among the 23 benchmark functions. The number of functions in which the performance is improved compared to the basic AOA and improved CAOA is 16 and 9, respectively.

Table 4. Statistical results over 23 benchmark functions.

Algorithm	**23 Benchmark Functions**		
	$+$	$-$	$\approx$
AOA	4	16	3
CAOA	5	9	9
RSA	5	9	9
WOA	6	12	5
GOA	4	16	3
PSO	4	15	4
L-SHADE	5	13	5

Note: "$+$", "$-$", "$\approx$" denote that the performance of the corresponding algorithm is statistically better than, worse than, and similar to that of MCLAOA, respectively.

In order to compare the results of each run and determine the significance of the results, a non-parametric pair-wise Wilcoxon rank-sum test has been employed. The tests were conducted at a significance level of 5%. For the Wilcoxon rank-sum test, the best-performing algorithm was chosen in each test function and compared to other algorithms. That is, if the best algorithm is MCLAOA, pair-wise comparisons are made between MCLAOA and AOA, MCLAOA and CAOA, MCLAOA and RSA, etc. Note that since the best algorithm cannot be compared with itself; N/A is written for the best algorithm in each function to indicate that it is not applicable. The results are presented in Table 5. It is evident from Table 5 that these results are statistically significant, as the p-values are significantly less than 0.05 for almost all functions.

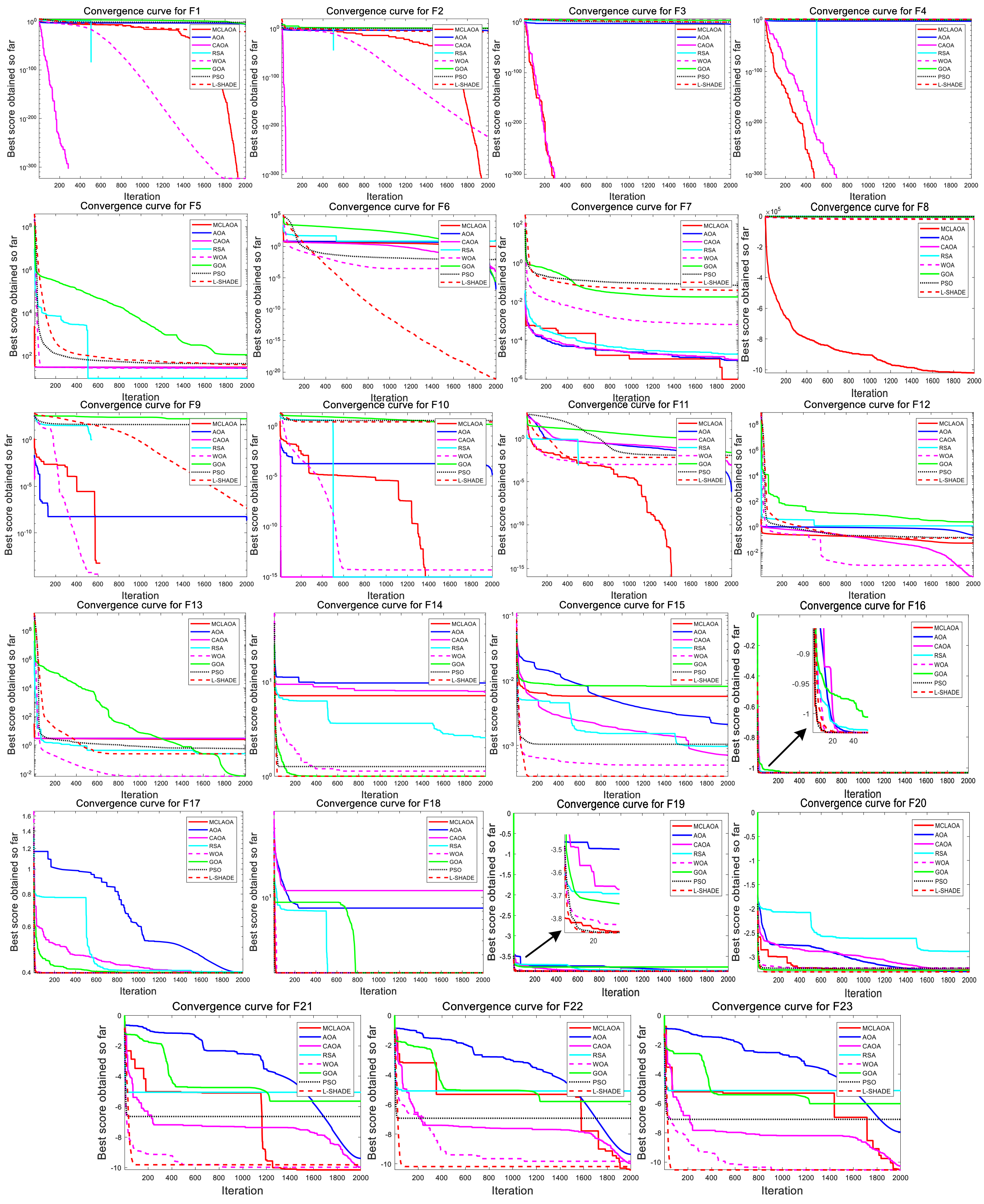

Figure 5. Convergence curves for 23 benchmark functions. Better viewed in color with zoom-in.

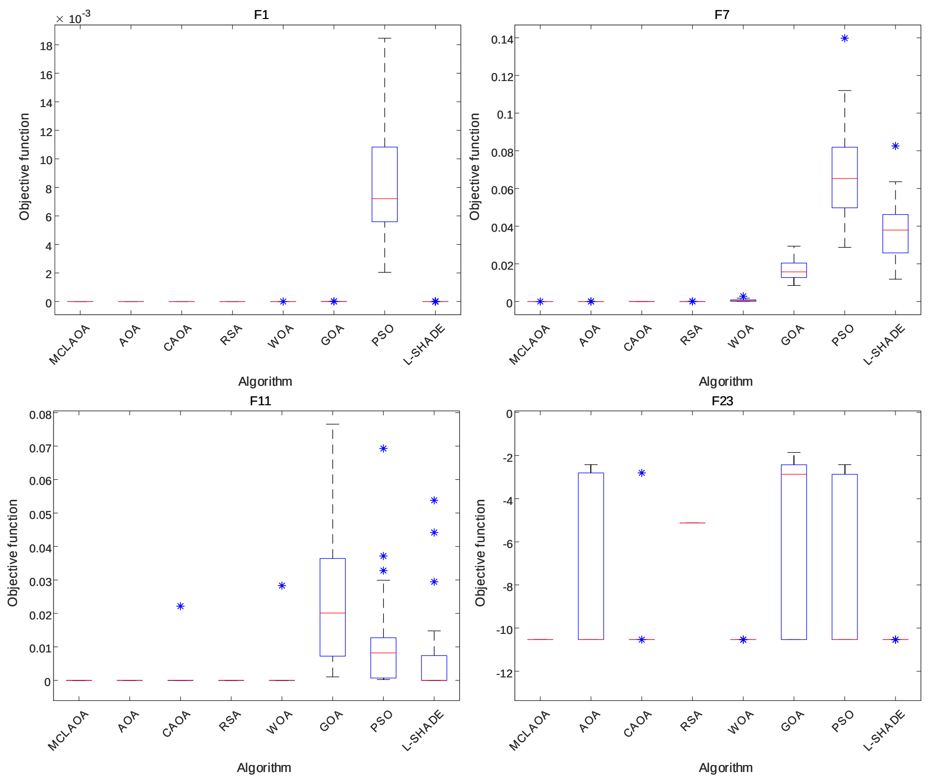

Figure 6. Box plots of the result of a global minimum for functions F_1, F_7, F_{11}, and F_{23}.

Table 5. *p*-values of the Wilcoxon rank-sum test over 23 benchmark functions.

Function	MCLAOA	AOA	CAOA	RSA	WOA	GOA	PSO	LSHADE
F_1	N/A	1.21E-12	N/A	N/A	5.20E-06	1.21E-12	1.21E-12	1.21E-12
F_2	N/A	1.21E-12	N/A	N/A	1.21E-12	1.21E-12	1.21E-12	1.21E-12
F_3	N/A	1.21E-12	N/A	N/A	1.21E-12	1.21E-12	1.21E-12	1.21E-12
F_4	N/A	1.21E-12	N/A	N/A	1.21E-12	1.21E-12	1.21E-12	1.21E-12
F_5	2.65E-06	7.96E-03	7.96E-03	N/A	7.96E-03	1.67E-05	2.88E-06	1.61E-06
F_6	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	N/A
F_7	N/A	1.60E-07	1.36E-07	3.47E-10	3.02E-11	3.02E-11	3.02E-11	3.02E-11
F_8	N/A	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11
F_9	N/A	1.10E-02	N/A	N/A	N/A	1.21E-12	1.21E-12	1.21E-12
F_{10}	N/A	1.21E-12	N/A	N/A	9.84E-10	1.21E-12	1.21E-12	1.21E-12
F_{11}	N/A	1.21E-12	1.21E-12	N/A	0.3337	1.21E-12	1.21E-12	1.18E-12
F_{12}	3.02E-11	3.02E-11	N/A	3.02E-11	0.3183	3.02E-11	8.15E-05	0.3790
F_{13}	2.67E-09	5.57E-10	6.12E-10	6.49E-07	0.2707	5.19E-02	1.04E-04	N/A
F_{14}	1.20E-12	1.20E-12	1.20E-12	1.21E-12	1.21E-12	6.09E-13	6.58E-04	N/A
F_{15}	1.99E-10	3.00E-10	3.63E-10	3.30E-10	3.99E-10	7.71E-11	2.91E-02	N/A
F_{16}	6.28E-04	1.21E-12	1.21E-12	1.21E-12	1.19E-12	1.20E-12	N/A	N/A
F_{17}	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	6.15E-11	N/A	N/A
F_{18}	5.55E-05	5.20E-12	5.20E-12	5.20E-12	5.20E-12	5.19E-12	N/A	2.35E-03
F_{19}	N/A	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	6.12E-14	1.69E-14
F_{20}	2.20E-03	7.61E-09	1.88E-09	4.10E-12	1.79E-10	1.41E-09	6.32E-05	N/A
F_{21}	N/A	6.97E-03	4.51E-02	3.01E-11	3.99E-04	2.71E-02	0.6616	1.23E-09
F_{22}	N/A	7.80E-12	7.80E-12	7.80E-12	7.80E-12	7.80E-12	2.79E-02	9.39E-06
F_{23}	N/A	1.46E-11	1.46E-11	1.46E-11	1.46E-11	1.46E-11	5.74E-02	6.48E-06

Note: N/A represents the best algorithm in terms of optimization performance among all the algorithms for the corresponding function.

In order to calculate the ranking of each algorithm with statistical significance, we conducted a Friedman rank test for all tested algorithms over 23 benchmark functions, and the test results are shown in the last two rows of Table 3. The proposed MCLAOA algorithm ranked first among all algorithms with a Friedman value of 2.1034. Through a series of experiments, it has been verified that the proposed MCLAOA can be regarded as an advanced optimizer with statistically significant results.

4.3. Scalability Analysis

This section uses scalability analysis to verify the reliability of the proposed MCLAOA. Considering that the dimensions of fixed-dimension multimodal functions (F_{14}–F_{23}) are fixed and cannot be changed, this paper selects one function from unimodal functions and multimodal functions, respectively, for analysis, namely F_1 and F_{10}. In the process of scalability analysis, the dimension ranges from 100 to 500, with a step size of 100. The termination condition (i.e., $FES = 100,000$ with $Np = 50$) and parameter settings are consistent with the above experimental conditions, and each function is independently executed 30 times at each dimension. The experimental results are shown in Figure 7, where the x-axis represents the dimension and the y-axis represents the average fitness value obtained from 30 independent runs at each dimension. It is worth noting that the red dashed box represents the enlarged content.

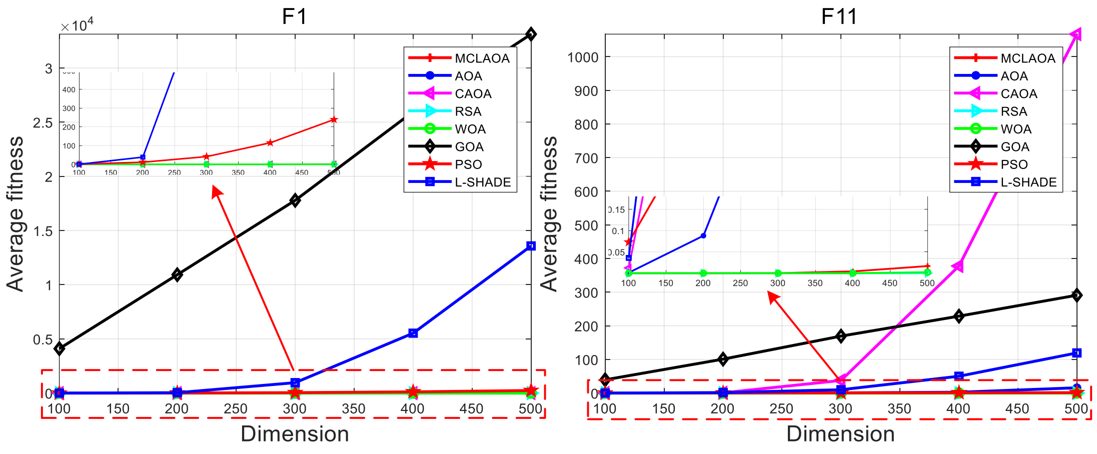

Figure 7. Scalability analysis for functions.

From Figure 7, it can be seen that for F_1, almost all algorithms can converge to the global optimum in all dimensions, except for the GOA, the PSO, and the L-SHADE. For F_{11}, the average fitness of MCLAOA, RSA, and WOA are very close as the dimension increases. At dimension 500, the average fitness of MCLAOA is slightly lower than the RSA and the WOA, which is determined by their mathematical models. However, compared to the AOA and the CAOA, the MCLAOA performs the best in all dimensions, which strongly proves that the proposed MCLAOA has been greatly improved. These results demonstrate that the proposed MCLAOA is reliable, especially compared to the AOA and the CAOA, and exhibits excellent performance even when facing high-dimensional optimization problems.

4.4. Results Comparisons Using CEC2020 Benchmark Problems

To further verify the strong competitiveness and optimization applicability of the proposed MCLAOA, we selected a more complex functional benchmark (CEC2020 benchmark problems [48]) and advanced comparison algorithm (the slime mould algorithm, SMA [49] and the hybridizing TLBO with GOA, TLGOA [50]). Due to space limitations, the detailed description and experimental results of CEC2020 are represented in the Appendix A. These CEC2020 benchmark problems can be divided into four types: unimodal functions (CF_1), basic functions (CF_2–CF_4), hybrid functions (CF_5–CF_7), and composition functions (CF_8–CF_{10}). For details on the CEC2020 benchmark problems with a dimension of 10, please refer to Table A1. The parameter settings of SMA and TLGOA are consistent with the original literature [49,50]. Among all the algorithms, the maximum function evaluation with a population size of $Np = 100$ and $FES = 100,000$ for CEC2020 benchmark problems, and they were independently run 30 times. Similar to the 23 benchmark functions, we also adopt *ave*, *std*, and *best* as evaluation metrics to assess the optimization performance of all algorithms. The experimental results are shown in Table A2, and the best results are marked in bold type.

From Table A2, it can be observed that the MCLAOA shows the best performance in terms of *ave* values on CF_2, CF_8, CF_9, and CF_{10}. The TLGOA performs the best on CF_4, CF_5, and CF_7. The SMA exhibits the best performance on CF_1, CF_3, and CF_6. Furthermore, compared to the standard AOA, the MCLAOA demonstrates significantly better *ave* values, indicating its effectiveness in improving optimization performance. It is worth noting that the Friedman rank test for the MCLAOA in Table A2 is 2.3000, ranking first. The p-values corresponding to the Wilcoxon rank-sum test in Table A3 are almost all significantly smaller than 0.05. These results demonstrate that the experimental results obtained from Table A2 are statistically significant. Therefore, the proposed MCLAOA also demonstrates promising performance on more complex benchmark problems.

In summary, we comprehensively analyzed the optimization performance of MCLAOA from several aspects, including accuracy, convergence curve, box plots, statistical analysis, and scalability analysis. These results also lay the foundation for the application of the algorithm to solve more complex optimization problems.

4.5. Engineering Design Problem

To date, we have analyzed the performance of the proposed MCLAOA on unconstrained function benchmarks. Next, we will discuss optimization problems under complex constraint conditions in real-world scenarios. In this paper, two engineering examples, three-bar truss design and a pressure vessel design, are employed to analyze the proposed MCLAOA.

4.5.1. Three-Bar Truss Design Problem

The objective of the three-bar truss design problem is to minimize the weights of the bar structures under certain constraints [16]. The three-bar truss design mainly involves two optimization parameters: the cross-sections with A_1 and A_2. There are three constraint conditions, and the mathematical model is shown in the following equations. Here, we compare the proposed MCLAOA with some existing optimization algorithms, and the experimental results are shown in Table 6. The experimental results demonstrate that the proposed MCLAOA exhibits strong competitiveness.

$$\text{Take } x = [x_1\ x_2] = [A_1\ A_2],$$
$$\text{Min. } f(x) = (2\sqrt{2}x_1 + x_2)l,$$
$$\text{Subject to}\quad \begin{aligned} g_1(x) &= \frac{\sqrt{2}x_1+x_2}{\sqrt{2}x_1^2+2x_1x_2}P - \sigma \le 0,\\ g_2(x) &= \frac{x_2}{\sqrt{2}x_1^2+2x_1x_2}P - \sigma \le 0,\\ g_3(x) &= \frac{1}{\sqrt{2}x_2+x_1}P - \sigma \le 0,\end{aligned}$$
$$\text{where}\quad \begin{aligned} &l = 100\text{ cm}, P = 2\text{ kN/cm}^2, \sigma = 2\text{ kN/cm}^2,\\ &0 \le x_1 \le 1,\ 0 \le x_2 \le 1.\end{aligned}$$

Table 6. Comparative results for the three-bar truss design problem.

Algorithm	**Optimal Values of Design Variables**		**Optimal Cost**
	A_1	A_2	
MCLAOA	0.7548	0.2920	187.4918
HHO [44]	0.7887	0.4083	263.8958
IAOA [51]	0.7897	0.4045	263.8537
CAOA [18]	0.7841	0.4237	263.9362
AOASC [20]	0.7884	0.4081	263.8523
OSAOA [52]	1.00	0.00	282.84
AOA [16]	0.79369	0.39426	263.9154
RSA [39]	0.7887	0.40805	263.8928
GOA [41]	0.7889	0.4076	263.8959

4.5.2. Pressure Vessel Design Problem

The objective of the pressure vessel design problem is to determine the total cost of a cylindrical pressure vessel and minimize the result [16]. The pressure vessel design involves four design variables: the inner radius (R), the thickness of the head (Th), thickness of the shell (Ts), and the length of the cylindrical part without examining the head (L). There are four constraints, and the mathematical model can be represented by the following equations. Here, we will compare the proposed MCLAOA with some existing optimization algorithms in terms of pressure vessel design, and the experimental results are shown in Table 7. It can be clearly seen that the proposed MCLAOA has significant advantages in solving the pressure vessel design problem.

$$\begin{aligned}
&\text{Take} \quad x = [x_1\ x_2\ x_3\ x_4] = [T_s\ T_h\ R\ L],\\
&\text{Min. } f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3,\\
&\qquad\qquad g_1(x) = -x_1 + 0.0193x_3 \le 0,\\
&\qquad\qquad g_2(x) = -x_3 + 0.00954x_3 \le 0,\\
&\text{Subject to} \quad g_3(x) = -\pi x_3^2x_4 - \tfrac{4}{3}\pi x_3^2 + 129600 \le 0,\\
&\qquad\qquad g_4(x) = x_4 - 240 \le 0,\\
&\qquad\qquad 0 \le x_1, x_2 \le 99, 10 \le x_3, x_4 \le 200.
\end{aligned}$$

Table 7. Comparative results for the pressure vessel design problem.

Algorithm	**Optimal Values of Design Variables**				**Optimal Cost**
	Ts	***Th***	***R***	***L***	
MCLAOA	1.1134	0	67.2893	10	3675.9636
IAOA [51]	0.7637	0.3705	41.5666	184.1352	5813.5505
CAOA [18]	0.8416	0.4139	45.2890	155.7818	5822.6083
AOASC [20]	0.8254	0.4262	42.7605	169.3396	6048.6812
OSAOA [52]	0.8125	0.4375	42.0984	176.6512	6060.0479
MPA [53]	0.7782	0.3846	40.3196	200.00	5885.3353
HHO [44]	0.8176	0.4073	42.0917	176.7587	6000.4626
RSA [39]	0.8401	0.4190	43.3812	161.5556	6034.7591
WOA [40]	0.8125	0.4375	42.0983	176.6390	6059.7410
AOA [16]	0.8304	0.4162	42.7513	169.3454	6048.7844

5. Application of MCLAOA-BPNN for Cluster Fault Prediction

Due to the rapid development of computer technology, computer systems are widely used in various industries of the national economy [54,55]. Most current software systems can be viewed as cluster systems, which are parallel or distributed systems composed of a large number of independent computers [56]. As the number of nodes in cluster systems continues to increase, the frequency of node failures also increases, which will seriously affect normal usage. In recent years, although existing research work [57,58] in cluster system fault prediction has achieved good results, further improvement is needed in terms of prediction accuracy and efficiency. Therefore, this paper proposed MCLAOA to optimize BPNN parameters and design an MCLAOA-BPNN control cluster fault prediction method.

5.1. Control Cluster System

To meet the demand of uninterrupted and reliable running of multiple computing jobs, a high-availability control cluster system is constructed. The network architecture of the system is shown in Figure 8. The high-availability cluster for multiple computing jobs consists of one central monitoring station and m computing units. Among them, the central monitoring station is responsible for simulating two virtual computers (A and B are backups of each other), completing the monitoring of the high-availability cluster. Each computing unit is responsible for simulating n virtual computers, completing the simulation of computing jobs, dynamic task allocation, migration, and other high availability cluster equipment functions. However, to ensure the sustainable operation of the cluster system, cluster fault prediction is crucial. Therefore, this paper designed a cluster fault prediction method based on BPNN, and used the proposed MCLAOA to optimize the parameters in the BPNN, thus improving the accuracy of cluster fault prediction.

For implementation details, the functions that control the cluster system were implemented using C++. Qt Creator was utilized to showcase these functions via a graphical interface, which also allowed for fault injection to observe the resource information of each PC. The proposed MCLAOA-BPNN was executed on MATLAB, providing fault prediction for the entire cluster system.

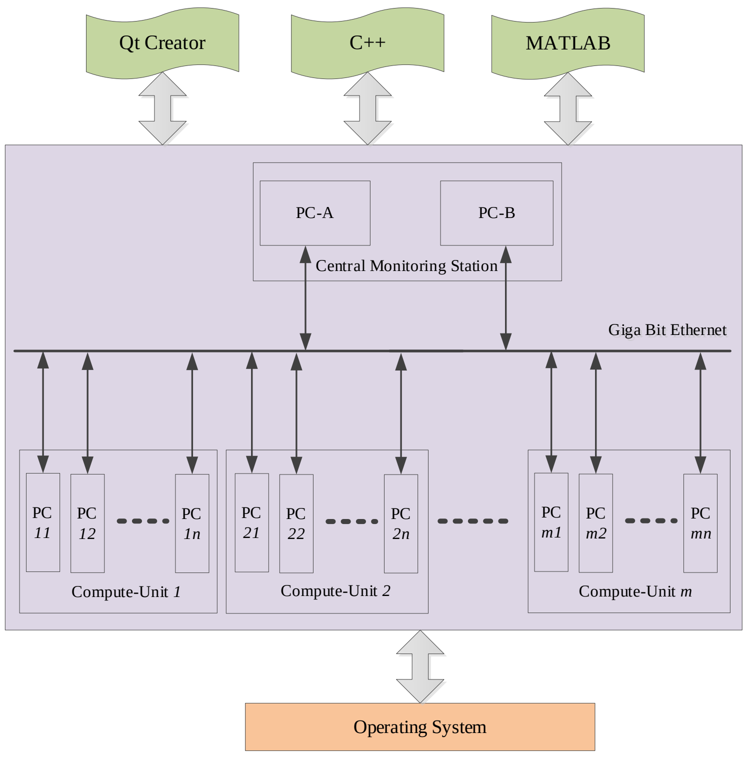

Figure 8. The network architecture of high-availability control cluster system.

5.2. Cluster Fault Prediction Based on MCLAOA-BPNN

5.2.1. BP Neural Network

The computation process of the BPNN consists of a forward computation process and a backward computation process, which includes the input layer, the hidden layer, and the output layer. The BPNN inputs the data from the input layer, processes the data in the hidden layer, and then calculates the difference between the processed data and the true data. If the obtained result does not meet the set error value, it enters the backpropagation process. During backpropagation, the weights and thresholds in each layer of neurons constantly change until the set error value or the predetermined number of training times is reached. The main process of BPNN is as follows:

Step 1: Parameter initialization: the number of nodes in the input layer, the hidden layer, and the output layer, as well as the initial weights and thresholds of each neuron.
Step 2: Forward propagation.
Step 3: Calculation of the error value between the output data and the expected data.
Step 4: Update of the weights and thresholds.
Step 5: Check whether the error value meets the set value. If not, return to **Step 4** and update the weights and thresholds until the set error value is reached or the maximum training times are reached.

5.2.2. MCLAOA Optimizes BPNN

During the training process of BPNN, the initial weights and thresholds are randomly generated, which will affect the prediction performance of the model. Therefore, this paper adopts the proposed MCLAOA to optimize the weights and thresholds in BPNN, called MCLAOA-BPNN. The flowchart of the proposed MCLAOA-BPNN cluster fault prediction method is shown in Figure 9, where the red dashed box is the proposed MCLAOA, and the blue dashed box is the MCLAOA-BPNN fault prediction model proposed in this chapter. The specific implementation process is as follows:

Step 1: Analyze the cluster system, determine the fault prediction indicators that affect the cluster system based on the network structure of the cluster system, and construct feature vectors.
Step 2: Initialize the weights and thresholds of BPNN, the parameters of MCLAOA, and read the initial index data of the cluster system as the initial sample data.
Step 3: Pre-process the sample data.
Step 4: Use the MCLAOA to optimize the weights and thresholds of the BPNN and construct the MCLAOA-BPNN fault prediction model.
Step 5: Check whether the termination condition is met. If the termination condition is met, the optimal weights and thresholds are output. Otherwise, skip to **Step 4**.
Step 6: The optimized weights and thresholds are used as the weights and thresholds of the MCLAOA-BPNN model.

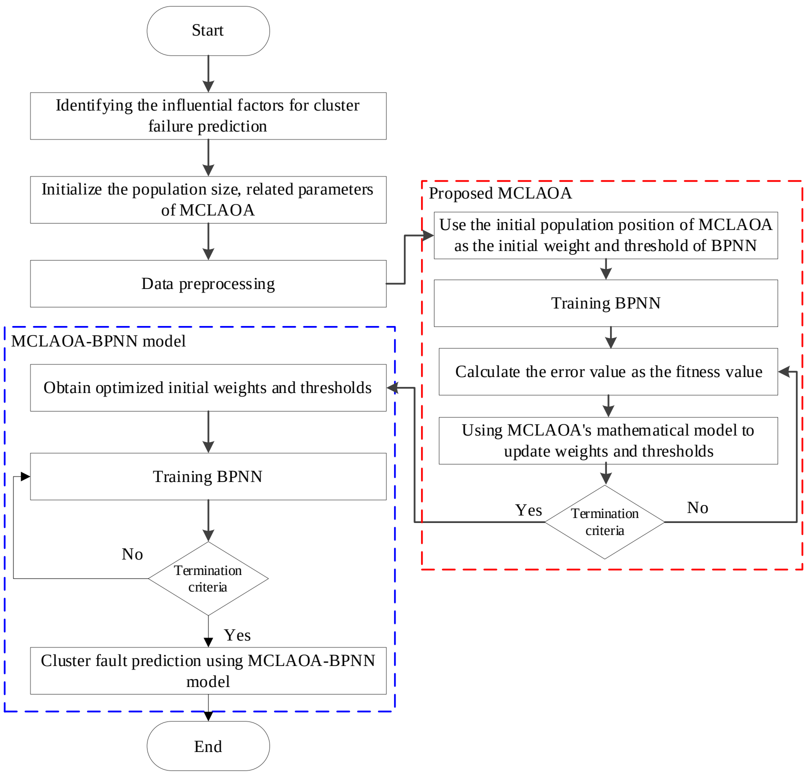
Start
Identifying the influential factors for cluster failure prediction
Initialize the population size, related parameters of MCLAOA
Data preprocessing
Proposed MCLAOA
Use the initial population position of MCLAOA as the initial weight and threshold of BPNN
Training BPNN
Calculate the error value as the fitness value
Using MCLAOA's mathematical model to update weights and thresholds
Yes
Termination criteria
No
MCLAOA-BPNN model
Obtain optimized initial weights and thresholds
Training BPNN
No
Termination criteria
Yes
Cluster fault prediction using MCLAOA-BPNN model
End

Figure 9. The flowchart of the MCLAOA-BPNN cluster fault prediction.

5.3. Experimental Results and Analysis

The high-availability cluster system constructed in this paper has a total of 42 nodes, including 2 central nodes and 40 computing nodes, i.e., $m \times n = 40$. The operating system is Ubuntu 16.04. For the proposed MCLAOA-BPNN fault prediction model, all experiments were performed on MATLAB 2017b, and they were run on a PC with Intel Core i7-10700 2.90 GHz and 16 GB RAM.

We used six main factors that affect cluster performance as sample data, including CPU consumption, memory usage, operating system processes load, net traffic, I/O operations, and number of processes. To simulate faulty behavior, we injected node failures, program errors, network faults, and performance anomalies to obtain fault data. In the data collection process, we collected sample data from 50 moments, normalized the sample data, and used 40 moments as training data and 10 moments as testing data. All data were collected from our own and benchmark (https://ieee-dataport.org/open-access/big-data-machine-learning-benchmark-spark, accessed on 6 June 2019) [59].

5.3.1. Evaluation Criteria

To better evaluate the results of the data, we employed mean absolute error (MAE), root mean square error ($RMSE$), and mean absolute percentage error ($MAPE$) as evaluation metrics for the model [60]. The MAE can provide a measure of the overall accuracy of the predictions. The $RMSE$ gives more weight to larger errors. The $MAPE$ provides a relative measure of the prediction accuracy. The MAE and the $MAPE$ are mainly used to measure the degree of difference between predicted values and true values, where the smaller the value, the higher the prediction accuracy of the model. The $RMSE$ represents the degree of fluctuation in the difference, where the smaller the value, the more stable the prediction results. The formulas for calculating the MAE, the $RMSE$, and the $MAPE$ are as follows:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - y'_i\right|,$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - y'_i)^2},$$

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i - y'_i}{y_i}\right|,$$

where N denotes the number of observations, y'_i represents the predicted value and y_i is the true value.

5.3.2. Compared Algorithms and Parametric Setup

To evaluate the performance of the proposed MCLAOA-BPNN model, we compared it with the basic BPNN model and swarm-optimized BPNN models (such as the PSO [42], the AOA [16], the CAOA [18], and the sine cosine algorithm (SCA) [61]). In all experiments, the parameter settings were as follows: all population sizes were 50, the number of iterations was 500, and other parameters were set to default values. Additionally, based on the influencing factors, the input layer of the cluster fault prediction model in this paper was set to 6 and the output layer was set to 1. However, the number of hidden layers was not specified, but it is crucial for prediction accuracy. Therefore, we trained the MCLAOA-BPNN cluster fault prediction model with a range of hidden layer numbers (5–12), and the average value of 10 test results for MAE is shown in Table 8, with the best results marked in bold type. It can be seen from Table 8 that the model's hidden layer was set to 7. Therefore, the final architecture is determined to be a three-layer MCLAOA-BPNN with a configuration of 6-7-1. Furthermore, Figure 10 demonstrates that the proposed model has converged after six epochs, where one epoch refers to the number of training iterations, representing one forward propagation and one backward propagation of the BPNN.

Table 8. Network training MAE for different numbers of hidden layer nodes.

Nodes	5	6	7	8	9	10	11	12
MAE	1.00E-03	1.01E-03	**6.14E-04**	2.46E-03	2.21E-03	1.88E-03	1.01E-03	1.51E-03

Note: The best results of each function were marked in **bold** type.

5.3.3. Comparison with other BPNN Models

To verify that the proposed model is highly competitive, we compared MCLAOA-BPNN with other fault prediction models, including the BPNN [62], the PSO-BPNN [42], the AOA-BPNN [16], the CAOA-BPNN [18], and the SCA-BPNN [61]. The errors between the predicted values and true values of different prediction models on different sample data are shown in Figure 11. It can be seen from Figure 11 that the prediction accuracy of BPNN has been improved by swarm-optimized BPNN. The proposed MCLAOA-BPNN shows significant performance, especially compared to the AOA-BPNN and the CAOA-BPNN.

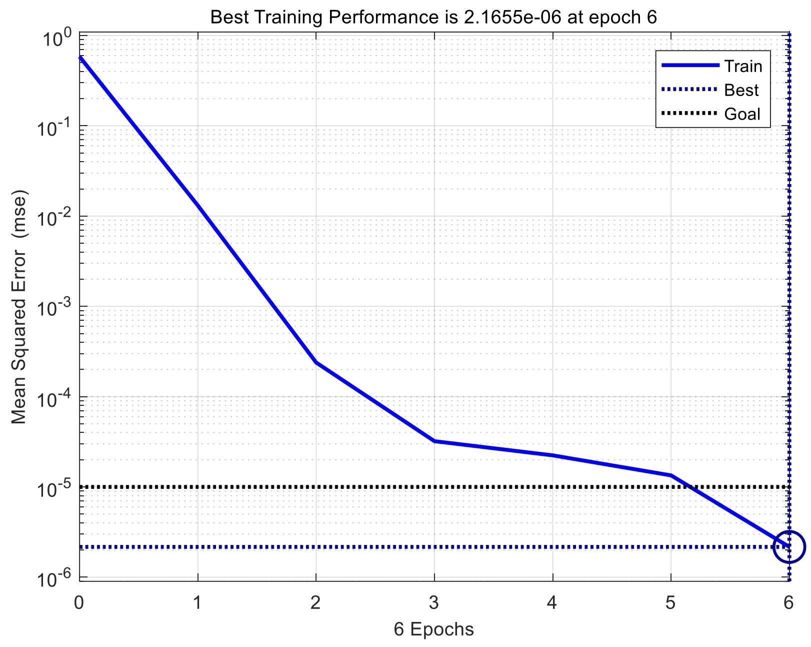

Figure 10. Number of training samples.

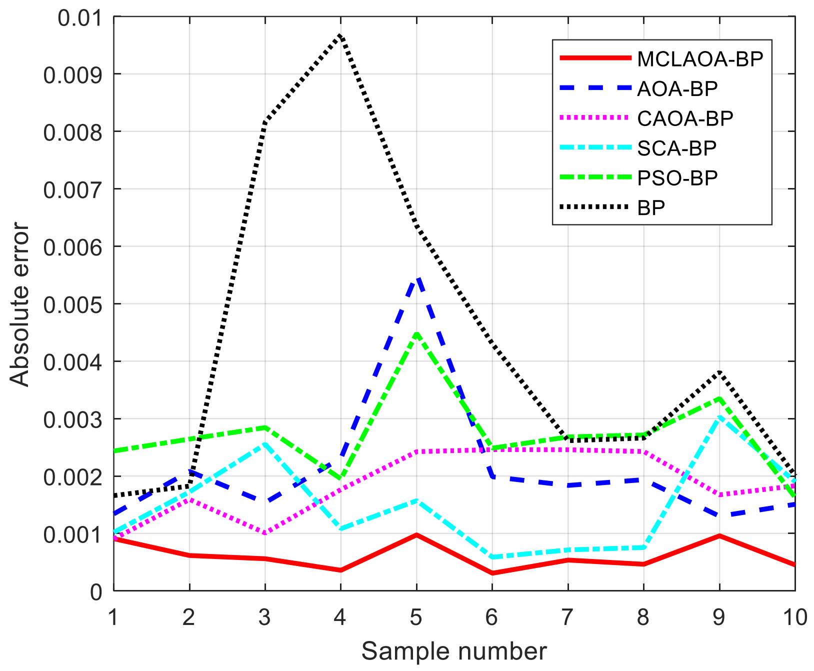

Figure 11. Comparison of predicted absolute error curves of each model.

In order to clearly observe results, MAE, $RMSE$, and $MAPE$ were employed, as shown in Table 9. From Table 9, it can be seen that compared with the AOA-BPNN and the CAOA-BPNN, the proposed MCLAOA-BPNN improves 1.526/1.236, 1.783/1.283, and 0.8762/0.6111 in terms of MAE, $RMSR$, and $MAPE$. Compared with the basic BPNN and the other swarm-optimized BPNN, the proposed MCLAOA-BPNN's prediction accuracy also ranks first. These results demonstrate that our model can better perform cluster fault prediction.

Table 9. Predictive results evaluation.

Models	MAE	RMSE	MAPE
BPNN	4.31E-03	5.08E-03	2.3614
PSO-BPNN	2.72E-03	2.82E-03	1.3876
SCA-BPNN	1.49E-03	1.68E-03	0.7221
CAOA-BPNN	1.85E-03	1.94E-03	0.9187
AOA-BPNN	2.14E-03	2.44E-03	1.1838
MCLAOA-BPNN	**6.14E-04**	**6.57E-04**	**0.3076**

Note: The best results of each function were marked in **bold** type.

6. Conclusions

In the fault prediction of control cluster systems, the improper setting of initial weights and thresholds in a traditional BPNN can lead to low accuracy. To address this issue, this paper proposes a new swarm intelligence algorithm called MCLAOA and utilizes MCLAOA to optimize the initial weights and thresholds of BPNN, constructing the MCLAOA-BPNN control cluster fault prediction model. To validate the effectiveness of the proposed MCLAOA, 23 benchmark functions, CEC2020 benchmark problems, and two engineering examples were employed. Furthermore, we compared the proposed MCLAOA-BPNN with other swarm-intelligence-based BPNN models to demonstrate its high prediction accuracy.

The following points present the specific experimental results.

- It can be observed from Table 5 that the *p*-values of the Wilcoxon rank-sum test for the compare algorithms are less than 0.05 in most functions. This indicates that the *ave* and *std* obtained by all algorithms in Table 1 have statistical significance.
- From the last two rows of Table 1, it can be observed that the Friedman rank test of MCLAOA is 2.1034, ranking first. According to the statistical results of the 23 benchmark functions in Table 4, it can be observed that the MCLAOA has improved convergence performance in 16 and 13 functions compared to the basic AOA and LSHADE, respectively.
- The convergence curve and box plots prove that MCLAOA has a faster convergence speed and better robustness.
- Scalability analysis confirms that the MCLAOA has strong and stable performance.
- From Tables A2 and A3, it can be observed that the MCLAOA exhibits significant advantages on the CEC2020 benchmark problems and demonstrates statistical significance.
- According to the experimental results of MAE, $RMSE$, and $MAPE$, compared with the basic BPNN/AOA-BPNN, the MCLAOA-BPNN improved by 3.696/1.526, 4.423/1.783, 2.0538/0.8762. Furthermore, the MCLAOA-BPNN outperforms other swarm-intelligence-based BPNN models.

The main limitations of the proposed MCLAOA are as follows: To ensure convergence speed, the MS strategy is only applied in the exploration phase of AOA. Once the algorithm falls into a local optimum during the exploitation phase, it lacks the ability to escape from it. As a result, it exhibits poor performance in handling more complex practical application scenarios such as image processing, engineering design, and other issues. Furthermore, a large number of iterations can increase the computational cost of the fault prediction model. In the future, we plan to introduce mutation operators to enhance the algorithm's ability to escape local optima. We will also design a convergence monitoring technique to determine whether the desired value has been achieved, whether the expected value is reached, and if so, whether it can terminate iterations and reduce unnecessary losses. Moreover, the proposed MCLAOA can be extended to handle structural optimization, feature selection, etc.

Author Contributions: Conceptualization, T.X. and Z.G.; Methodology, T.X. and Y.Z.; Software, T.X. and Z.G.; Validation, Z.G.; Writing—original draft, T.X.; Writing—review and editing, Z.G. and Y.Z.; Visualization, Y.Z.; Funding acquisition, Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Natural Science Foundation of China (No.62072235).

Data Availability Statement: The data used to support the findings of this study are available from the corresponding author upon request.

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

Nomenclature

The following symbols are used in this manuscript:

t	Current iteration number
T	Maximum iteration number
Max	Maximum value of the accelerated function
Min	Minimum value of the accelerated function
∂	A small integer
UB_j	Upper bound value of the jth dimension
LB_j	Lower bound value of the jth dimension
η	Control parameter
α	Sensitive parameter
S_j	Step factor
Np	Population size
p	Subpopulation size
q	Number of individuals in each subpopulation
$\arg f$	Inverse function of fitness evaluation
T_F	Teacher factor
D	Dimension
m	Number of compute-units
n	Number of PCs in each compute-unit
N	Number of observations

Appendix A

Table A1. CEC2020 benchmark problems.

Type	No.	Description	f_{min}
Unimodal functions	CF_1	Shifted and Rotated Bent Cigar Function (CEC 2017 F_1)	100
Basic functions	CF_2	Shifted and Rotated Schwefel's Function (CEC 2014 F_{11})	1100
	CF_3	Shifted and Rotated Lunacek bi-Rastrigin Function (CEC 2017 F_7)	700
	CF_4	Expanded Rosenbrock's plus Griewangk's Function (CEC2017 F_{19})	1900
Hybrid functions	CF_5	Hybrid Function 1 ($N = 3$) (CEC 2014 F_{17})	1700
	CF_6	Hybrid Function 2 ($N = 4$) (CEC 2017 F_{16})	1600
	CF_7	Hybrid Function 3 ($N = 5$) (CEC 2014 F_{21})	2100
Composition functions	CF_8	Composition Function 1 ($N = 3$) (CEC 2017 F_{22})	2200
	CF_9	Composition Function 2 ($N = 4$) (CEC 2017 F_{24})	2400
	CF_{10}	Composition Function 3 ($N = 5$) (CEC 2017 F_{25})	2500

Table A2. Numerical results of MCLAOA with other algorithms using CEC2020 benchmark problem.

Function	Criteria	MCLAOA	AOA	CAOA	RSA	WOA	GOA	SMA	TLGOA
CF_1	*ave*	4.7251E+08	6.5860E+09	3.9179E+08	1.8028E+10	2.3351E+05	2.2205E+09	**8.5186E+03**	5.8672E+06
	std	5.2285E+08	2.4988E+09	5.0399E+08	2.7575E+09	5.7869E+05	1.7592E+09	4.0192E+03	1.2834E+07
	best	1.3534E+07	1.2835E+09	1.2241E+03	4.5193E+09	1.3832E+04	1.4507E+04	289.6965	1.3823E+06
CF_2	*ave*	**1.3909E+03**	1.7983E+03	1.8092E+03	2.3823E+03	2.1676E+03	2.1102E+03	1.5779E+03	1.9750E+03
	std	130.6380	183.0234	154.0381	194.0336	287.2063	393.2515	226.7420	283.4210
	best	1.2222E+03	1.4506E+03	1.4609E+03	1.9660E+03	1.6097E+03	1.2672E+03	1.2337E+03	1.5287E+03

Table A2. *Cont.*

Function	Criteria	MCLAOA	AOA	CAOA	RSA	WOA	GOA	SMA	TLGOA
CF_3	*ave*	764.4841	804.0515	795.0880	802.4801	779.2079	8.8384E+02	**721.4592**	726.2413
	std	17.0783	6.9201	11.4560	13.2465	26.5078	67.9737	5.3659	8.0244
	best	730.7717	794.3927	765.6510	765.6002	741.4918	8.1811E+02	711.5759	711.7954
CF_4	*ave*	6.6057E+03	9.5217E+03	1.6702E+04	5.5289E+05	2.3003E+04	8.5273E+06	2.6294E+03	**1.9512E+03**
	std	4.0780E+03	7.6283E+03	1.2394E+04	7.6822E+05	4.3024E+04	1.6047E+07	1.5176E+03	50.5600
	best	2.3001E+03	2.0331E+03	1.9303E+03	9.4937E+03	2.1231E+03	3.1979E+05	1.9041E+03	1.9075E+03
CF_5	*ave*	4.2807E+03	4.9853E+04	4.2818E+03	4.5244E+05	1.3354E+05	3.2091E+04	9.4876E+03	**3.2977E+03**
	std	2.2491E+03	2.8177E+04	1.0096E+03	1.3636E+05	2.1080E+05	5.5254E+04	6.7329E+03	5.6970E+03
	best	2.2322E+03	1.1207E+04	3.1850E+03	4.4142E+04	5.1664E+03	2.7526E+03	1.9067E+03	1.8024E+03
CF_6	*ave*	1.8286E+03	1.9828E+03	1.9183E+03	2.0728E+03	1.8357E+03	3.0539E+03	**1.6663E+03**	1.8319E+03
	std	124.5011	138.4976	144.2710	93.7524	117.3325	331.0472	69.1540	132.5534
	best	1.6122E+03	1.7364E+03	1.6192E+03	1.9127E+03	1.6397E+03	2.4461E+03	1.6065E+03	1.6400E+03
CF_7	*ave*	4.1952E+03	5.6449E+03	6.2259E+03	1.7514E+05	3.2918E+04	6.9666E+03	2.8949E+03	**2.7665E+03**
	std	1.7962E+03	2.3694E+03	2.6621E+03	1.7081E+05	3.2594E+04	7.5302E+03	1.7753E+03	410.4163
	best	2.1461E+03	2.3547E+03	2.3133E+03	7.8506E+03	4.2554E+03	2.4166E+03	2.1203E+03	2.1776E+03
CF_8	*ave*	**2.3265E+03**	2.9057E+03	2.5958E+03	2.9702E+03	2.3315E+03	5.7708E+03	2.3266E+03	2.3581E+03
	std	38.6359	315.3173	117.4490	240.2314	15.3485	1.9670E+03	166.5267	237.6234
	best	2.3060E+03	2.2885E+03	2.3035E+03	2.5197E+03	2.2393E+03	2.4165E+03	2.2000E+03	2.3045E+03
CF_9	*ave*	**2.7189E+03**	2.7927E+03	2.7211E+03	2.8507E+03	2.7519E+03	3.0282E+03	2.7475E+03	2.7426E+03
	std	103.1237	88.3431	137.8636	44.4965	77.6254	52.7887	47.5504	102.2214
	best	2.5337E+03	2.5774E+03	2.5000E+03	2.7494E+03	2.5045E+03	2.9284E+03	2.5000E+03	2.4563E+03
CF_{10}	*ave*	**2.9229E+03**	3.2008E+03	2.9648E+03	2.9232E+03	2.9583E+03	2.9573E+03	2.9688E+03	2.9410E+03
	std	49.2665	145.6641	35.1046	23.0547	31.1292	62.4765	55.9072	29.2071
	best	2.8138E+03	2.9688E+03	2.8996E+03	3.0303E+03	2.9011E+03	2.8886E+03	2.8979E+03	2.8986E+03
Friedman rank test		2.3000	5.6000	4.3000	7.4000	4.7000	6.6000	2.6000	2.5000
rank		1	6	4	8	5	7	3	2

Note: The best results of each function were marked in **bold** type in terms of *ave*.

Table A3. *p*-values of the Wilcoxon rank-sum test over CEC2020 benchmark problem.

Function	MCLAOA	AOA	CAOA	RSA	WOA	GOA	SMA	TLGOA
CF_1	3.0199E-11	3.0199E-11	5.9706E-05	3.0199E-11	3.0199E-11	3.0199E-11	N/A	3.0199E-11
CF_2	N/A	6.7220E-10	1.7769E-10	3.0199E-11	4.0772E-11	1.1737E-09	4.7138E-04	2.6099E-10
CF_3	3.3384E-11	3.0199E-11	3.0199E-11	3.0199E-11	3.0199E-11	3.0199E-11	N/A	0.0170
CF_4	3.0199E-11	4.9752E-11	1.4643E-10	3.0199E-11	3.0199E-11	3.0199E-11	0.8534	N/A
CF_5	6.5277E-08	7.3891E-11	1.0702E-09	3.0199E-11	1.4643E-10	5.5727E-10	8.8829E-06	N/A
CF_6	5.0912E-06	9.9186E-11	3.8249E-09	3.0199E-11	9.5139E-06	3.0199E-11	N/A	1.7294E-07
CF_7	0.0163	4.1127E-07	5.5999E-07	3.0199E-11	3.0199E-11	7.0881E-08	0.0032	N/A
CF_8	N/A	5.5727E-10	8.8910E-10	3.0199E-11	0.1453	3.3384E-11	6.1210E-10	0.3871
CF_9	N/A	5.8737E-04	0.4463	2.2273E-09	0.7958	3.0199E-11	0.0657	0.7845
CF_{10}	N/A	7.3891E-11	2.2539E-04	3.3384E-11	1.1058E-04	0.0905	2.6806E-04	0.0281

Note: N/A represents the best algorithm in terms of optimization performance among all the algorithms for the corresponding function.

References

1. Saxena, D.; Gupta, I.; Singh, A.K.; Lee, C.N. A fault tolerant elastic resource management framework toward high availability of cloud services. *IEEE Trans. Netw. Serv. Manag.* **2022**, *19*, 3048–3061. [CrossRef]
2. Somasekaram, P.; Calinescu, R.; Buyya, R. High-availability clusters: A taxonomy, survey, and future directions. *J. Syst. Softw.* **2022**, *187*, 111208. [CrossRef]
3. Reisizadeh, A.; Prakash, S.; Pedarsani, R.; Avestimehr, A.S. Coded computation over heterogeneous clusters. *IEEE Trans. Inf. Theory* **2019**, *65*, 4227–4242. [CrossRef]
4. Wael, K.; Jingwei, H. Cluster resource scheduling in cloud computing: Literature review and research challenges. *J. Supercomput.* **2022**, *78*, 6898–6943.

5. Arunarani, A.; Manjula, D.; Sugumaran, V. Task scheduling techniques in cloud computing: A literature survey. *Future Gener. Comput. Syst.* **2019**, *91*, 407–415. [CrossRef]
6. Jena, U.K.; Das, P.K.; Kabat, M.R. Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment—ScienceDirect. *J. King Saud Univ. Comput. Inf. Sci.* **2022**, *34*, 2332–2342.
7. Ghomi, E.J.; Rahmani, A.M.; Qader, N.N. Load-balancing Algorithms in Cloud Computing: A Survey. *J. Netw. Comput. Appl.* **2017**, *88*, 50–71. [CrossRef]
8. Luo, Q.; Hu, S.; Li, C.; Li, G.; Shi, W. Resource Scheduling in Edge Computing: A Survey. *IEEE Commun. Surv. Tutorials* **2021**, *23*, 2131–2165. [CrossRef]
9. Amin, A.A.; Hasan, K.M. A review of Fault Tolerant Control Systems: Advancements and applications. *Measurement* **2019**, *143*, 58–68. [CrossRef]
10. Abbaspour, A.; Mokhtari, S.; Sargolzaei, A.; Yen, K.K. A Survey on Active Fault-Tolerant Control Systems. *Electronics* **2020**, *9*, 1513. [CrossRef]
11. Pinto, J.; Jain, P.; Kumar, T. Hadoop Distributed Computing Clusters for Fault Prediction. In Proceedings of the 2016 International Computer Science and Engineering Conference (ICSEC), Chiang Mai, Thailand, 14–17 December 2016; pp. 1–6.
12. Mukwevho, M.A.; Celik, T. Toward a Smart Cloud: A Review of Fault-Tolerance Methods in Cloud Systems. *IEEE Trans. Serv. Comput.* **2021**, *14*, 589–605. [CrossRef]
13. Das, D.; Schiewe, M.; Brighton, E.; Fuller, M.; Cerny, T.; Bures, M.; Frajtak, K.; Shin, D.; Tisnovsky, P. *Failure Prediction by Utilizing Log Analysis: A Systematic Mapping Study*; Association for Computing Machinery: New York, NY, USA, 2020.
14. Bacanin, N.; Stoean, R.; Zivkovic, M.; Petrovic, A.; Rashid, T.A.; Bezdan, T. Performance of a novel chaotic firefly algorithm with enhanced exploration for tackling global optimization problems: Application for dropout regularization. *Mathematics* **2021**, *9*, 2705. [CrossRef]
15. Malakar, S.; Ghosh, M.; Bhowmik, S.; Sarkar, R.; Nasipuri, M. A GA based hierarchical feature selection approach for handwritten word recognition. *Neural Comput. Appl.* **2020**, *32*, 2533–2552. [CrossRef]
16. Abualigah, L.; Diabat, A.; Mirjalili, S.; Abd Elaziz, M.; Gandomi, A.H. The arithmetic optimization algorithm. *Comput. Methods Appl. Mech. Eng.* **2021**, *376*, 113609. [CrossRef]
17. Yıldız, B.S.; Patel, V.; Pholdee, N.; Sait, S.M.; Bureerat, S.; Yıldız, A.R. Conceptual comparison of the ecogeography-based algorithm, equilibrium algorithm, marine predators algorithm and slime mold algorithm for optimal product design. *Mater. Test.* **2021**, *63*, 336–340. [CrossRef]
18. Li, X.D.; Wang, J.S.; Hao, W.K.; Zhang, M.; Wang, M. Chaotic arithmetic optimization algorithm. *Appl. Intell.* **2022**, *52*, 16718–16757. [CrossRef]
19. Çelik, E. IEGQO-AOA: Information-Exchanged Gaussian Arithmetic Optimization Algorithm with Quasi-opposition learning. *Knowl. Based Syst.* **2023**, *260*, 110169. [CrossRef]
20. Abualigah, L.; Ewees, A.A.; Al-qaness, M.A.; Elaziz, M.A.; Yousri, D.; Ibrahim, R.A.; Altalhi, M. Boosting arithmetic optimization algorithm by sine cosine algorithm and levy flight distribution for solving engineering optimization problems. *Neural Comput. Appl.* **2022**, *34*, 8823–8852. [CrossRef]
21. Mohamed, A.A.; Abdellatif, A.D.; Alburaikan, A.; Khalifa, H.A.E.W.; Elaziz, M.A.; Abualigah, L.; AbdelMouty, A.M. A novel hybrid arithmetic optimization algorithm and salp swarm algorithm for data placement in cloud computing. *Soft Comput.* **2023**, *27*, 5769–5780. [CrossRef]
22. Rajagopal, R.; Karthick, R.; Meenalochini, P.; Kalaichelvi, T. Deep Convolutional Spiking Neural Network optimized with Arithmetic optimization algorithm for lung disease detection using chest X-ray images. *Biomed. Signal Process. Control.* **2023**, *79*, 104197. [CrossRef]
23. Gölcük, İ.; Ozsoydan, F.B.; Durmaz, E.D. An improved arithmetic optimization algorithm for training feedforward neural networks under dynamic environments. *Knowl. Based Syst.* **2023**, *263*, 110274. [CrossRef]
24. Shirazi, M.I.; Khatir, S.; Benaissa, B.; Mirjalili, S.; Wahab, M.A. Damage assessment in laminated composite plates using modal Strain Energy and YUKI-ANN algorithm. *Compos. Struct.* **2023**, *303*, 116272. [CrossRef]
25. Kaveh, A.; Hamedani, K.B. Improved arithmetic optimization algorithm and its application to discrete structural optimization. *Structures* **2022**, *35*, 748–764. [CrossRef]
26. Salimi, H. Stochastic fractal search: A powerful metaheuristic algorithm. *Knowl. Based Syst.* **2015**, *75*, 1–18. [CrossRef]
27. Mirjalili, S. Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Comput. Appl.* **2016**, *27*, 1053–1073. [CrossRef]
28. Guha, D.; Roy, P.; Banerjee, S. Quasi-oppositional symbiotic organism search algorithm applied to load frequency control. *Swarm Evol. Comput.* **2017**, *33*, 46–67. [CrossRef]
29. Truong, K.H.; Nallagownden, P.; Baharudin, Z.; Vo, D.N. A quasi-oppositional-chaotic symbiotic organisms search algorithm for global optimization problems. *Appl. Soft Comput.* **2019**, *77*, 567–583. [CrossRef]
30. Mahajan, S.; Abualigah, L.; Pandit, A.K.; Altalhi, M. Hybrid Aquila optimizer with arithmetic optimization algorithm for global optimization tasks. *Soft Comput.* **2022**, *26*, 4863–4881. [CrossRef]
31. Lalama, Z.; Boulfekhar, S.; Semechedine, F. Localization optimization in wsns using meta-heuristics optimization algorithms: A survey. *Wirel. Pers. Commun.* **2022**, *122*, 1197–1220. [CrossRef]

32. Gad, A.G. Particle swarm optimization algorithm and its applications: A systematic review. *Arch. Comput. Methods Eng.* **2022**, *29*, 2531–2561. [CrossRef]
33. Rahman, M.A.; Sokkalingam, R.; Othman, M.; Biswas, K.; Abdullah, L.; Abdul Kadir, E. Nature-inspired metaheuristic techniques for combinatorial optimization problems: Overview and recent advances. *Mathematics* **2021**, *9*, 2633. [CrossRef]
34. Dhal, K.G.; Sasmal, B.; Das, A.; Ray, S.; Rai, R. A Comprehensive Survey on Arithmetic Optimization Algorithm. *Arch. Comput. Methods Eng.* **2023**, *30*, 3379–3404. [CrossRef] [PubMed]
35. Zhang, H.; Gao, Z.; Zhang, J.; Liu, J.; Nie, Z.; Zhang, J. Hybridizing extended ant lion optimizer with sine cosine algorithm approach for abrupt motion tracking. *EURASIP J. Image Video Process.* **2020**, *2020*, 4. [CrossRef]
36. Gao, Z.; Zhuang, Y.; Chen, C.; Wang, Q. Hybrid modified marine predators algorithm with teaching-learning-based optimization for global optimization and abrupt motion tracking. *Multimed. Tools Appl.* **2023**, *82*, 19793–19828. [CrossRef]
37. Liang, J.J.; Qin, A.K.; Suganthan, P.N.; Baskar, S. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Trans. Evol. Comput.* **2006**, *10*, 281–295. [CrossRef]
38. Rao, R.V.; Savsani, V.J.; Vakharia, D. Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. *Comput. Aided Des.* **2011**, *43*, 303–315. [CrossRef]
39. Abualigah, L.; Abd Elaziz, M.; Sumari, P.; Geem, Z.W.; Gandomi, A.H. Reptile Search Algorithm (RSA): A nature-inspired meta-heuristic optimizer. *Expert Syst. Appl.* **2022**, *191*, 116158. [CrossRef]
40. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
41. Saremi, S.; Mirjalili, S.; Lewis, A. Grasshopper optimisation algorithm: Theory and application. *Adv. Eng. Softw.* **2017**, *105*, 30–47. [CrossRef]
42. Kennedy, J.; Eberhart, R. Particle Swarm Optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; pp. 1942–1948.
43. Tanabe, R.; Fukunaga, A.S. Improving the Search Performance of SHADE Using Linear Population Size Reduction. In Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, China, 6–11 July 2014; pp. 1658–1665.
44. 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]
45. Faramarzi, A.; Heidarinejad, M.; Stephens, B.; Mirjalili, S. Equilibrium optimizer: A novel optimization algorithm. *Knowl. Based Syst.* **2020**, *191*, 105190. [CrossRef]
46. Naruei, I.; Keynia, F. Wild horse optimizer: A new meta-heuristic algorithm for solving engineering optimization problems. *Eng. Comput.* **2022**, *38*, 3025–3056. [CrossRef]
47. Derrac, J.; García, S.; Molina, D.; Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Expert Syst. Appl.* **2011**, *1*, 3–18. [CrossRef]
48. Yue, C.T.; Price, K.V.; Suganthan, P.N.; Liang, J.J.; Ali, M.Z.; Qu, B.Y.; Awad, N.H.; Biswas.; P.P. Problem Definitions and Evaluation Criteria for the CEC 2020 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization. In Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion, Cancún, Mexico, 8–12 July 2020.
49. Li, S.; Chen, H.; Wang, M.; Heidari, A.A.; Mirjalili, S. Slime mould algorithm: A new method for stochastic optimization. *Future Gener. Comput. Syst.* **2020**, *111*, 300–323. [CrossRef]
50. Zhang, H.; Gao, Z.; Ma, X.; Zhang, J.; Zhang, J. Hybridizing teaching-learning-based optimization with adaptive grasshopper optimization algorithm for abrupt motion tracking. *IEEE Access* **2019**, *7*, 168575–168592. [CrossRef]
51. Zheng, R.; Jia, H.; Abualigah, L.; Liu, Q.; Wang, S. An improved arithmetic optimization algorithm with forced switching mechanism for global optimization problems. *Math. Biosci. Eng* **2022**, *19*, 473–512. [CrossRef]
52. Yang, Y.; Gao, Y.; Tan, S.; Zhao, S.; Wu, J.; Gao, S.; Zhang, T.; Tian, Y.C.; Wang, Y.G. An opposition learning and spiral modelling based arithmetic optimization algorithm for global continuous optimization problems. *Eng. Appl. Artif. Intell.* **2022**, *113*, 104981. [CrossRef]
53. Faramarzi, A.; Heidarinejad, M.; Mirjalili, S.; Gandomi, A.H. Marine Predators Algorithm: A nature-inspired metaheuristic. *Expert Syst. Appl.* **2020**, *152*, 113377. [CrossRef]
54. Catal, C. Software fault prediction: A literature review and current trends. *Expert Syst. Appl.* **2011**, *38*, 4626–4636. [CrossRef]
55. Ahmed, Q.; Raza, S.A.; Al-Anazi, D.M. Reliability-based fault analysis models with industrial applications: A systematic literature review. *Qual. Reliab. Eng. Int.* **2021**, *37*, 1307–1333. [CrossRef]
56. Agrawal, A. Concepts for distributed systems design. *Proc. IEEE* **1986**, *74*, 236. [CrossRef]
57. Shafiq, M.; Alghamedy, F.H.; Jamal, N.; Kamal, T.; Daradkeh, Y.I.; Shabaz, M. Scientific programming using optimized machine learning techniques for software fault prediction to improve software quality. *IET Software* **2023**, 1–11. [CrossRef]
58. Tameswar, K. Towards Optimized K Means Clustering Using Nature-Inspired Algorithms for Software Bug Prediction. Available online: https://ssrn.com/abstract=4358066 (accessed on 14 February 2023).
59. Jairson, R.; Germano, V. Big Data Machine Learning Benchmark on Spark. *IEEE Dataport* **2019**. [CrossRef]
60. Al-Musaylh, M.S.; Deo, R.C.; Li, Y.; Adamowski, J.F. Two-phase particle swarm optimized-support vector regression hybrid model integrated with improved empirical mode decomposition with adaptive noise for multiple-horizon electricity demand forecasting. *Appl. Energy* **2018**, *217*, 422–439. [CrossRef]

61. Mirjalili, S. SCA: A sine cosine algorithm for solving optimization problems. *Knowl.-Based Syst.* **2016**, *96*, 120–133. [CrossRef]
62. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]