IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 9, 2021, accepted November 22, 2021, date of publication November 25, 2021,
date of current version December 13, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3130640

An Improved Cuckoo Search Algorithm Utilizing Nonlinear Inertia Weight and Differential Evolution for Function Optimization Problem

CHENG-XU ZHANG[1], KAI-QING ZHOU[1], SHAO-QIANG YE[1], AND AZLAN MOHD ZAIN[2], (Member, IEEE)

[1]College of Information Science and Engineering, Jishou University, Jishou 416000, China
[2]Big Data Centre, Universiti Teknologi Malaysia, Skudai, Johor 80310, Malaysia

Corresponding author: Kai-Qing Zhou (kqzhou@jsu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 62066016; in part by the Natural Science Foundation of Hunan Province, China, under Grant 2020JJ5458; and in part by the Jishou University Graduate Research and Innovation Project JDY20016.

ABSTRACT This paper proposes an improved cuckoo search (CS) algorithm combining nonlinear inertial weight and differential evolution algorithm (WCSDE) to overcome the shortcomings of the CS algorithm, such as low convergence accuracy, lack of information exchange within the population, and inadequate local search capabilities. Compared with other CS variants, two strategies are proposed in this paper to improve the properties of the WCSDE. On the one hand, a non-linearly decreasing inertia weight with the number of evolutionary iterations is employed in the WCSDE to improve the update method of the bird's nest position, enhance the balance between the exploration and development capabilities, and strengthen the local optimization capability. On the other hand, the mutation and cross-selection mechanisms of the differential evolution (DE) algorithm are introduced to make up for the lack of the mutual relationship between the populations, avoid the loss of practical information, and increase the convergence accuracy. In the experiment part, 13 classic benchmark functions are selected to execute the function optimization tasks among the standard CS, the WCSDE, and other four CS variants to verify the effectiveness of the proposed algorithm from two aspects. The results and corresponding statistical analysis reveal that the proposed algorithm has better global search ability and strengthener robustness.

INDEX TERMS Cuckoo search algorithm, differential evolution algorithm, nonlinear inertia weight, adaptive adjustment strategy, function optimization.

I. INTRODUCTION

Swarm intelligence (SI) algorithm is a new optimization method to obtain the optimal solution of complex optimization problems by simulating social animals' group behavior and utilizing the information transmission and cooperation among individuals in the population. Cuckoo search (CS) algorithm is a novel SI algorithm, proposed by Yang and Deb, to simulate the cuckoo's nest parasitic brooding behavior combined with the Levy flight behavior [1]. Compared with other SI algorithms (such as the genetic algorithm (GA) [2], the ant colony optimization (ACO) algorithm [3], the particle swarm optimization (PSO) algorithm [4]), the CS algorithm has some unique advantages, like self-organization and parallelism, strong generality, simple operations, few parameters, and strong global searchability. At present, the applications of CS algorithm have gone deep into various disciplines and fields, such as NP-complete problem [5], [6], engineering design (mainly including three aspects of process planning [7], [8], parameter estimation [9] and prediction recognition [10]), power energy [11], machine learning [12], image processing [13] and other fields.

To overcome the inherent bottlenecks of CS, such as low convergence accuracy, poor local optimization ability, and easy to fall into local optimization, a series of improvement measures have been used to improve the performance

The associate editor coordinating the review of this manuscript and approving it for publication was Amjad Anvari-Moghaddam.

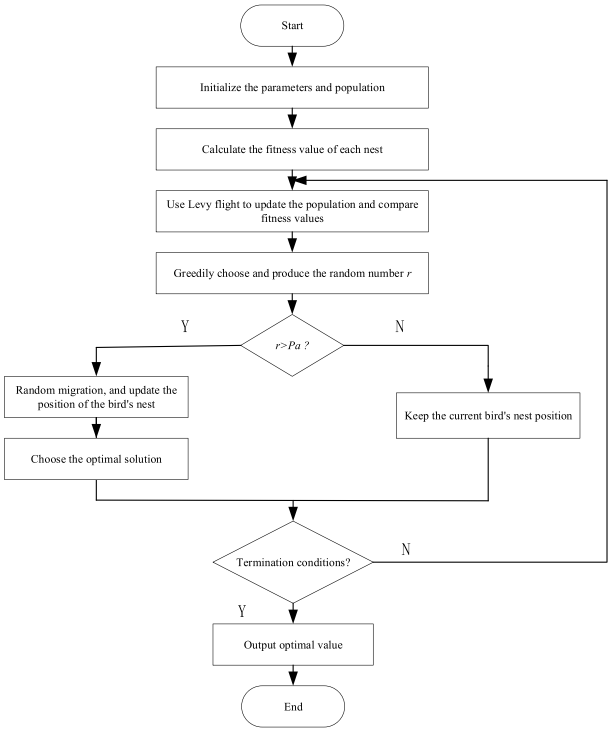


FIGURE 1. The entire CS flowchart.

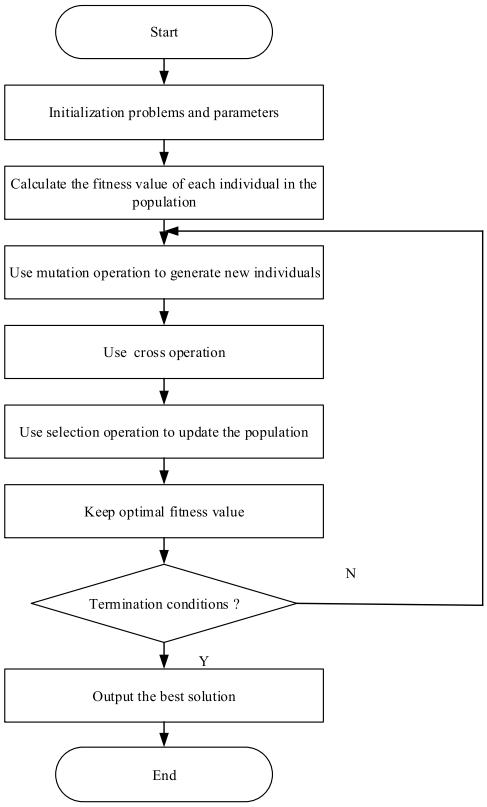


FIGURE 2. The entire DE flowchart.

of standard CS. Zhang *et al.* propose a modified adaptive cuckoo search (MACS) to adjust the search range of different stages and obtain more information in the solution space through utilizing the group and parallelize the population strategy, incentives mechanism, and adaptive step-size method. Through nine benchmark functions to verify the algorithm's performance, MACS is better than the basic CS algorithm on most test problems and has application potential in practical problems [14]. Walton *et al.* [15] use an adaptive value to replace the step factor of Levy flight to ensure the step size will decrease while the number of iterations increases, the local search capability in the later stage of the algorithm has been remarkably enhanced. The algorithm is easy to execute by adjusting two parameters and owns a high convergence speed. Valian *et al.* propose a modified CS using an adaptive step size scaling factor that changes with iterations to optimize complex engineering problems. The algorithm has been tested on four well-known reliability optimization problems, and the simulation results are better than other methods [16]. To converge to the globally optimal solution, Xue and Deng [17] amend an improved CS, which sorts the population into three subgroups with different flight scales due to the average fitness value of the individuals in the population. Simulations show that the algorithm has good optimization performance. Mlakar *et al.* give a hybrid adaptive CS, which uses the adaptive control parameters and the linear reduction of the population size to broaden the original CS. The algorithm has been tested on 30 benchmark functions in the cec2014 test set, and the effect is comparable to some powerful variants of CS [18]. The above methods all improve the relevant parameters of the CS. It can be seen from the experimental results that it has a certain effect, but there is still much room for improvement. For example, only dynamic adjustment of the parameters cannot make up for the CS algorithm's lack of information exchange and other problems.

Rani *et al.* employ a linear weight coefficient to enhance the standard CS. Experimental results reveal that the proposed algorithm also far exceeds other modern EA competitors, including standard CS, PSO, and GA [19]. Tuba *et al.* display a novel CS framework, which replaces one of the randomly selected individuals in the random transfer operator formula with one of the individuals sorted according to the fitness value matrix. The optimization experiments of eight benchmark functions show that the results are better than the basic cuckoo search algorithm [20]. Cheng *et al.* present a revised CS by adding a random movement operator and adjust the parameters by an improvement rate to the standard CS for dynamical selecting and updating the rules of the population. It has been tested with six CS variants on 42 benchmark functions over different dimensions, and the results show that the proposed algorithm is a competitive method [21]. Li *et al.* [22] introduce the orthogonal learning strategy to improve the local search ability of the cuckoo search algorithm, and 23 benchmark functions are used to verify the method's performance. Nguyen strengthens

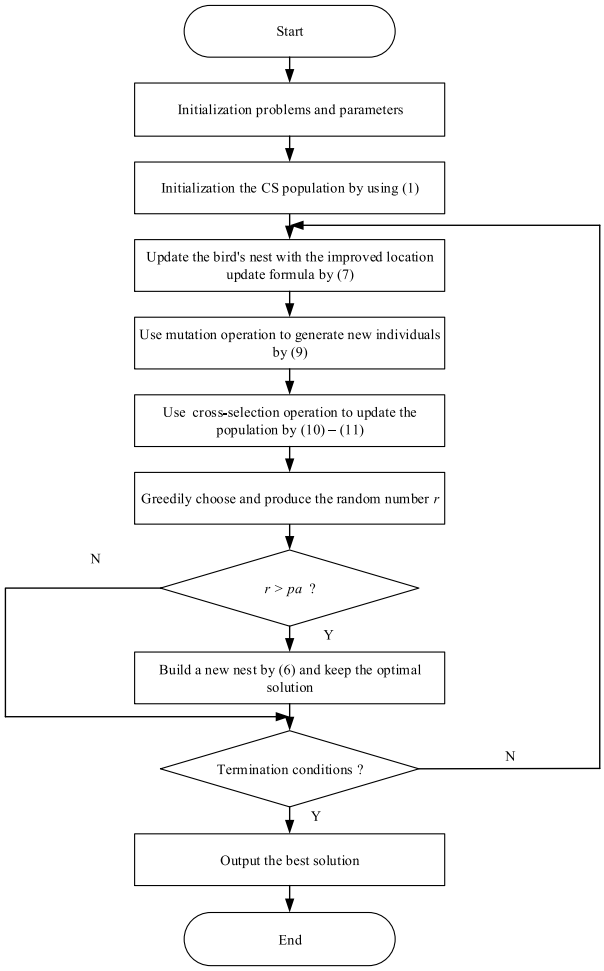


FIGURE 3. The entire WCSDE flowchart.

the local search performance and increases the diversity of the solution of the standard CS through generating some new candidate solutions at the current optimal solution and apply the modified algorithm to the reconstruction of the distribution network. The calculation results from the simple distribution network to complex distribution network show that compared with other improved CS, this algorithm can efficiently and accurately find the global optimal solution and the total power loss is smaller [23]. Li *et al.* designed a CS-differential evolution (DE) framework by introducing the mutation operator, crossover operator, and selection operator and creating two novel mutation rules based on a random number and best individual to enhance the diversity of the population. The accuracy and performance of the proposed method are evaluated through 18 classic benchmark functions and the cec2013 test suite. The results show that the proposed algorithm can balance the exploration and development of the algorithm [24]. Zhang *et al.* execute a hybrid algorithm by combining CS and DE (CSDE) to resolve constrained engineering problems. The proposed hybrid algorithm divides the population into two subgroups and uses CS and DE for these two subgroups. This strategy can complement their shortcomings, avoid premature convergence, and quickly find the globally optimal solution [25]. Wang *et al.* propose a new quantum chaotic CS algorithm (QCCS) to reinforce the global search capability and accelerate search speed using the chaotic mapping for population initialization and the non-homogeneous quantum update technology. Experimental results on six well-known real-life data sets show that the proposed QCCS is significantly better than the recent eight well-known algorithms, including hybrid cuckoo search, differential evolution, and hybrid K-means, etc. [26]. Liu and Fu mix the CS and the shuffled frog-leaping algorithm (SLFA) to expedite the speed and raise convergence accuracy by using the local search mechanism of the SLFA. In addition, the algorithm proved to be convergent [27]. Inspired by the particle swarm algorithm (PSO), Li and Yin suggest a novel CS variant by adding neighborhood information to the new population provides multiple candidate solutions for increasing the diversity of the algorithm. Thirty benchmark functions are selected in the literature, and the results are significantly better than CS and PSO [28]. In addition, Lim *et al.* raise a hybrid cuckoo search-genetic algorithm (CSGA) by employing the reproduction behavior and the evolution strategy and applying it to optimize the hole processing process. The results show that CSGA is better than ACO, PSO, immune-based algorithm, and CS [29]. The above methods show that the method of mixing multiple algorithms has a good convergence effect and has gradually become a mainstream method. Combining the advantages of a specific algorithm into the CS algorithm to compensate for the inherent shortcomings of the CS algorithm can effectively improve the algorithm's performance.

Compared with the traditional CS, the mentioned CS variants have better performance and more precise convergence accuracy to a certain extent. Nevertheless, these variants still have some drawbacks, such as weaker local search ability and deficient information exchange among the populations. Inspired by the findings above, an improved CS algorithm combining a nonlinear inertia weight and differential evolution (WCSDE) is proposed to enhance the effective information interaction and strengthen the local search ability based on the following motivations.

On the one hand, the nonlinear inertia weight mechanism is applied to modify the bird's nest update formula for dynamically adjusts the extent of flight movement. With the increase of iterations, the inertia weight will continuously reduce the extent of movement to ensure the balance between development and exploration and strengthen the local optimization ability in the later stage of the algorithm.

On the other hand, the mutation and cross-selection mechanisms of DE are selected to perform mutation operation on its position after updating the bird's nest and then seeking the optimal fitness value using the cross-select operation.

This proposed WCSDE makes up for the shortcomings of information exchange between populations in the standard CS algorithm and enhances information utilization to obtain better convergence accuracy. In experimental part, 13 classic

TABLE 1. 13 selected benchmark functions.

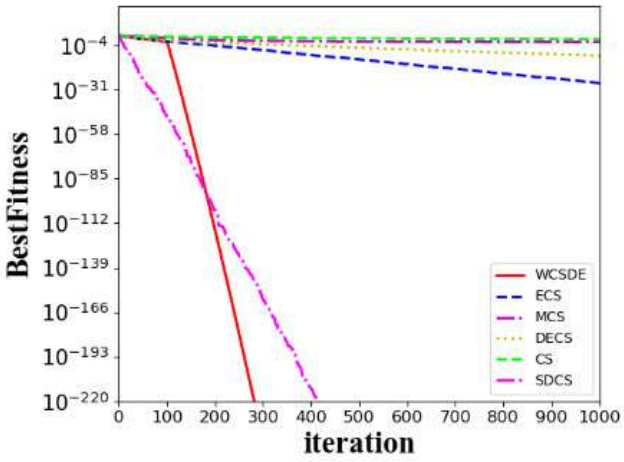
No	Function	Formula	Dim	Interval	$f(x)_{min}$
F1	Sphere	$F_1=\sum_{i=1}^{D}x_i^2$	30,50	[-5.12,5.12]	0
F2	Griewank	$F_2=\sum_{i=1}^{D}\frac{x_i^2}{4000}-\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	30,50	[-600,600]	0
F3	Rastrigin	$F_3=\sum_{i=1}^{D}\left(x_i^2-10\cos(2\pi x_i)+10\right)$	30,50	[-600,600]	0
F4	Ackley	$F_4=-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right)-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right)+20+e$	30,50	[-32,32]	0
F5	Schwefel2.22	$F_5=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert$	30,50	[-10,10]	0
F6	Bohachevsky	$F_6=\sum_{i=1}^{D-1}\begin{bmatrix}x_i^2+2x_{i+1}^2-0.3\cos(3\pi x_i)\\-0.4\cos(4\pi x_{i+1})+0.7\end{bmatrix}$	30,50	[-15,15]	0
F7	Quartic	$F_7=\sum_{i=1}^{D}ix_i^4+rand()$	30,50	[-100,100]	0
F8	Sum Squares	$F_8=\sum_{i=1}^{D}ix_i^2$	30,50	[-100,100]	0
F9	Elliptic	$F_9=\sum_{i=1}^{N}(10^6)^{\frac{i-1}{D-1}}x_i^2$	30,50	[-100,100]	0
F10	Rosenbrock	$F_{10}=\sum_{i=1}^{D-1}\left[100(x_{i+1}-x_i^2)^2+(x_i-1)^2\right]$	30,50	[-30,30]	0
F11	Alpine	$F_{11}=\sum_{i=1}^{D}\lvert x_i\sin(x_i)+0.1x_i\rvert$	30,50	[-10,10]	0
F12	Powell	$F_{12}=\sum_{i=1}^{d/4}\begin{bmatrix}(x_{4i-3}+10x_{4i-2})^2+5(x_{4i-1}-x_{4i})^2\\+(x_{4i-2}-2x_{4i-1})^4+10(x_{4i-3}-x_{4i})^4\end{bmatrix}$	30,50	[-4,5]	0
F13	Step	$F_{13}=\sum_{i=1}^{D}\left[x_i+0.5\right]^2$	30,50	[-100,100]	0

TABLE 2. Parameter settings of each CS variant and standard CS.

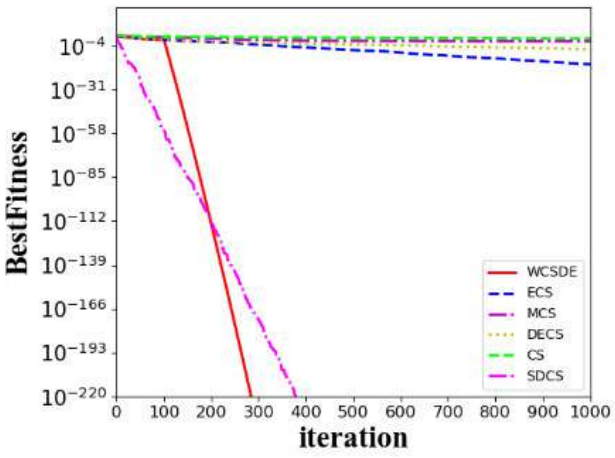
Standard CS and CS Variants	Parameters
WCSDE	$N=35,\lambda=1.5,\alpha=0.01,w_0=1.5,t_0=100,F=0.6,CR=0.8,and\ P_a=0.25$
CS	$N=35,\lambda=1.5,\alpha=0.01,and\ P_a=0.25$
ECS	$N=35,\lambda=1.5,\alpha=0.01,and\ P_a=0.25$
MCS	$N=35,\lambda=1.5,\alpha_{\min}=0.1,\alpha_{\max}=1.5,and\ P_a=0.25$
DECS	$N=35,\lambda=1.5,\alpha=0.01,P_a=0.25,F=0.5,and\ CR=0.8$
SDCS	$N=35,\lambda=1.5,\alpha=0.01,J=0.3,w=0.005$ $Pm=\frac{Se}{N\times 2},Pa=\begin{cases}\max(0,Pm-w) & if\ Pm<0.5\\ \min(1,Pm+w) & if\ Pm>0.5\end{cases}$

benchmark functions are selected to execute function optimization tasks using the standard CS [1], the WCSDE, and other four CS variants (ECS [32], MCS [33], DECS [25], and SDCS [34]) for verifying the optimization ability of the WCSDE algorithm. Experimental results show that the performance of the WCSDE algorithm is much better than other variants after numerical and statistical analysis.

The rest parts of this article are organized below. Section 2 recalls the standard CS and the standard DE briefly. Section 3 illustrates the main implementation steps

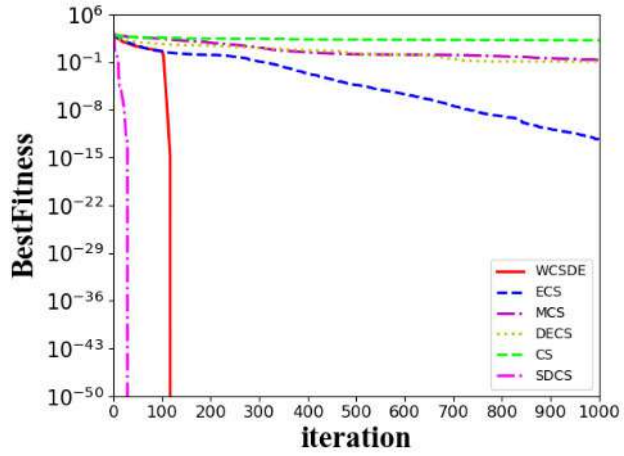


(a) 30-dimension situation

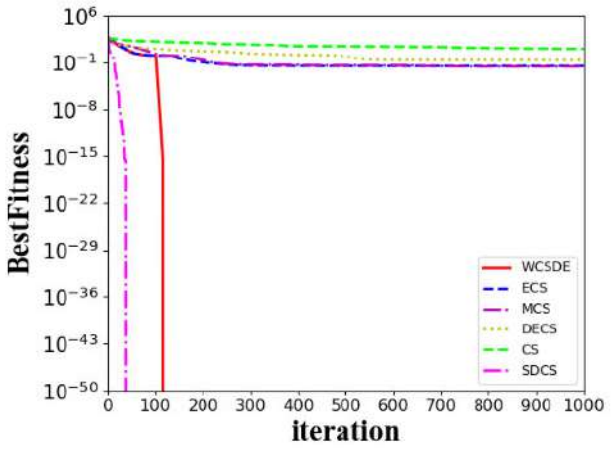


(b) 50-dimension situation

FIGURE 4. Performance comparison on F1.



(a) 30-dimension situation



(b) 50-dimension situation

FIGURE 5. Performance comparison on F2.

and explores the time complexity of the WCSDE in detail. Section 4 analyzes the performance of the proposed algorithm by testing 13 classic benchmark functions from different viewpoints. Section 5 summarizes the whole manuscript.

II. STANDARD CS AND DE ALGORITHMS

This section brief recalls the basic principles of the standard CS and DE algorithms, respectively.

A. THE STANDARD CS

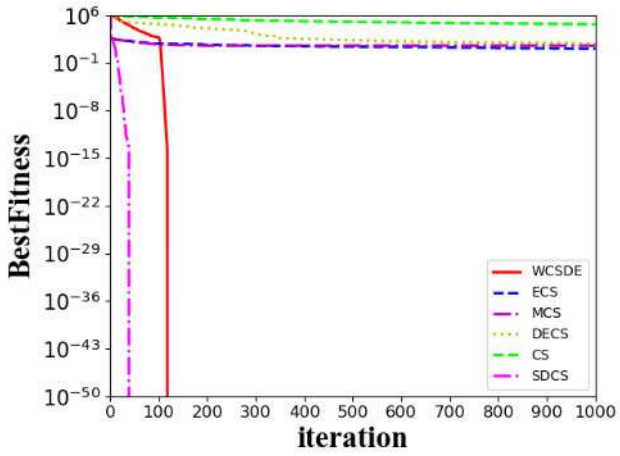
The CS algorithm is a kind of natural initiation algorithm that simulates the cuckoos' reproduction mode and combines with the Levy flight behavior to seek the potential optimal solution to industrial problems. In a CS algorithm, each individual represents a feasible solution in the search space, and the moving process of the individual position represents searching for the optimal solution. On the one hand, the Levy flight and random migration mechanisms are used to explore the entire solution space and quickly find the globally optimal solution. On the other hand, the CS can effectively maintain the diversity of the population in the search domain by controlling the balance between exploration and exploitation.

The standard CS, proposed by Yang and Deb in 2009, includes the following steps.

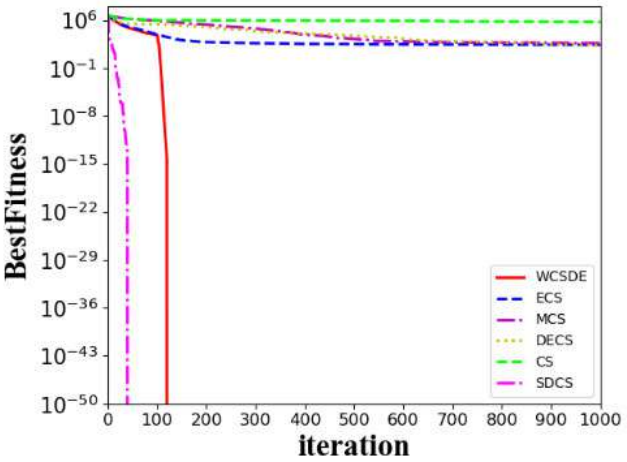
Step 1: Parameters initialization. The corresponding values of objective function $f(x)$, population size N, dimension D, search domain range $[LB, UB]$, discovery probability Pa, step factor α and maximum iteration times are defined in this step. Equation (1) is the population location initialization formula.

$$x_0 = LB + rand\,(D) \times (UB - LB) \tag{1}$$

where $rand(D)$ represents the 0-1 random number in the D dimension; x_0 is the initial nest position.

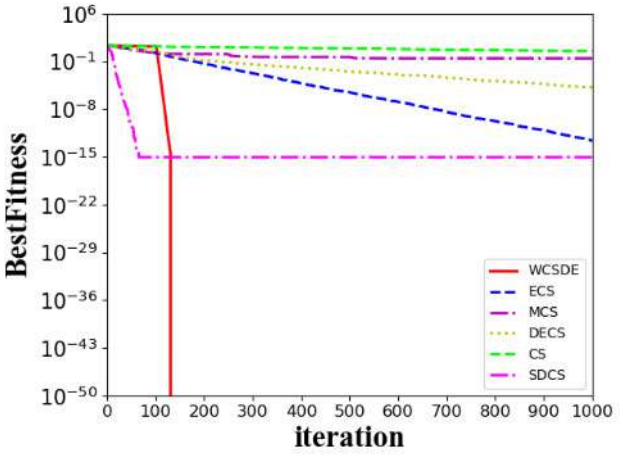


(a) 30-dimension situation

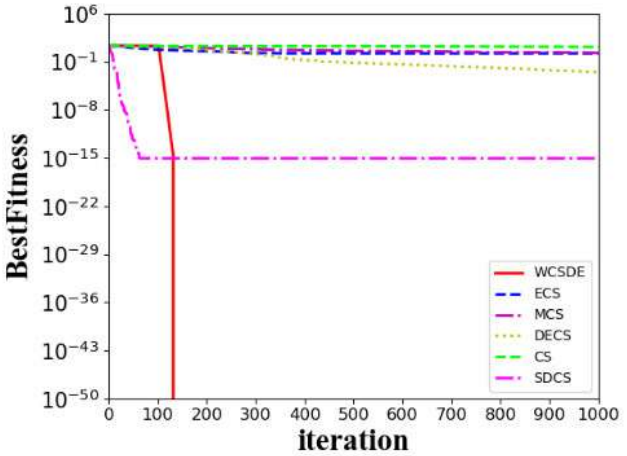


(b) 50-dimension situation

FIGURE 6. Performance comparison on F3.



(a) 30-dimension situation



(b) 50-dimension situation

FIGURE 7. Performance comparison on F4.

Step 2: Population updating. In the population renewal step, the birds' nests are migrated after initialization, the Levy flight mechanism is used to obtain the new birds' nests, and the fitness values are calculated and compared, and the best ones will be retained. The position of the i_{th} nest in the $t+1$ generation x_i^{t+1} can be calculated by (2).

$$x_i^{t+1} = x_i^t + \alpha \otimes Levy\,(\lambda)\,, \quad i \in [1, N] \tag{2}$$

where x_i^{t+1} and x_i^t represent the positions in the $t+1$, t iteration of the bird's Nest i, respectively; α is the step size scaling factor in the updating process ($\alpha > 0$); $\otimes$ represents point-to-point multiplication; $Levy(\lambda)$ represents the random searching path of a Levy flight, whose flight direction obeys a uniform distribution, and whose walk length conforms to the Levy distribution by (3).

$$Levy \sim u = t^{-\lambda}, \quad 1 \le \lambda \le 3 \tag{3}$$

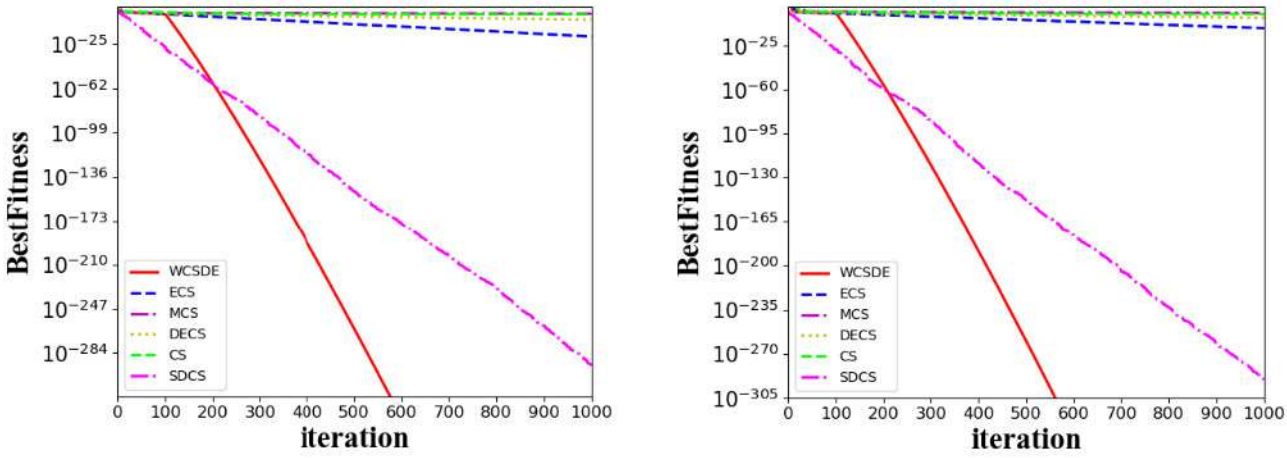
Since the probability density function of Levy distribution has no fixed form, the Mantegna algorithm [30] is widely used to simulate Levy flight, and the formula for generating step size is illustrated by (4) below.

$$s_L = \frac{\mu}{|\nu|^{\frac{1}{\beta}}}, \quad 1 \le \beta \le 2 \tag{4}$$

where μ and ν are normal distributed random numbers: $\mu \sim N(0, \delta_\mu^2)$, $\nu \sim N(0, \delta_\nu^2)$, $\delta_\nu = 1$. The calculation formula of parameter δ_μ is given by (5).

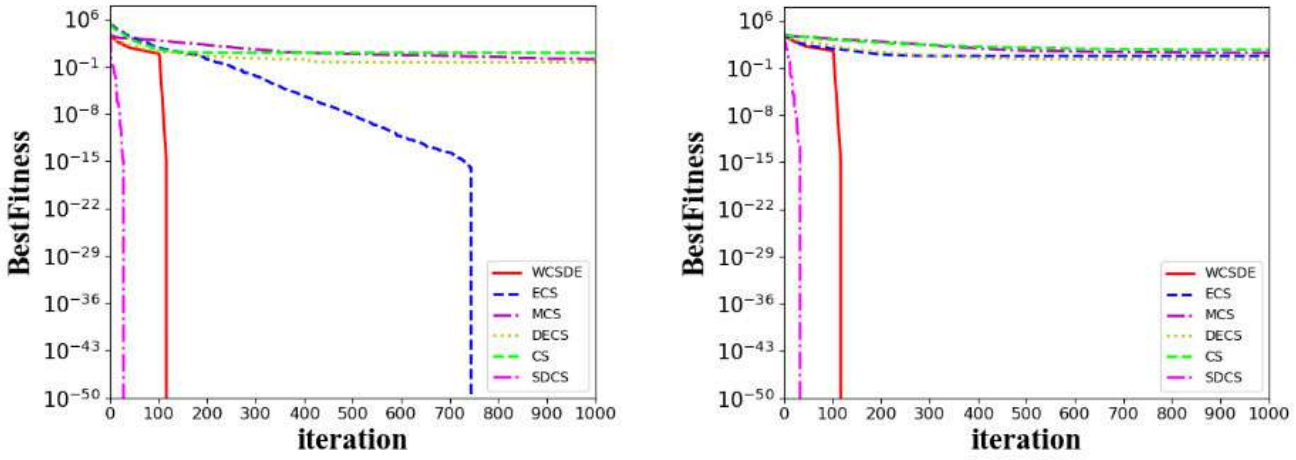
$$\delta_\mu = \left\{ \frac{\Gamma\,(1+\beta)\sin\,(\pi\beta/2)}{\Gamma\left[(1+\beta)/2\right] 2^{(\beta-1)/2}\,\beta} \right\}^{1/\beta} \tag{5}$$

Step 3: Random migration. A random number $r \in (0, 1)$ followed by a uniform distribution is generated and compared with the probability of being searched Pa of cuckoo. If $r > Pa$, the current nest is discarded, the position of the nest is updated



(a) 30-dimension situation (b) 50-dimension situation

FIGURE 8. Performance comparison on F5.



(a) 30-dimension situation (b) 50-dimension situation

FIGURE 9. Performance comparison on F6.

using (6), and the fitness value is calculated, and the position of the nest with the optimal fitness value is retained. Otherwise, the position of the nest remains unchanged.

$$x_{ij}^{t+1} = x_{\min,j}^{t} + rand\ (0,\ 1) \times \left(x_{\max,j}^{t} - x_{\min,j}^{t}\right) \qquad (6)$$

where $x_{\min,j}^{t}$ and $x_{\max,j}^{t}$ represent the lower and upper bound of an individual in the j_{th} dimension, respectively; *rand(0,1)* represents an uniformly distributed random number.

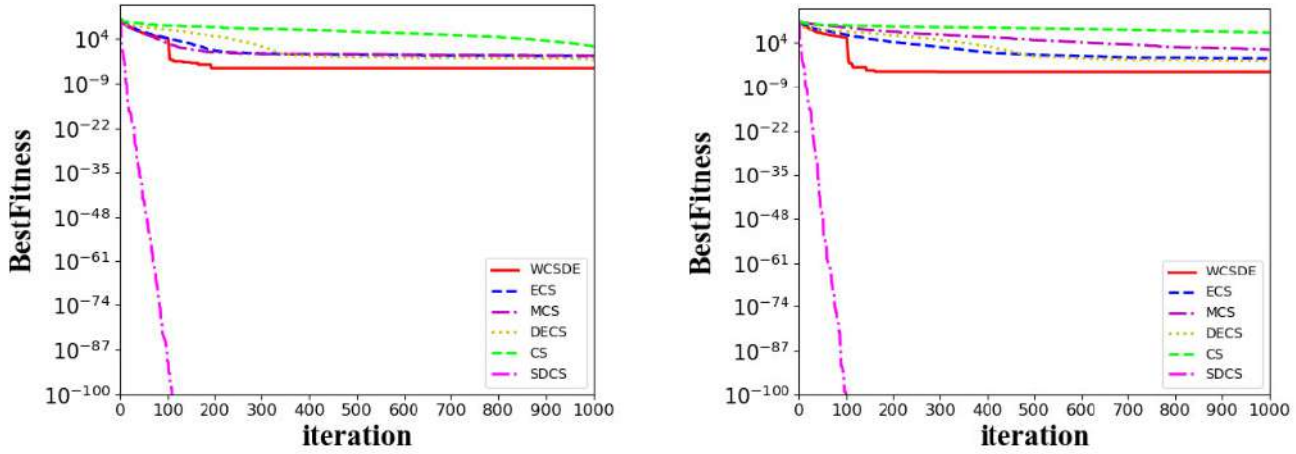
Step 4: Preferential Selection. Retain the solution with the better fitness value, and update the population and the globally optimal solution by comparing the fitness value of the candidate solution with the current solution.

Step 5: Terminate condition. To judge whether the current iteration number fulfills the maximum iteration number. If yes, output the optimal solution; Otherwise, move to Step 2.

The flowchart of the CS algorithm is illustrated as shown in Figure 1:

B. DIFFERENTIAL EVOLUTION ALGORITHM

DE algorithm is a population-based heuristic parallel global search algorithm proposed by Store and Price in 1997 [31]. Similar to the GA, the DE mainly includes initialization, mutation, crossover, selection, and other operations. The DE algorithm has been widely used because of its fast convergence, fewer control parameters, and strong robustness. Due to the random and changeability of mutation and crossover, the flexible update strategy of DE further effectively utilizes the characteristics of a population distribution to improve searchability. The perturbation of DE to the evolutionary individual is reflected by the different strategies between multiple individuals, which avoids the deficiency of mutation methods in GA. However, the DE still has some defects,



(a) 30-dimension situation

(b) 50-dimension situation

FIGURE 10. Performance comparison on F7.

like easily falling into the local optimum at the end of the iteration.

The flowchart of the DE algorithm is discussed as shown in Figure 2:

III. THE PROPOSED WCSDE ALGORITHM

In standard CS algorithm, the fixed search step size leads to the imbalance between the exploration and development operations, which lacks a significant movement in the early stage to find the globally optimal solution and a slight movement to refine the accuracy of the optimal solution in the later stage. Hence there is an information waste and slow convergence phenomena in the search process in the later period of the standard CS due to the execution of the evaluation independently and lack of adequate information-sharing mechanism within the population.

Because of the above problems, a hybrid WCSDE algorithm is discussed in this section to obtain higher convergence accuracy, increase the information interaction between individuals, and strengthen the local search ability by combining nonlinear inertial weights and DE in detail.

A. NONLINEAR INERTIA WEIGHT STRATEGY FOR CAPTURING THE RANDOM FLIGHT PATH

It is necessary to employ a large inertia weight in the early search stage to quickly find the globally optimal solution and avoid falling into the local optimal solution. Considering the need to accelerate the convergence speed and improve the local search ability later, it requires a smaller inertia weight in the later phase. Combined with this finding and the nonlinear inertia weight, an improved method of gaining a random flight path is proposed, as shown (7).

$$x_i^{t+1} = \omega * x_i^t + \alpha \otimes (x_j^t - x_i^t) \otimes Levy\,(\lambda)\,, \quad i \in [1, N] \tag{7}$$

where $j \in (0, N)$ and $j \neq i$, x_j^t is the individual position in the j_{th} population, w_0 is a positive real number, t the number of current iterations, t_0 is a given positive integer, and w is calculated by (8).

$$w = \begin{cases} w_0, & t \leq t_0 \\ (\frac{1}{t})^{0.3}, & t > t_0 \end{cases} \tag{8}$$

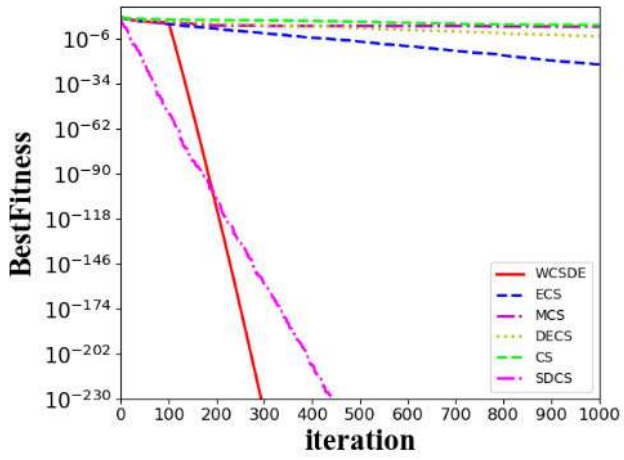
B. MUTATION AND CROSS-SELECTION STRATEGY BASED ON THE DE ALGORITHM

The basic CS algorithm immediately discards and chooses the best after updating the bird's nest's position with Levy flight. This method has not established an information exchange mechanism between populations and fully exploited the information. Therefore, the mutation of the DE is employed in the WCSDE to enhance the inner-information-sharing mechanism among populations. The nest position is mutated by difference calculation, and then the cross-matching is carried out according to the given probability. Finally, the fitness value is calculated and the better one will be selected for preservation.

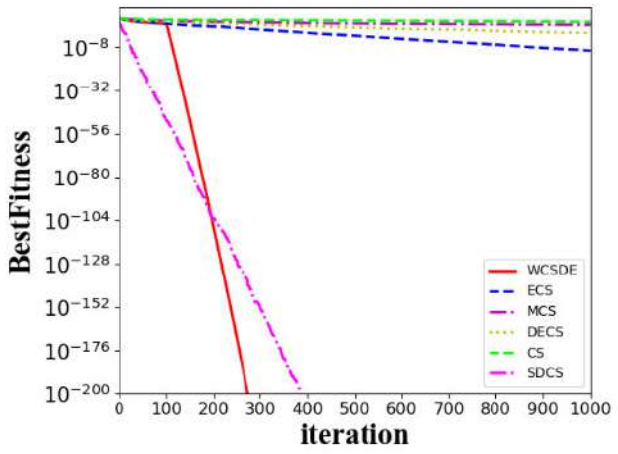
After updating the nest's location by (7), the difference vector of two individuals will be randomly selected from the population is used as the Source of random variation of the third individual. Then, the weighted difference vector combined with the individual will be mutated using (9).

$$V_i^{t+1} = x_{r_1}^t + F(x_{r_2}^t - x_{r_3}^t) \tag{9}$$

where $i \neq r_1 \neq r_2 \neq r_3$, F is the scaling factor, x_i^t represents the i_{th} individual in the population of the t_{th} generation. Each "gene" in the "chromosome" must be judged whether it is within the set search range in the evolution process. If it is beyond the range, the "gene" will be regenerated by the initialization operation.

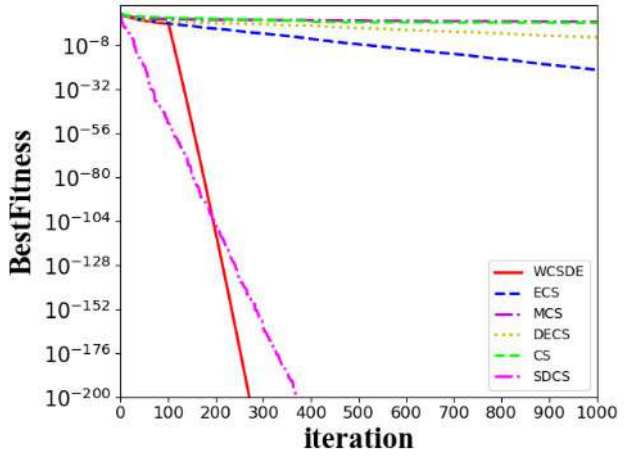


(a) 30-dimension situation

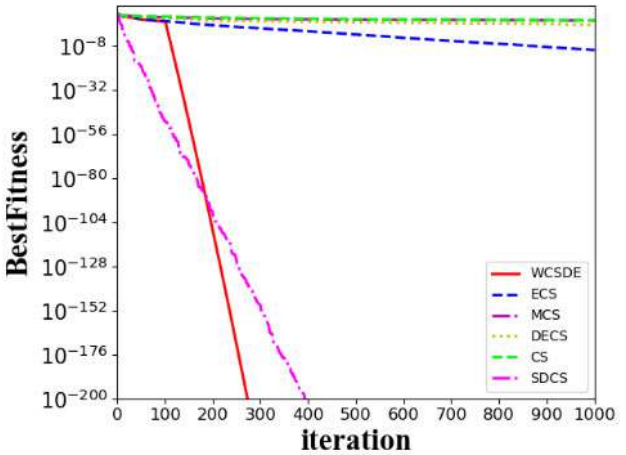


(b) 50-dimension situation

FIGURE 11. Performance comparison on F8.



(a) 30-dimension situation



(b) 50-dimension situation

FIGURE 12. Performance comparison on F9.

After the mutation intermediate $\left\{V_i^{t+1}\right\}$ is obtained through (9), the cross operation is conducted between t_{th} and the $t+1_{th}$ generation population $\left\{x_i^t\right\}$.

$$u_{j,i}^{t+1} = \begin{cases} V_{j,i}^{t+1}, & if\ rand(0,1) \le CR\ or\ j = Jrand \\ x_{j,i}^{t}, & otherwise \end{cases} \tag{10}$$

where CR is the crossover probability, j represents a dimension, and $Jrand$ is a random integer in $[1, 2, \ldots, D]$. It is necessary to ensure that at least one "gene" in each "chromosome" of the mutation intermediate is inherited to the next generation, and cannot be replaced by the "gene" in x_i^t for exchanging information between individuals effectively.

Finally, the greedy algorithm is adopted to select better individuals to enter the next generation using (11).

$$x_i^{t+1} = \begin{cases} u_i^{t+1}, & if\ f(u_i^{t+1}) \le f(x_i^t) \\ x_i^t, & otherwise \end{cases} \tag{11}$$

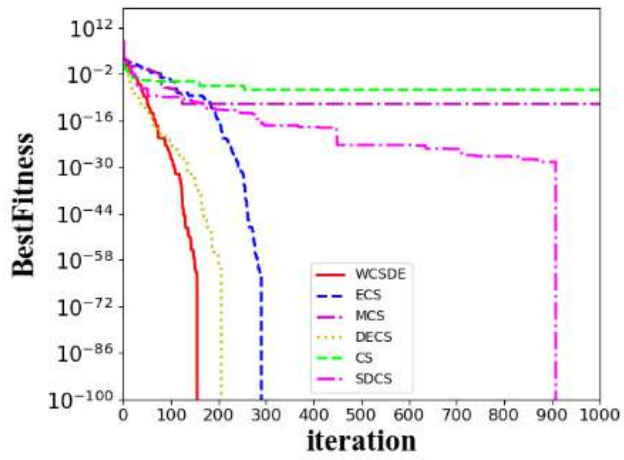
C. HIGHLIGHTS OF THE PROPOSED ALGORITHM

According to formula (8), the inertia weight changes with the number of iterations. The greater the number of iterations, the smaller the inertia weight to be used later in the search. Move the range to find the best, and improve the ability of local search. Secondly, combined with the advantages of the DE algorithm, according to formula (9), the obtained solution is mutated, the diversity is increased, and the information exchange between the populations is established. The optimal solution is selected through formula (10) and formula (11), enhancing WCSDE Searchability.

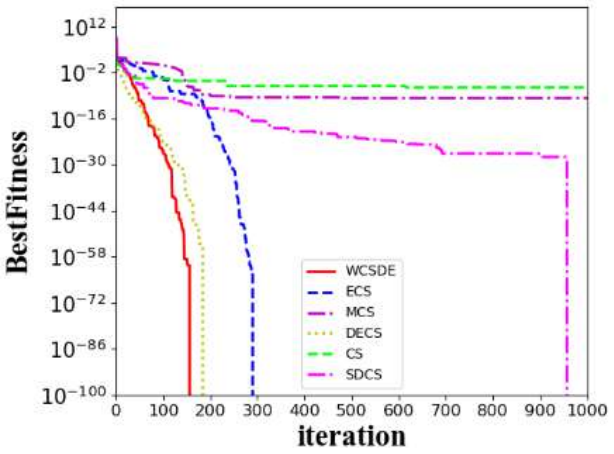
D. FLOWCHART OF THE WCSDE ALGORITHM

According to the above two improvements, the presented WCSDE algorithm includes the following steps:

Step 1: Problem modeling and parameters initialization. Define the objective function $f(x)$, population size N,

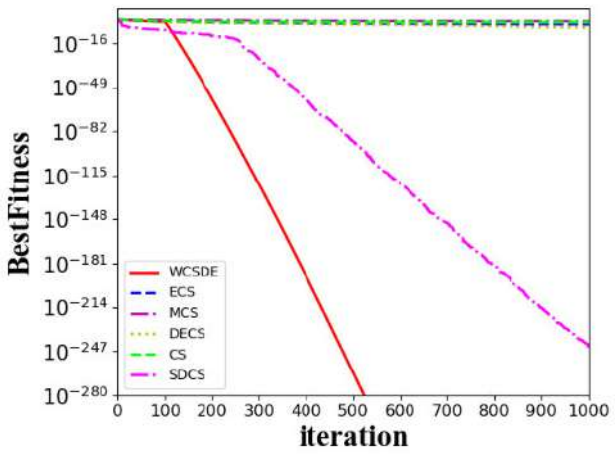


(a) 30-dimension situation

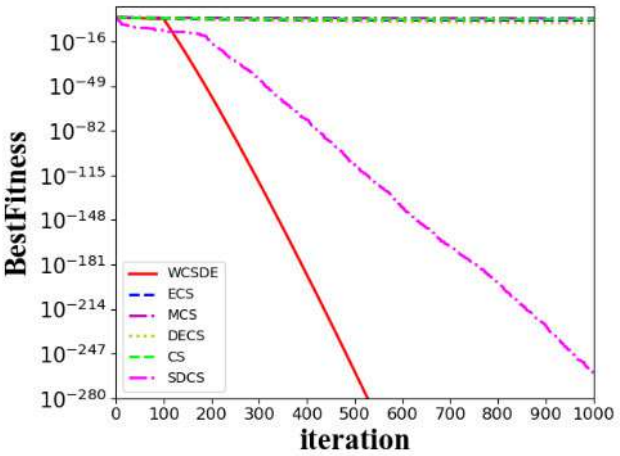


(b) 50-dimension situation

FIGURE 13. Performance comparison on F10.



(a) 30-dimension situation



(b) 50-dimension situation

FIGURE 14. Performance comparison on F11.

problem dimension D, discovery probability Pa, step size α, scaling factor F, crossover probability CR, inertial weight w, maximum iteration times T, and search domain range [LB, UB].

Step 2: Population initialization. Initialize the population and the initial position using (1).

Step 3: Population updating. The value of inertia weight w is obtained by (8), and the population position is updated by (7).

Step 4: Population mutation operation. The mutation intermediate $\left\{V_i^{t+1}\right\}$ is obtained by the individual variation through (9).

Step 5: Cross operation. Implement the crossover operation between individuals by (10), and judge whether the mutation intermediate needs to inherit to the next generation due to the crossover probability CR.

Step 6: Select operation. The greedy algorithm is used to select the best individual, and the fitness of the parent generation individual and the offspring individual is compared through (11). The solution with better fitness value will remain, and the population and globally optimal solution are updated.

Step 7: Random migration. After executing the mutation and cross-selection operations, the new generation of individuals is discarded to generate a random number $r \in (0, 1)$ with uniform distribution. Then, the obtained random number will be compared with the probability of cuckoo being found Pa. If $r > Pa$, the current nest is discarded, the position of the nest will be updated by (6), and the fitness value will be re-calculated. The nest position of the optimal fitness value will remain after comparing the fitness value of the candidate solution and the current solution. Otherwise, it remains the exited solution.

TABLE 3. Experimental results with a fixed number of iterations.

	Dim	ECS	CS	MCS	DECS	SDCS	WCSDE
		MEAN ± *SD*	*MEAN* ± *SD*	*MEAN* ± *SD*	*MEAN* ± *SD*	*MEAN* ± *SD*	*MEAN* ± *SD*
F1	30	3.72E-24±3.76E-24	7.17E-01±7.96E-02	1.54E-02±1.45E-03	7.26E-11±5.71E-11	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**
	50	1.19E-15±1.01E-15	3.85E+00±0.43E+00	8.14E-02±6.77E-03	1.57E-06±1.69E-06	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**
F2	30	2.94E-03±7.72E-03	1.24E+01±3.53E+00	1.70E-02±1.18E-02	2.44E-02±2.42E-02	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**
	50	2.32E-03±5.94E-03	1.44E+02±1.57E+01	0.28E+00±0.07E+00	1.61E-01±2.47E-01	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**
F3	30	1.28E+01±2.59E+00	4.23E+04±1.25E+04	4.60E+01±1.02E+01	6.19E+01±1.72E+01	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**
	50	2.49E+02±5.57E+01	5.50E+05±7.69E+04	5.25E+02±5.78E+01	3.24E+02±6.40E+02	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**
F4	30	6.80E-13±6.11E-13	3.68E+00±2.30E+00	2.24E-01±5.22E-02	4.08E-05±3.61E-05	8.88E-16±0.00E+00	**0.00E+00±0.00E+00**
	50	1.23E+00±0.53E+00	1.43E+01±8.45E-01	1.91E+00±0.16E+00	5.78E-03±3.06E-03	8.88E-16±0.00E+00	**0.00E+00±0.00E+00**
F5	30	1.50E-19±1.17E-19	1.81E+00±5.03E-01	5.54E-01±3.35E-02	2.38E-05±8.55E-06	7.1E-281±0.00E+00	**0.00E+00±0.00E+00**
	50	9.78E-12±4.57E-12	1.11E+01±0.77E+00	1.71E+00±7.86E-02	3.72E-03±2.35E-03	4.3E-288±0.00E+00	**0.00E+00±0.00E+00**
F6	30	2.07E+00±1.56E+00	1.10E+01±2.77E+00	1.14E+00±2.39E-01	1.45E-01±2.06E-02	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**
	50	7.60E+00±2.19E+00	4.31E+01±2.83E+00	1.06E+01±1.60E+00	3.53E+00±1.77E+00	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**
F7	30	8.80E-02±2.61E-02	2.45E+02±4.39E+02	2.05E-01±6.33E-02	1.46E-02±3.13E-03	**0.00E+00±0.00E+00**	1.94E-05±1.50E-05
	50	0.41E+00±0.13E+00	1.61E+07±8.75E+06	6.16E+01±2.16E+01	6.11E-02±1.62E-02	**0.00E+00±0.00E+00**	2.08E-05±1.40E-05
F8	30	4.16E-22±2.84E-22	1.23E+03±2.00E+03	8.40E+01±6.81E+01	1.54E-04±5.03E-04	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**
	50	2.03E-10±1.34E-10	8.21E+05±8.29E+05	2.27E+04±1.07E+04	2.82E+01±6.00E+01	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**
F9	30	3.93E-22±3.95E-22	1.06E+05±8.93E+01	1.36E+04±7.37E+03	1.96E-03±3.56E-03	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**
	50	3.47E-10±3.06E-10	1.05E+06±2.79E+05	9.43E+05±3.66E+05	3.40E+03±3.22E+03	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**
F10	30	**0.00E+00±0.00E+00**	2.32E-07±2.64E-07	4.05E-10±4.70E-10	**0.00E+00±0.00E+00**	9.16E-01±4.93E+00	**0.00E+00±0.00E+00**
	50	**0.00E+00±0.00E+00**	1.50E-07±1.51E-07	2.76E-10±3.98E-10	**0.00E+00±0.00E+00**	4.77E+00±1.43E+01	**0.00E+00±0.00E+00**
F11	30	1.71E-02±1.51E-02	3.50E+00±0.74E+00	1.19E+00±4.02E-01	3.12E-03±2.61E-03	1.95E-18±1.04E-17	**0.00E+00±0.00E+00**
	50	0.46E+00±0.23E+00	1.73E+01±3.29E+00	4.97E+00±1.03E+00	8.65E-03±4.37E-03	9.79E-17±2.62E-16	**0.00E+00±0.00E+00**
F12	30	9.23E-03±4.01E-03	1.80E+01±6.20E+00	3.57E-01±1.02E-01	5.20E-03±2.35E-03	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**
	50	8.23E-03±3.43E-03	1.81E+01±7.96E+00	4.10E-01±1.14E-01	5.27E-03±3.37E-03	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**
F13	30	1.97E-24±2.91E-24	7.48E-01±1.11E-01	9.75E-02±1.14E-01	2.51E-09±2.01E-09	**0.00E+00±0.00E+00**	2.29E-25±5.19E-25
	50	4.52E-12±3.13E-12	2.32E+03±6.94E+02	4.74E+00±0.62E+00	4.75E-05±4.92E-05	**0.00E+00±0.00E+00**	4.64E-13±5.91E-13

Step 8: Terminate condition. Determine whether the current iteration number reaches the maximum iteration number. If so, the optimal solution will be output, otherwise, go to Step3.

According to the above steps, the flowchart of the WCSDE algorithm is revealed as shown in Figure 3.

E. TIME COMPLEXITY ANALYSIS

Time complexity is an essential theoretical criterion to measure the performance of the proposed algorithm by calculating time. The time complexity of the proposed WCSDE is given based on the mentioned steps above.

Firstly, the time complexity of the WCSDE under each iteration could be calculated from the following phases.

Assume the dimension of the objective function f as Dim and the population size as N, respectively. The time complexity of the initial stage and the time complexity of evaluating the objective function of each individual in the population are O($N \times Dim$) and O($f(Dim)$), respectively.

According to (7), the time complexity of the random flight path update stage is O($N \times (Dim + O(Levy))$, where O($Levy$) is a random number following the Levy distribution. Hence, the order of its computational complexity is a constant order denoised as O(1). Furthermore, the time complexity is O($N \times Dim$) while updating the population.

In (9)-(11), the time complexity of optimization using mutation and cross-selection operation is O($N \times Dim$); In the random walk stage, another new nest is built using (6), and the corresponding time complexity is O($N \times Dim$).

TABLE 4. The wilcoxon test results of the experimental results with a fixed number of iterations.

	Dim	ECS		CS		MCS		DECS		SDCS		WCSDE
		Wilcox	Rank	Wilcox	Rank	Wilcox	Rank	Wilcox	Rank	Wilcox	Rank	Rank
F1	30	+	3	+	6	+	5	+	4	=	1	1
	50	+	3	+	6	+	5	+	4	=	1	1
F2	30	+	3	+	6	+	4	+	5	=	1	1
	50	+	3	+	6	+	5	+	4	=	1	1
F3	30	+	3	+	6	+	4	+	5	=	1	1
	50	+	3	+	6	+	5	+	4	=	1	1
F4	30	+	3	+	6	+	5	+	4	+	2	1
	50	+	3	+	6	+	4	+	5	+	2	1
F5	30	+	3	+	6	+	5	+	4	+	2	1
	50	+	3	+	6	+	5	+	4	+	2	1
F6	30	+	5	+	6	+	4	+	3	=	1	1
	50	+	4	+	6	+	5	+	3	=	1	1
F7	30	+	4	+	6	+	5	+	3	-	1	2
	50	+	4	+	6	+	5	+	3	-	1	2
F8	30	+	3	+	6	+	5	+	4	=	1	1
	50	+	3	+	6	+	5	+	4	=	1	1
F9	30	+	3	+	6	+	5	+	4	=	1	1
	50	+	3	+	6	+	5	+	4	=	1	1
F10	30	=	1	+	5	+	4	=	1	+	6	1
	50	=	1	+	5	+	4	=	1	+	6	1
F11	30	+	4	+	6	+	5	+	3	+	2	1
	50	+	4	+	6	+	5	+	3	+	2	1
F12	30	+	4	+	6	+	5	+	3	=	1	1
	50	+	4	+	6	+	5	+	3	=	1	1
F13	30	+	3	+	6	+	5	+	4	-	1	2
	50	+	3	+	6	+	5	+	4	-	1	2
Ave			3.19		5.92		4.77		3.58		1.61	1.15
Final			3		6		5		4		2	1

(**PS**: **Ave** is average rank, and **Final** is finally rank.)

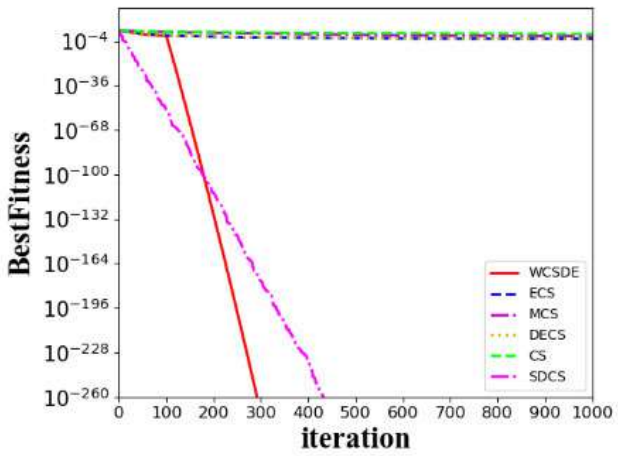
To sum up, the worst time complexity of the WCSDE under each iteration is approximately as

$$\begin{aligned} &\mathrm{O}(N \times Dim) + \mathrm{O}(N \times (Dim + f(Dim)) + \mathrm{O}(N \times (Dim \\ &\quad + f(Dim)) + \mathrm{O}(N \times (Dim + f(Dim)) \\ &\quad \approx \mathrm{O}(N \times (Dim + f(Dim)). \end{aligned}$$

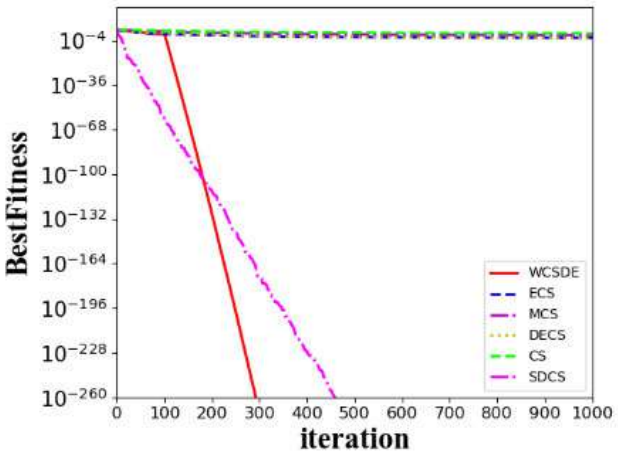
Therefore, the entire time complexity of the WCSDE is $\mathrm{O}(T_{max} \times N \times (Dim + f(Dim))$ while reaching the maximum number of iterations T_{max}.

IV. FUNCTION OPTIMIZATION USING THE WCSDE

In this part, 13 benchmark functions are selected to verify the feasibility and correctness of function optimization ability of the proposed WCSDE. The entire experiment includes two modules. In the first part, the number of iterations is used as a criterion to illustrate the feasibility of the proposed algorithm. In the second part, the number of function evaluations (FEs) is employed as an indicator to highlight the characteristics of the WCSDE. The computer used in the experiment is Intel ® Core ™ i5 - 9300H CPU @ 2.40 GHz,

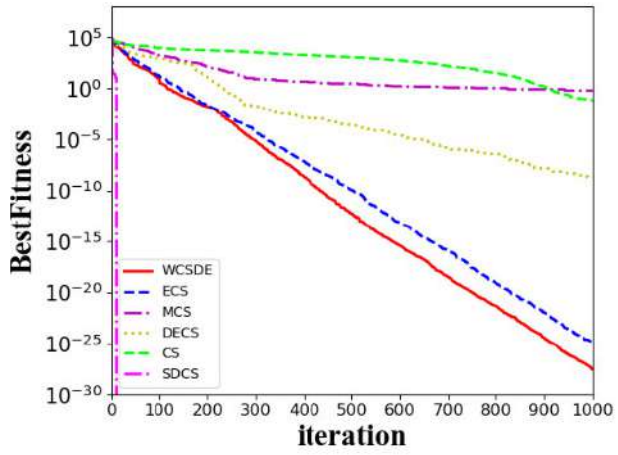


(a) 30-dimension situation

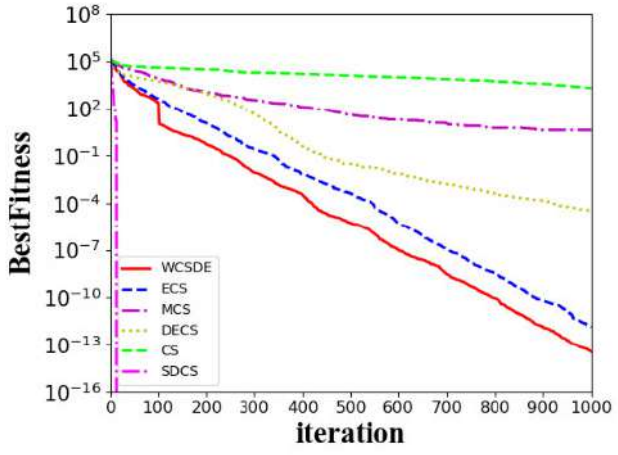


(b) 50-dimension situation

FIGURE 15. Performance comparison on F12.



(a) 30-dimension situation



(b) 50-dimension situation

FIGURE 16. Performance comparison on F13.

16GB memory, the Windows 10 operating system and other environments. The programming language Python 3.7 is used to realize the functional code.

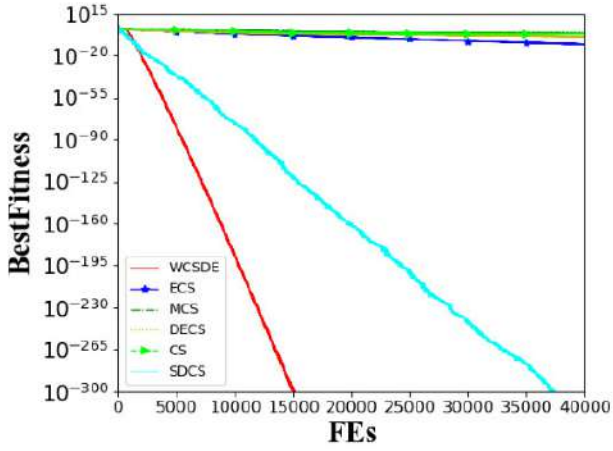
Details of the selected ten test functions are listed in Table 1.

The optimizations among the 13 benchmark functions are implemented utilizing the standard CS, the WCSDE and other four CS variants (enhance Cuckoo Search (ECS) [32], modified Cuckoo Search (MCS) [33], a hybrid optimization algorithm based on Differential Evolution and Cuckoo Search (DECS) [25], and Snap-drift cuckoo search (SDCS) [34]. In the first experiment, 1000 iterations were set as the criterion, and each algorithm was run independently 30 times. In the second experiment, the number of function evaluations was set as 40,000 times, and each algorithm was run independently 30 times. By referring to the relevant literature of the selected algorithm, the related parameter settings are shown in Table 2.

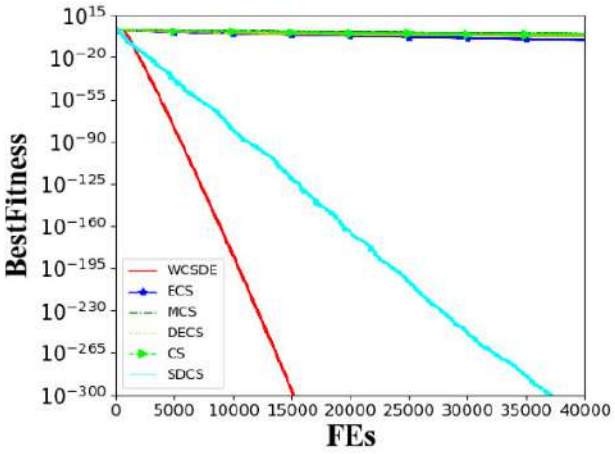
Figures 4-29 are a set of random running results in 30 experiments. Functions F1-F13 are the benchmark functions, and the test dimensions are 30 and 50, respectively. Moreover, figures 4-29 (a) record the convergence curve of the six CS algorithms of 30 dimensions. Figures 4-29 (b) are the convergence curve of the six CS algorithms of 50 dimensions. Figures 4-16 are the first part of the experimental convergence curve.

It demonstrates that WCSDE and SDCS have significant advantages over other algorithms in terms of convergence performance.

The WCSDE can quickly converge to the globally optimal value within 150 iterations when optimizing F2, F3, F4, F6, and F10. Similarly, the SDCS provides faster

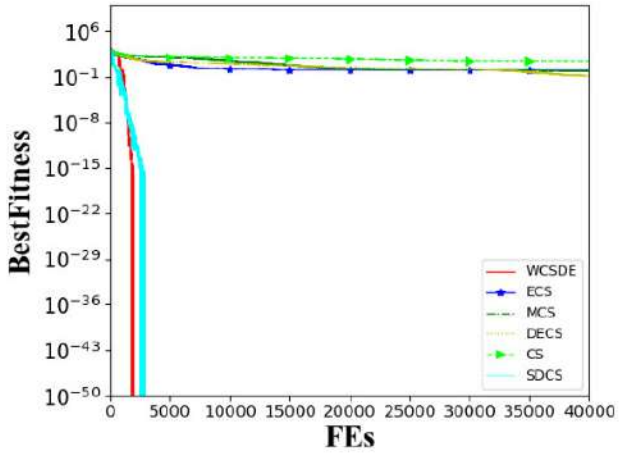


(a) 30-dimension situation

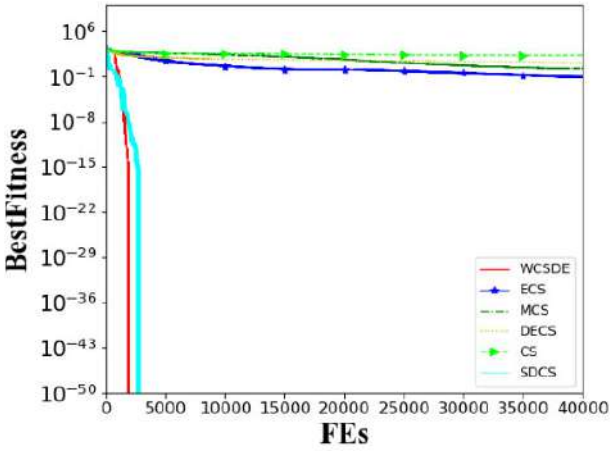


(b) 50-dimension situation

FIGURE 17. Performance comparison on F1.

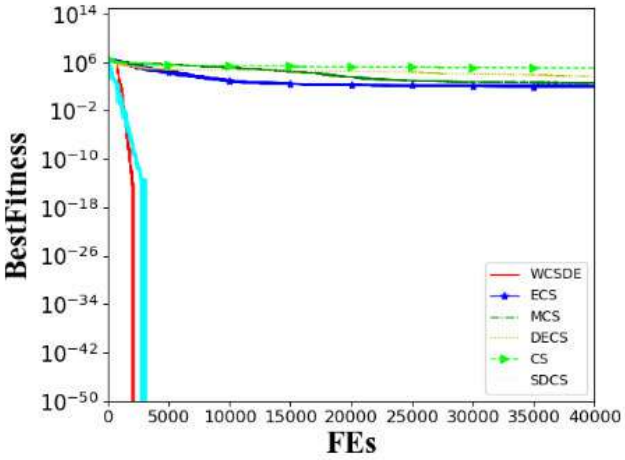


(a) 30-dimension situation

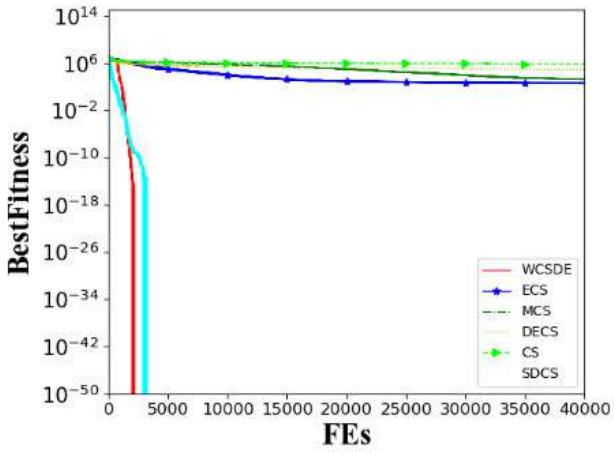


(b) 50-dimension situation

FIGURE 18. Performance comparison on F2.

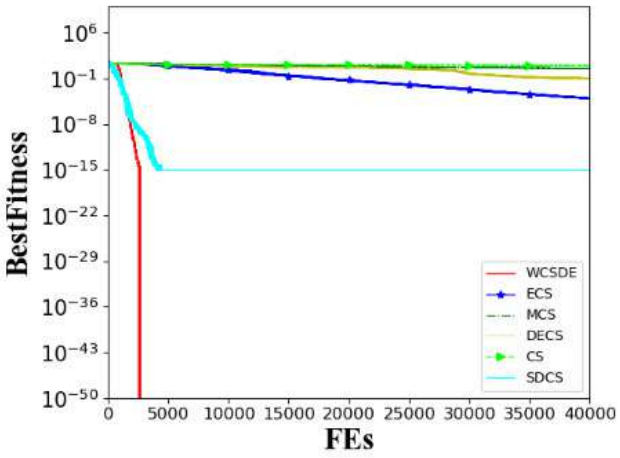


(a) 30-dimension situation

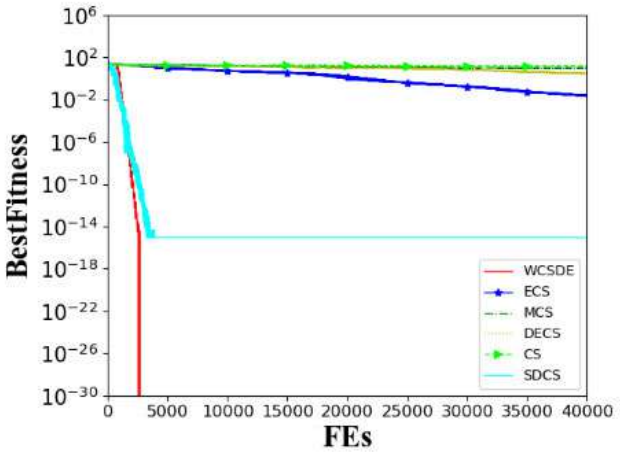


(b) 50-dimension situation

FIGURE 19. Performance comparison on F3.

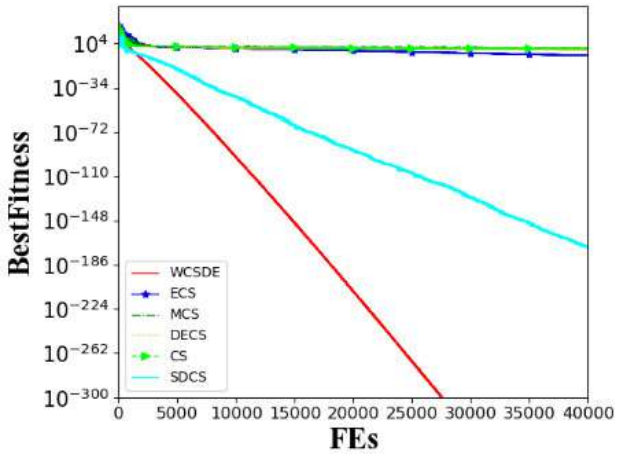


(a) 30-dimension situation

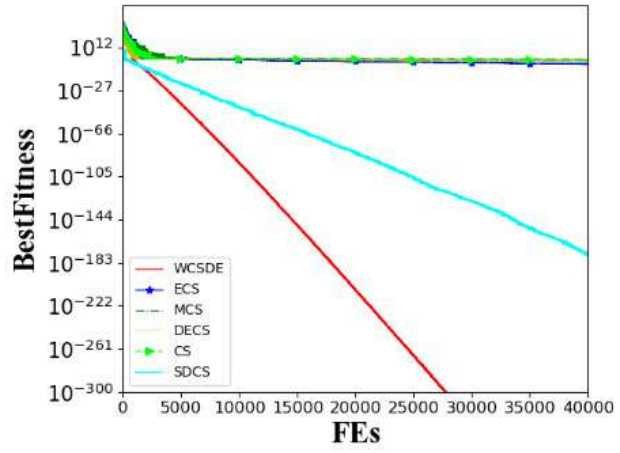


(b) 50-dimension situation

FIGURE 20. Performance comparison on F4.



(a) 30-dimension situation



(b) 50-dimension situation

FIGURE 21. Performance comparison on F5.

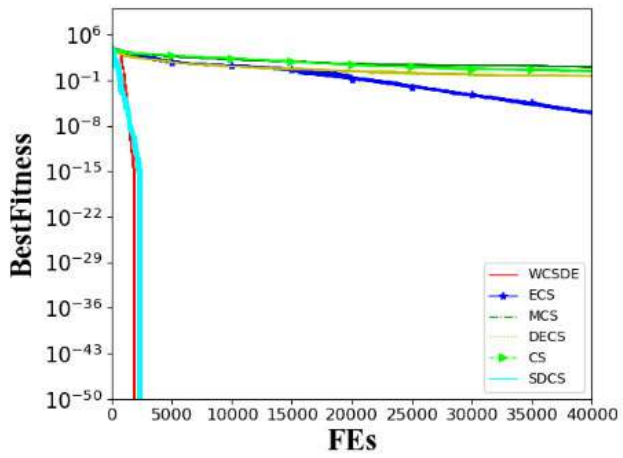
convergence speed and accurate globally optimal values for these problems, especially for functions F2, F3, F6, F7, and F13. It indicates that WCSDE achieved better overall performance than the other five algorithms in function F4, converging to the theoretical optimal value.

In the case of function F10, although ECS, DECS, and SDCS can also converge to the global optimum, the convergence efficiency is not as good as the WCSDE algorithm.

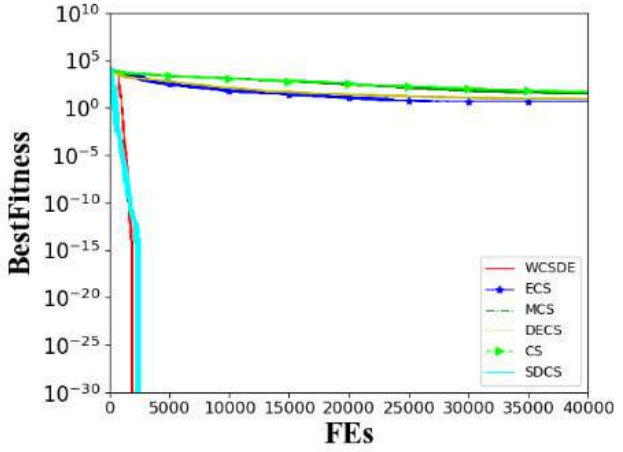
Secondly, the WCSDE can converge to the globally optimal solution within the maximum number of iterations while optimizing the functions F1, F5, F8, F9, F11, and F12 while ECS, MCS, and DECS algorithms are insufficient in the downward trend, especially MCS and DECS stopped in-depth search in the early stage. It further reveals that the WCSDE has more vital convergence ability, better adaptive ability, and more exact convergence accuracy than other variants. Thirdly, when optimizing the 50-dimensional F2, F3, and F6 problems, the performance of ECS is severely restricted, which is significantly lower than the 30-dimensional search capability.

In addition, in the functions F1, F4, F5, F8, F9, F11, and F12, it shows that SDCS has efficient convergence efficiency within 200 iterations, but the WCSDE exhibits more substantial local search capabilities in subsequent iterations. It also means that the non-linear inertia weight moves smaller in the later search process, which quickly improves the search accuracy of the algorithm.

To sum up, the WCSDE shows outstanding convergence performance while optimizing the above 13 functions under two different dimensions. Significantly, the WCSDE achieves ideal optimization results with similar 30-dimensional and 50-dimensional conditions and can handle high-dimensional data. Hence, the WCSDE has a robust adaptive ability to

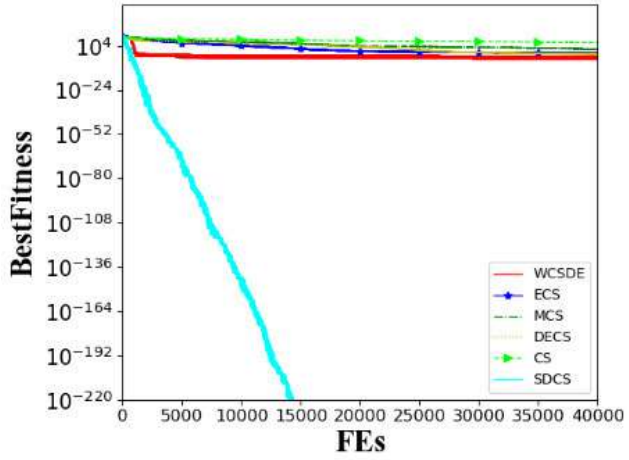


(a) 30-dimension situation

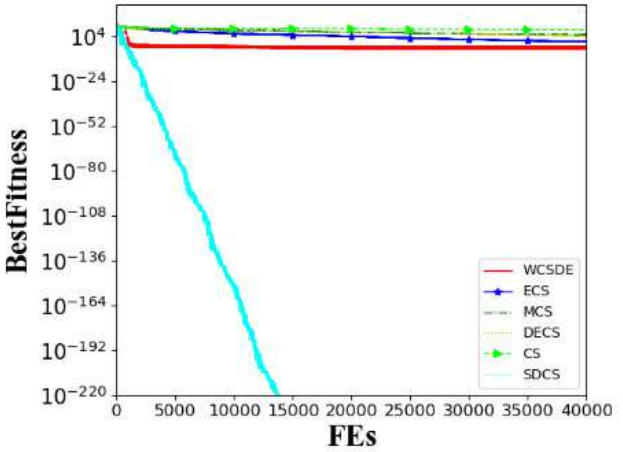


(b) 50-dimension situation

FIGURE 22. Performance comparison on F6.

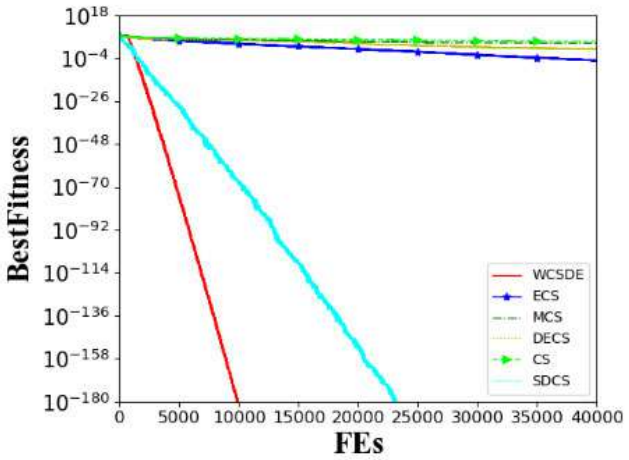


(a) 30-dimension situation

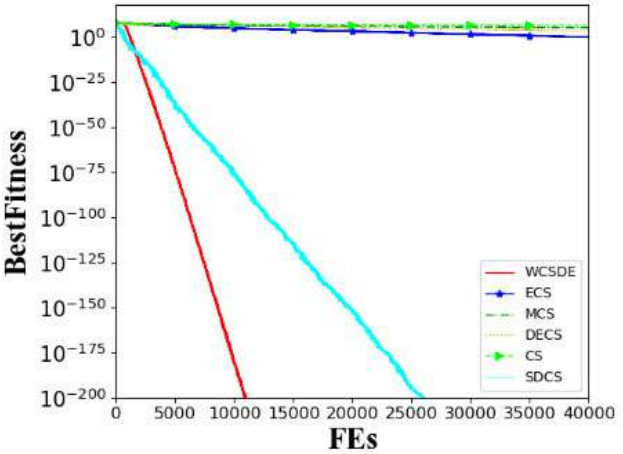


(b) 50-dimension situation

FIGURE 23. Performance comparison on F7.

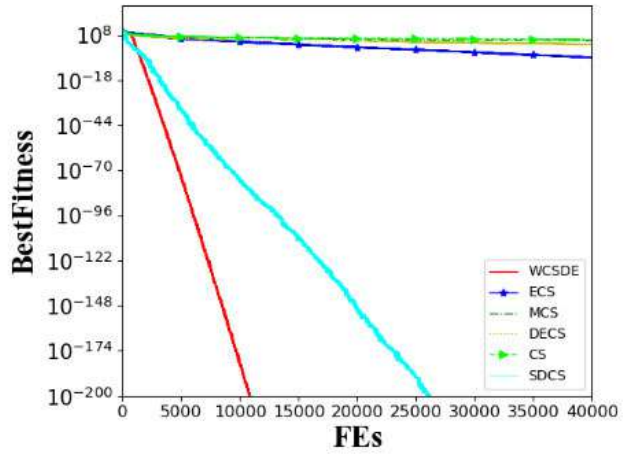


(a) 30-dimension situation

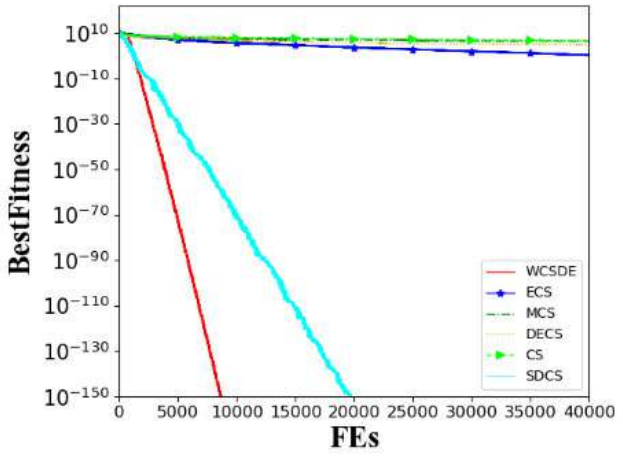


(b) 50-dimension situation

FIGURE 24. Performance comparison on F8.

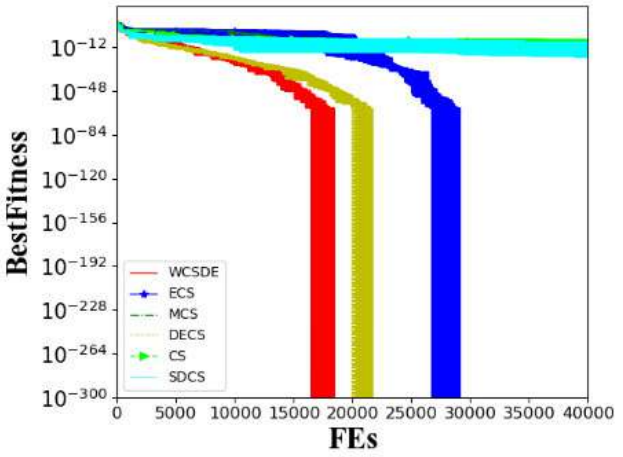


(a) 30-dimension situation

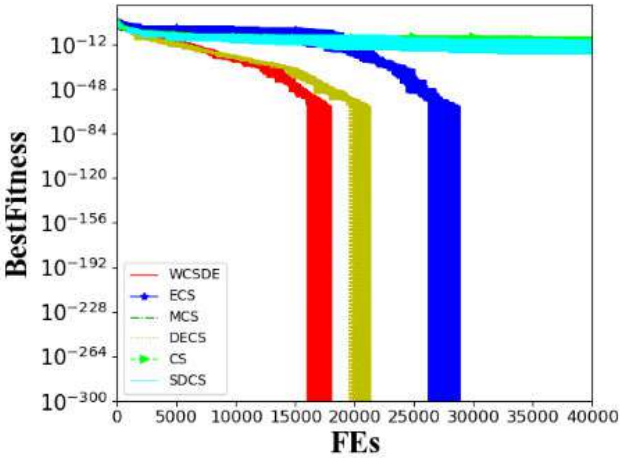


(b) 50-dimension situation

FIGURE 25. Performance comparison on F9.



(a) 30-dimension situation



(b) 50-dimension situation

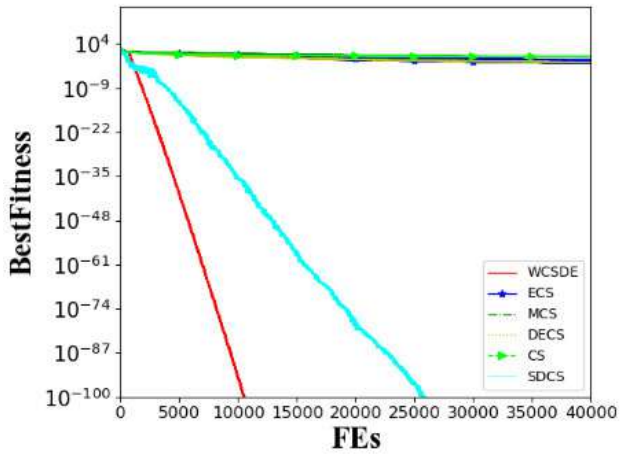
FIGURE 26. Performance comparison on F10.

optimize unimodal and multimodal functions and obtain higher convergence accuracy.

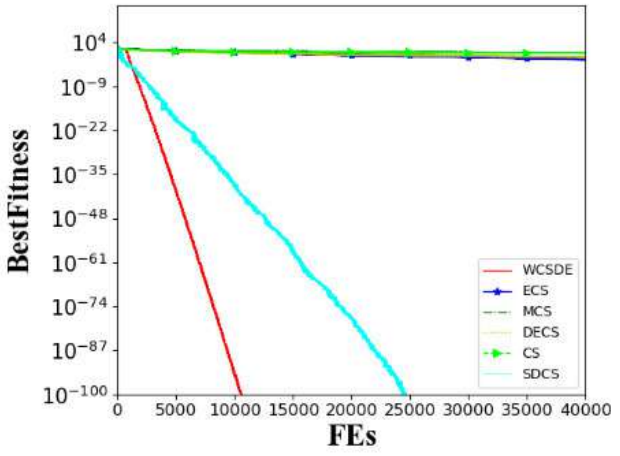
The average value and standard deviation are used to judge the superiority of the standard CS and improved CS algorithms' performance in Tables 3 and 4. In these tables, the best value is shown in bold black font, and the robustness of these algorithms is determined based on the average value and standard deviation. If the global optimal solution's value is calculated more accurately, the corresponding global convergence and the algorithm's stability are better and more robust, respectively.

Table 3 shows that the WCSDE algorithm can find the theoretically optimal solution except in the optimization of the F7 and F13 functions. Especially in optimizing F4, F5 and F11, we have achieved better mean and standard deviation results than the other five CS algorithms. However, when optimizing the F7 and F13 functions, the SDCS algorithm has a more robust search capability. It can be seen from the above table that WCSDE shows an excellent ability to deal with high-dimensional problems, gives full play to the global optimization ability, quickly jumps out of the local optimal state, and obtains high convergence accuracy. In addition, in optimizing the F7 and F13 functions, the mean and standard deviation obtained by WCSDE are also smaller than the other three variants. By analyzing the mean and standard deviation data obtained by statistical analysis, it is concluded that WCSDE has better optimization performance than the different five algorithms as a whole.

A non-parametric Wilcoxon signed-rank test is performed to determine whether there is a significant difference among the results obtained by the algorithms. Statistical tests are performed on the average results obtained by 30 runs of each algorithm to reveal the performance

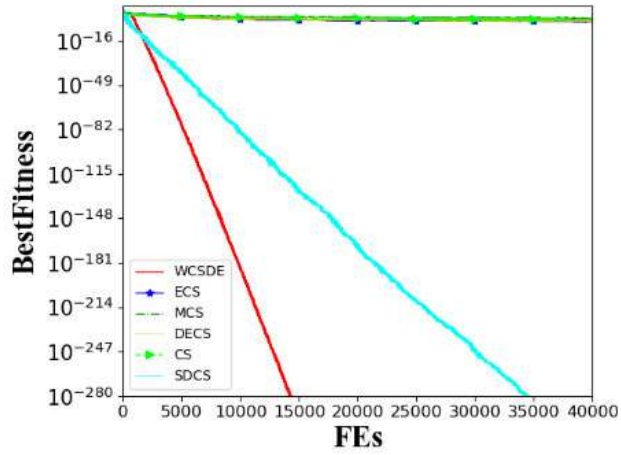


(a) 30-dimension situation

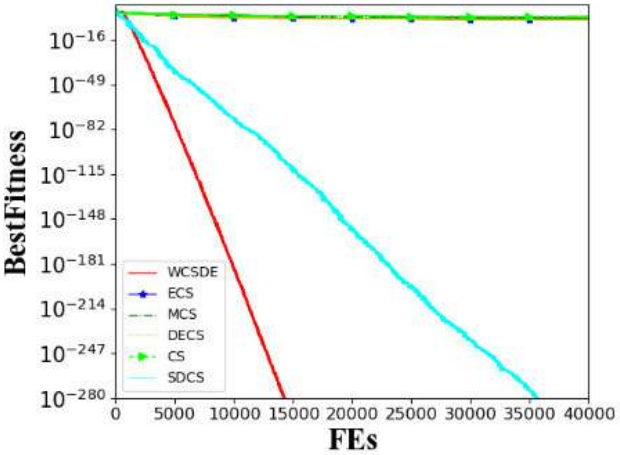


(b) 50-dimension situation

FIGURE 27. Performance comparison on F11.

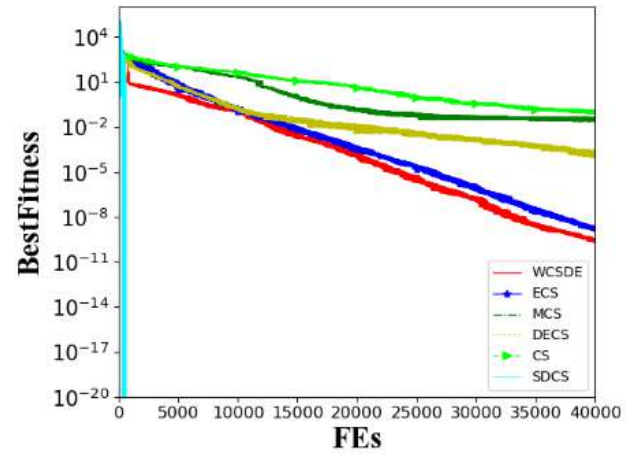


(a) 30-dimension situation

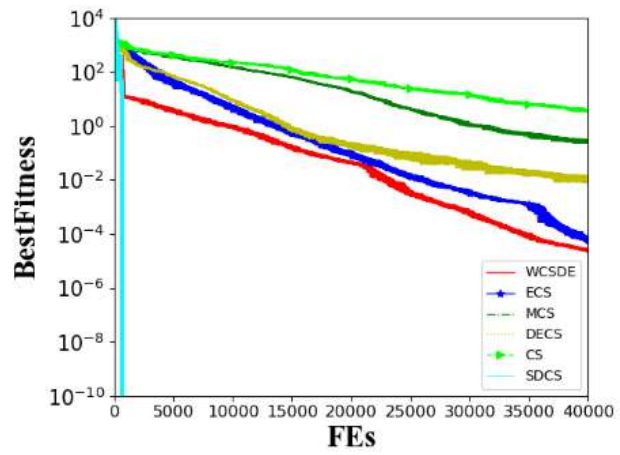


(b) 50-dimension situation

FIGURE 28. Performance comparison on F12.



(a) 30-dimension situation



(b) 50-dimension situation

FIGURE 29. Performance comparison on F13.

TABLE 5. Experimental results of fixed function evaluation times.

	Dim	ECS	CS	MCS	DECS	SDCS	WCSDE
		MEAN ± *SD*	*MEAN* ± *SD*	*MEAN* ± *SD*	*MEAN* ± *SD*	*MEAN* ± *SD*	*MEAN* ± *SD*
F1	30	2.52E-11±3.94E-12	4.49E-02±4.02E-03	3.36E-02±2.31E-03	1.02E-04±1.82E-05	3.8E-288±0.00E+00	**0.00E+00±0.00E+00**
	50	3.67E-06±3.76E-06	2.77E-01±2.23E-02	1.69E-01±1.11E-02	3.62E-03±3.40E-04	1.1E-284±0.00E+00	**0.00E+00±0.00E+00**
F2	30	1.09E-01±3.09E-01	2.53E+01±7.66E+00	6.29E-01±5.79E-02	2.44E-02±2.42E-02	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**
	50	9.03E-01±1.94E-01	2.04E+02±2.85E+01	1.33E+00±4.59E-02	1.17E+01±3.72E+00	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**
F3	30	1.16E+02±1.81E+01	1.06E+05±1.60E+04	3.05E+02±9.44E+01	4.96E+03±3.46E+02	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**
	50	3.83E+02±2.54E+01	6.86E+05±8.44E+04	1.93E+03±5.61E+02	4.90E+04±2.19E+04	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**
F4	30	8.09E-05±8.27E-06	8.91E+00±8.66E+00	3.13E+00±2.16E+00	9.53E-02±5.08E-03	8.88E-16±0.00E+00	**0.00E+00±0.00E+00**
	50	2.23E-02±1.48E-03	1.58E+01±4.81E-01	9.28E+00±8.16E+00	2.97E+00±2.51E-01	8.88E-16±0.00E+00	**0.00E+00±0.00E+00**
F5	30	5.65E-07±2.15E-07	1.19E+00±6.41E-02	7.68E-01±5.74E-02	5.02E-02±1.12E-02	1.7E-155±2.4E-145	**0.00E+00±0.00E+00**
	50	8.25E-04±7.17E-05	2.94E+01±2.78E-01	1.43E+01±2.12E-01	5.48E-01±3.97E-02	1.3E-154±6.2E-142	**0.00E+00±0.00E+00**
F6	30	1.04E-06±2.15E-07	1.09E+01±4.77E-01	2.93E+00±1.45E-01	4.55E-01±7.92E-02	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**
	50	7.46E-01±1.43E+00	9.21E+01±2.59E+00	2.84E+01±3.90E+00	8.37E+00±9.32E-02	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**
F7	30	4.21E-01±9.55E-02	1.29E+06±2.64E+05	8.22E+01±3.92E+00	1.21E-01±4.18E-02	**0.00E+00±0.00E+00**	7.58E-04±8.26E-04
	50	6.65E+00±6.67E-01	1.41E+08±3.64E+07	1.06E+05±1.21E+04	1.53E+04±6.80E+03	**0.00E+00±0.00E+00**	9.08E-04±4.16E-04
F8	30	7.33E-06±1.18E-06	3.39E+04±3.36E+03	4.90E+03±3.99E+01	2.16E+01±1.95E+01	1.9E-283±0.00E+00	**0.00E+00±0.00E+00**
	50	1.13E+00±7.60E-02	3.56E+06±2.77E+05	2.71E+05±3.37E+04	3.63E+03±2.68E+02	2.0E-283±0.00E+00	**0.00E+00±0.00E+00**
F9	30	1.06E-05±2.22E-06	2.13E+05±8.77E+03	6.09E+04±6.81E+03	1.21E+03±1.18E+03	1.9E-307±0.00E+00	**0.00E+00±0.00E+00**
	50	2.12E+00±3.84E+00	4.53E+06±4.54E+05	1.68E+06±6.14E+05	1.02E+05±2.52E+04	3.8E-306±0.00E+00	**0.00E+00±0.00E+00**
F10	30	**0.00E+00±0.00E+00**	1.34E-07±1.71E-07	9.64E-08±8.69E-08	**0.00E+00±0.00E+00**	8.96E-06±2.86E-05	**0.00E+00±0.00E+00**
	50	**0.00E+00±0.00E+00**	4.55E-07±5.04E-07	3.73E-07±4.70E-07	**0.00E+00±0.00E+00**	2.73E-05±2.73E-04	**0.00E+00±0.00E+00**
F11	30	5.16E-02±5.11E-02	2.21E+00±1.74E+00	1.13E+00±6.35E-01	3.65E-02±3.41E-02	1.9E-155±7.8E-148	**0.00E+00±0.00E+00**
	50	1.26E-01±1.23E-01	9.88E+00±8.69E+00	7.64E+00±6.53E+00	3.81E-01±3.30E-02	7.2E-155±7.5E-143	**0.00E+00±0.00E+00**
F12	30	2.62E-02±2.02E-02	2.16E+01±1.55E+01	1.36E+01±8.02E+00	7.87E-02±7.82E-03	1.5E-323±0.00E+00	**0.00E+00±0.00E+00**
	50	7.65E-02±6.54E-02	8.18E+00±7.31E-01	6.23E+00±4.49E+00	7.18E-02±1.01E-02	9.5E-312±0.00E+00	**0.00E+00±0.00E+00**
F13	30	7.89E-08±5.11E-08	4.09E+02±1.90E+02	1.13E+00±7.99E-01	5.57E-03±3.69E-03	**0.00E+00±0.00E+00**	2.09E-09±6.36E-09
	50	1.69E-03±7.44E-04	1.01E+04±1.39E+03	9.95E+01±2.61E+01	5.61E-01±4.76E-01	**0.00E+00±0.00E+00**	2.62E-05±4.23E-05

and feasibility of WCSDE. In the Wilcoxon test, the '+' means that the proposed algorithm is superior to the selected algorithm, '−' means the opposite, '=' implies that the two algorithms get the same result, and the Rank column is the accuracy ranking of their average solutions.

Table 4 illustrates that the performance of the WCSDE algorithm is much better than that of other CSs, except that the performance is slightly inferior to SDCS in optimizing the F7 and F13, and the best results are obtained when processing other functions. In addition, the Wilcoxon rank-sum test is used to verify the performance comparison ranking results of the improved CS algorithm in different dimensions. The performance of the WCSDE algorithm ranks first compared with other variants, showing the superiority of the algorithm performance.

Figures 17-29 are the second part of the experimental convergence curve. These figures show that WCSDE and SDCS still have excellent performance when the number of function evaluations is used as the criterion. WCSDE can quickly converge to the global optimal value within 3000 times of function evaluation when optimizing F2, F3, F4, and F6. In the case of function F4, CS, MCS, DECS, and SDCS all fall into the local optimum, which leads to the stagnation of the search. Only the WCSDE jumps out of the local optimum and finds the global optimum. In the unimodal function F10, although the DECS and ECS also can see the global optimal value within the limited standard, the search efficiency of WCSDE is higher, and the optimal solution is obtained in fewer function evaluation times. When optimizing F1, F5, F8, F9, F11, and F12, the WCSDE has a faster downward trend than the SDCS, and the convergence

TABLE 6. The wilcoxon test results of the experimental results fixed function evaluation times.

	Dim	ECS		CS		MCS		DECS		SDCS		WCSDE
		Wilcox	Rank	Wilcox	Rank	Wilcox	Rank	Wilcox	Rank	Wilcox	Rank	Rank
F1	30	+	3	+	6	+	5	+	4	+	2	1
	50	+	3	+	6	+	5	+	4	+	2	1
F2	30	+	4	+	6	+	5	+	3	=	1	1
	50	+	3	+	6	+	4	+	5	=	1	1
F3	30	+	3	+	6	+	4	+	5	=	1	1
	50	+	3	+	6	+	4	+	5	=	1	1
F4	30	+	3	+	6	+	5	+	4	+	2	1
	50	+	3	+	6	+	5	+	4	+	2	1
F5	30	+	3	+	6	+	5	+	4	+	2	1
	50	+	3	+	6	+	5	+	4	+	2	1
F6	30	+	3	+	6	+	5	+	4	=	1	1
	50	+	3	+	6	+	5	+	4	=	1	1
F7	30	+	4	+	6	+	5	+	3	-	1	2
	50	+	3	+	6	+	5	+	4	-	1	2
F8	30	+	3	+	6	+	5	+	4	+	2	1
	50	+	3	+	6	+	5	+	4	+	2	1
F9	30	+	3	+	6	+	5	+	4	+	2	1
	50	+	3	+	6	+	5	+	4	+	2	1
F10	30	=	1	+	5	+	4	=	1	+	6	1
	50	=	1	+	5	+	4	=	1	+	6	1
F11	30	+	4	+	6	+	5	+	3	+	2	1
	50	+	3	+	6	+	5	+	4	+	2	1
F12	30	+	3	+	6	+	5	+	4	+	2	1
	50	+	4	+	6	+	5	+	3	+	2	1
F13	30	+	3	+	6	+	5	+	4	-	1	2
	50	+	3	+	6	+	5	+	4	-	1	2
Ave			3.00		5.92		4.81		3.73		1.92	1.15
Final			3		6		5		4		2	1

efficiency is higher. However, in functions F7 and F13, the performance is not as good as SDCS, especially F7 also falls into the local optimum.

Table 5 shows that the WCSDE has a slight lack of performance in optimizing the F7 and F13 functions, and the theoretical optimal solution is found in the other functions. In addition, the WCSDE has better convergence and can reach the global optimum, but the different five CS algorithms cannot converge while optimizing the functions F1, F4, F5, F8, F9, F10, F11, and F12. The ECS is affected by high dimensionality in processing 50-dimensional F1, F4, F5, F6, F8, F9, and F13, and the performance is reduced, while WCSDE shows better high-dimensional performance. The comparison of Table 5 and Table 3 concludes that a large number of function evaluation times in the second experiment show the advantage of the WCSDE nonlinear inertia weight in the later stage of narrowing the optimization stride. It can find the theoretical optimal solution through powerful local search searchability.

Table 6 depicts that the WCSDE algorithm is better than MCS, DECS, ECS, and SDCS, especially compared with the standard CS algorithm, which has a significant improvement. The performance is slightly inferior to SDCS in the optimized F7 and F13 functions. The best results are obtained when

processing other functions. Finally, by ranking the Rank average, it is also evident that WCSDE has an excellent performance in optimizing the above functions.

By verifying two experiments, the WCSDE highlights powerful searchability, strong local searchability, adaptive ability, and robustness. This remarkable effect can be obtained due to use the nonlinear inertia weight to dynamically adjust the search range, enhance the adaptive capacity of the algorithm, and improve the later local searchability. The information exchange among populations is established to enhance the searchability of WCSDE by combining it with the advantages of the DE algorithm.

V. CONCLUSION

This paper presents a novel hybrid algorithm WCSDE to make up for the inherent defects of the standard CS and significantly improve the optimization performance based on the following two points. On the one hand, a nonlinear inertial weight strategy during the bird's nest position update stage is employed to seek the balance between exploration and development, and to enhance the global optimization ability in the early stage and the local optimization ability in the later stage. On the other hand, after updating the position of the population, the mutation and cross-selection strategies are used to effectively improve the search accuracy by combining the gradient information of the optimal solution and the sub-optimal solution and transmit the information among different levels of solutions. The experimental results and related statistical analysis show that the WCSDE algorithm has better search accuracy and robustness than the other five CS algorithms. In the future, we will try to settle some industrial problems with the constrained conditions to using the WCSDE algorithm for improving the diversity and applicability of the algorithm.

REFERENCES

[1] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proc. World Congr. Nature Biologically Inspired Comput. (NaBIC)*, 2009, pp. 210–214.
[2] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Mach. Learn.*, vol. 3, nos. 2–3, pp. 95–99, 1988.
[3] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," in *Proc. Eur. Conf. Artif. Life*. Paris, France, 1991, pp. 134–142.
[4] G. C. Chen and Y. U. Jin-Shou, "Particle swarm optimization algorithm," *Inf. Control*, vol. 186, no. 3, pp. 454–458, 2005.
[5] A. Gherboudj, A. Layeb, and S. Chikhi, "Solving 0–1 knapsack problems by a discrete binary version of cuckoo search algorithm," *Int. J. Bio-Inspired Comput.*, vol. 4, no. 4, pp. 229–236, 2012.
[6] A. Ouaarab, B. Ahiod, and X. S. Yang, "Discrete cuckoo search algorithm for the travelling salesman problem," *Neural Comput. Appl.*, vol. 24, pp. 1659–1669, 2014.
[7] M. K. Marichelvam, T. Prabaharan, and X. S. Yang, "Improved cuckoo search algorithm for hybrid flow shop scheduling problems to minimize makespan," *Appl. Soft Comput.*, vol. 19, no. 1, pp. 93–101, Jun. 2014.
[8] G. V. S. K. Karthik and S. Deb, "A methodology for assembly sequence optimization by hybrid cuckoo-search genetic algorithm," *J. Adv. Manuf. Syst.*, vol. 17, no. 1, pp. 47–59, Mar. 2018.
[9] A. R. Yildiz, "Cuckoo search algorithm for the selection of optimal machining parameters in milling operations," *Int. J. Adv. Manuf. Technol.*, vol. 64, nos. 1–4, pp. 55–61, Jan. 2013.
[10] A. Mohamad, A. M. Zain, N. E. N. Bazin, and A. Udin, "A process prediction model based on cuckoo algorithm for abrasive waterjet machining," *J. Intell. Manuf.*, vol. 26, no. 6, pp. 1247–1252, Dec. 2015.
[11] B. Yang, J. Miao, Z. C. Fan, J. Long, and X. H. Liu, "Modified cuckoo search algorithm for the optimal placement of actuators problem," *Appl. Soft Comput.*, vol. 67, pp. 48–60, Jun. 2018.
[12] N. M. Nawi, A. Khan, M. Z. Rehman, T. Herawan, and M. M. Deris, "CSLMEN: A new cuckoo search Levenberg Marquardt Elman network for data classification," in *Recent Advances on Soft Computing and Data Mining. Advances in Intelligent Systems and Computing*, vol. 287. Cham, Switzerland: Springer, 2014, pp. 173–182.
[13] E. Daniel and J. Anitha, "Optimum wavelet based masking for the contrast enhancement of medical images using enhanced cuckoo search algorithm," *Comput. Biol. Med.*, vol. 71, pp. 149–155, Apr. 2016.
[14] Y. Zhang, L. Wang, and Q. Wu, "Modified adaptive cuckoo search (MACS) algorithm and formal description for global optimisation," *Int. J. Comput. Appl. Technol.*, vol. 44, no. 2, p. 73, 2012.
[15] S. Walton, O. Hassan, K. Morgan, and M. Brown, "Modified cuckoo search: A new gradient free optimisation algorithm," *Chaos, Solitons Fractals*, vol. 44, no. 9, pp. 710–718, 2011.
[16] E. Valian, S. Tavakoli, S. Mohanna, and A. Haghi, "Improved cuckoo search for reliability optimization problems," *Comput. Ind. Eng.*, vol. 64, no. 1, pp. 459–468, 2013.
[17] Y. G. Xue and H. W. Deng, "The cuckoo search algorithm based on dynamic grouping to adjust flight scale," *Appl. Mech. Mater.*, vol. 543, pp. 1822–1826, Mar. 2014.
[18] U. Mlakar, I. Fister, and I. Fister, "Hybrid self-adaptive cuckoo search for global optimization," *Swarm Evol. Comput.*, vol. 29, pp. 47–72, Aug. 2016.
[19] K. N. A. Rani, W. F. Hoon, M. F. A. Malek, N. A. M. Affendi, L. Mohamed, N. Saudin, A. Ali, and S. C. Neoh, "Modified cuckoo search algorithm in weighted sum optimization for linear antenna array synthesis," in *Proc. IEEE Symp. Wireless Technol. Appl. (ISWTA)*, Sep. 2012, pp. 210–215.
[20] M. Tuba, M. Subotic, and N. Stanarevic, "Modified cuckoo search algorithm for unconstrained optimization problems," in *Proc. 5th Eur. Conf. Eur. Comput. Conf.*, Madison, WI, USA: WSEAS Press, 2011, pp. 263–268.
[21] J. Cheng, L. Wang, Q. Jiang, Z. Cao, and Y. Xiong, "Cuckoo search algorithm with dynamic feedback information," *Future Gener. Comput. Syst.*, vol. 89, pp. 317–334, Dec. 2018.
[22] X. Li, J. Wang, and M. Yin, "Enhancing the performance of cuckoo search algorithm using orthogonal learning method," *Neural Comput. Appl.*, vol. 24, no. 6, pp. 1233–1247, May 2013.
[23] T. T. Nguyen and T. T. Nguyen, "An improved cuckoo search algorithm for the problem of electric distribution network reconfiguration," *Appl. Soft Comput.*, vol. 84, Nov. 2019, Art. no. 105720.
[24] J. Li, Y.-X. Li, S.-S. Tian, and J.-L. Xia, "An improved cuckoo search algorithm with self-adaptive knowledge learning," *Neural Comput. Appl.*, vol. 32, no. 16, pp. 11967–11997, Aug. 2020.
[25] Z. Zhang, S. Ding, and W. Jia, "A hybrid optimization algorithm based on cuckoo search and differential evolution for solving constrained engineering problems," *Eng. Appl. Artif. Intell.*, vol. 85, pp. 254–268, Oct. 2019.
[26] S. I. Boushaki, N. Kamel, and O. Bendjeghaba, "A new quantum chaotic cuckoo search algorithm for data clustering," *Expert Syst. Appl.*, vol. 96, pp. 358–372, Apr. 2018.
[27] X. Liu and M. Fu, "Cuckoo search algorithm based on frog leaping local search and chaos theory," *Appl. Math. Comput.*, vol. 266, pp. 1083–1092, Jul. 2015.
[28] X. Li and M. Yin, "A particle swarm inspired cuckoo search algorithm for real parameter optimization," *Soft Comput.*, vol. 20, no. 4, pp. 1389–1413, 2016.
[29] W. C. E. Lim, G. Kanagaraj, and S. G. Ponnambalam, "A hybrid cuckoo search-genetic algorithm for hole-making sequence optimization," *J. Intell. Manuf.*, vol. 27, no. 2, pp. 417–429, Apr. 2016.
[30] X. S. Yang, *Nature-Inspired Metaheuristic Algorithms*, 2nd ed. Luniver Press, 2010.
[31] R. Storn and K. Price, "Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.
[32] A. M. Kamoona and J. C. Patra, "A novel enhanced cuckoo search algorithm for contrast enhancement of gray scale images," *Appl. Soft Comput.*, vol. 85, Dec. 2019, Art. no. 105749.
[33] P. Ong and Z. Zainuddin, "Optimizing wavelet neural networks using modified cuckoo search for multi-step ahead chaotic time series prediction," *Appl. Soft Comput.*, vol. 80, pp. 374–386, Jul. 2019.
[34] H. Rakhshani and A. Rahati, "Snap-drift cuckoo search: A novel cuckoo search optimization algorithm," *Appl. Soft Comput.*, vol. 52, pp. 771–794, Apr. 2017.

CHENG-XU ZHANG was born in Zhenjiang, China, in 1997. He received the B.S. degree in automation from the Tongda College of Nanjing University of Posts and Telecommunications, in 2019. He is currently pursuing the master's degree in information and communication engineering with Jishou University. His main research interests include soft computing techniques and neural networks.

KAI-QING ZHOU was born in Changsha, China, in 1984. He received the B.S. degree in computer science and technology from Jishou University, in 2006, the M.S. degree in computer applied techniques from the Changsha University of Science and Technology, in 2011, and the Ph.D. degree in computer science from Universiti Teknologi Malaysia, in 2016. He was Postdoctoral Fellow with the College of Information and Engineering, Central South University, from 2016 to 2018. He is currently an Associate Professor with the Department of Data Science and Big Data Technology, College of Information and Engineering, Jishou University. His main research interests include fuzzy Petri net and its applications, clinical decision support systems, and soft computing techniques.

SHAO-QIANG YE was born in Changsha, China, in 1996. He received the B.S. degree in network engineering from Jishou University, in 2019, where he is currently master's degree in intelligent computing and its applications. His main research interests include soft computing techniques and cloud computing.

AZLAN MOHD ZAIN (Member, IEEE) received the Ph.D. degree in computer science from Universiti Teknologi Malaysia, in 2010. He is currently a Professor of computer science with the Big Data Center, Universiti Teknologi Malaysia. His main research interests include artificial intelligence, modeling and optimization, machining, and statistical process control.