Applied Soft Computing Journal 81 (2019) 105481

Contents lists available at ScienceDirect

Applied Soft Computing Journal

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

Riesz fractional derivative Elite-guided sine cosine algorithm

Wenyan Guo *, Yuan Wang, Fengqun Zhao, Fang Dai

School of Science, Xi'an University of Technology, Yanxiang Street 58th, Xian 710054, China

HIGHLIGHTS

- An improved Sine Cosine Algorithm with mutation strategy is proposed.
- In new algorithm, we construct a new Riesz fractional derivative mutation strategy to enhance its efficiency and accuracy.
- The new algorithm uses probabilistic QOL and BOL to enhance the ability of global exploration of the population.
- The new algorithm tests on two sets of benchmark problems, 23 benchmark problems and CEC 2017 benchmark problems.

ARTICLE INFO

Article history:
Received 30 July 2018
Received in revised form 15 March 2019
Accepted 13 April 2019
Available online 21 May 2019

Keywords:
Sine cosine algorithm
Riesz fractional derivative
Quasi-opposition learning
Opposition based learning
Benchmark test problems
Engineering application problems

ABSTRACT

In order to improve the calculation accuracy of the sine cosine algorithm (SCA), Riesz fractional derivative sine cosine algorithm (RFSCA) based on the Riesz fractional derivative mutation strategy is proposed. The new algorithm uses quasi-opposition learning to initialize the population, which can increase the diversity of the population. Based on the approximate formula of Riesz fractional derivative with second-order accuracy, we construct a new mutation approach to update the optimal individual and improve the calculation accuracy of the algorithm. Furthermore, the proposed method is integrated into quasi-opposition learning and opposition-based learning strategies to enhance the ability of global exploration of the population and improve the convergence speed of the algorithm. The new algorithm is tested in two sets of test sets (classical benchmark of 23 problems and standard IEEE CEC 2017). The simulation experiments demonstrate that the proposed algorithm significantly outperforms the latest heuristic-based algorithms in both exploration, exploitation and solution quality. Two engineering questions (Welded beam design, pressure vessel design) are applied to confirm the superior performance of proposed algorithm.

1. Introduction

The Sine Cosine Algorithm (SCA) [1] was proposed by Mirjalili in 2016. This kind of algorithm uses the properties of the sine and cosine function in guiding the population to search for the global optimal solution from the global optimal individuals. The properties that the algorithm only use few parameters and quick convergence speed make this method become widely used. In this algorithm, the random and adaptive parameters balance the exploitation and exploration capabilities of the algorithm. However, it also fails in computational accuracy and it may have premature convergence. A new approach based on Opposition-Based learning Sine Cosine Algorithm (OBSCA) [2] has been recently proposed. The algorithm uses the opposition-based learning strategy in both the initialization of the algorithm and the entire population update process, which enhances the SCA population diversity and global exploration ability, but the exploitation capability still needs to be strengthened. Nenavath et al. [3] use differential evolution strategy to improve SCA to solve target tracking problem. Rizk-Allah [4] adopts multi-orthogonal search strategy to improve SCA and enhances the accuracy of SCA, the MOSCA efficiently enrich the exploratory capabilities of the search by introducing the MOSS.

Mutation Operators are commonly used to improve population diversity and enhance the algorithm's ability to jump out of local optima, which is first used in genetic algorithm and differential evolution algorithm. Later, many scholars incorporated mutation ideas into other swarm intelligence algorithms in order to improve the calculation accuracy of the algorithm. The common mutation strategies which based on the probability distribution are Gaussian mutation (GM) [5], Cauchy mutation (CM) [6,7], or the combination of the GM and CM, and the Wavelet mutation (WM) based on wavelet theory [8], etc. These mutation strategies are intend to add random disturbances near individuals to search a better solution. Derivative is a mathematical tool to study the rate of change. However, these ideas are hard to apply in engineering optimization problems because

* Corresponding author.
E-mail address: wyguo@xaut.edu.cn (W. Guo).

https://doi.org/10.1016/j.asoc.2019.04.044
1568-4946/

of their strong requirement on the whole property of functions rather than the estimation of function value obtained from numerical approximate calculation method. Therefore, the method of approximate solution to calculate the estimated derivative is often used in the design of optimization algorithm. The Fractional Derivative [9,10] (FD) is a generalization of the integer derivative, which is essentially the calculus of any order derivative and is an important branch of mathematics developed from n-times-derivatives and n-times-integrals. The evolution direction produced by the fractional derivative is more ergodic than the evolution direction produced by the integer derivative. This paper introduces the idea of fractional derivative for the first time in the swarm intelligent algorithm, and employ the fractional derivative to generate a more reasonable optimization evolution direction. This kind of method enhances the local exploitation ability of the algorithm and improves the performance of the algorithm. Fractional derivative can simulate many phenomena of biological evolution. Riesz Fractional Derivatives is the ideal calculation method for Fractional Partial Differential Equations on finite fields, it has a certain memory preservation, and it can effectively transfer information (Elite individual) in evolution. Riesz Fractional Derivatives retains the good information of the population and enhanced exploitation capability. Furthermore, the quasi-opposition learning strategy has better exploration capability than the opposition-based learning strategy. The two strategies are carried out by means of probability selection and retain their respective advantages. In this paper, we employ Riesz Fractional Derivatives [11] with second order accuracy and learning strategies to improve sine cosine algorithm. Then we propose a Riesz fractional derivative elite-guided sine cosine algorithm (RFSCA).

The rest paper is organized as follows. Section 2 presents the original sine cosine algorithm. In Section 3 gives the motives and innovations of this paper and describes RFSCA with two effective techniques. In Section 4, a comprehensive experimental study is carried out with two test sets. Furthermore, the control parameter α and p of RFSCA is discussed in this section. Also, two application examples on welded beam design and pressure vessel design problem are given in this section. Finally, some conclusions are drawn in Section 5. The experimental results of two test sets and the calculation results of two classical engineering problems verify the effectiveness of RFSCA.

2. Sine cosine algorithm

For n-dimensional optimization problem

$$\begin{aligned} &\min f\left(x_1, x_2, \ldots, x_n\right) \\ &\text{s.t. } l_i \leq x_i \leq u_i, i = 1, 2, \ldots, n \end{aligned} \tag{1}$$

where x_i is the ith decision variable and u_i, l_i are the upper and lower bounds of x_i. For the problem (1), the sine cosine algorithm utilizes the oscillating property of the sine and cosine functions to modify the ability of the individual to learn from the global optimal solution. The specific process is as follows: assuming that the population size is N and the position of the ith individual in the tth generation is $x_i^t = \left(x_{i1}^t, x_{i2}^t, \ldots, x_{in}^t\right), i = 1, 2, \ldots, N$, we can calculate the fitness value $f\left(x_i^t\right)$ for each individual and record the position $x_*^t \left(x_*^t = \arg\min f\left(x_i^t\right)\right)$ of the optimal individual. The jth dimension of the ith individual in the population is updated according to Eq. (2):

$$x_{ij}^{t+1} = \begin{cases} x_{ij}^t + r_1 \cdot \sin\left(r_2\right) \cdot \left|r_3 \cdot x_{*j}^t - x_{ij}^t\right| & r_4 < 0.5 \\ x_{ij}^t + r_1 \cdot \cos\left(r_2\right) \cdot \left|r_3 \cdot x_{*j}^t - x_{ij}^t\right| & r_4 \geq 0.5 \end{cases} \tag{2}$$

Algorithm 1:Sine Cosine Algorithm

```
Initialize a set of search agents (solutions)( x )
Do
    Evaluate each of the search agents by the objective function
    Update the best solution obtained so far ( x* )
    Update r1, r2, r3, and r4
    Update the position of search agents using Eq.(2)
While(t < maximum number of iterations)
Return the best solution obtained so far as the global optimum
```

Fig. 1. The pseudo-code of SCA.

where $r_2 \in (0, 2\pi)$, $r_3 \in (0, 2)$, $r_4 \in (0, 1)$ are random numbers of uniform distribution, r_1 is the control parameter, and

$$r_1 = a\left(1 - \frac{t}{t_{\max}}\right) \tag{3}$$

where a is a constant, and $t, t_{\max}$ are the current number of iterations and the maximum number of iterations respectively. The pseudo-code of SCA is shown in Fig. 1.

3. Riesz fractional derivative elite-guided sine cosine algorithm

In the sine cosine algorithm, individuals are not only guided by the global optimal individuals, but also learning from the best individuals and evolving the population toward the target location, which can effectively speed up the convergence of the swarm intelligence algorithm. Based on it, we can deduce that the quality of the global optimal individual is very crucial, and the global optimal individual in the SCA without itself updated strategy. Therefore, we apply the Riesz fractional derivative mutation strategy to update the optimal individuals in the population and improve the convergence speed and calculation accuracy of SCA. We also adopt the quasi-opposition learning strategy and the opposition-based learning strategy according to probability to balance the exploration and exploitation capabilities of SCA.

3.1. Riesz fractional derivative mutation strategy

In the sine cosine algorithm, there is no update strategy for the global optimal solution. However, the global optimal solution has an important influence on the nature of the algorithm. In this study, we use the Riesz fractional derivative to update the global optimal solution $\boldsymbol{x}_*$. The Riesz fractional derivative [11] is defined as follows.

Definition. Let function $g(x)$ be continuous in interval $[0, x]$. Divide the interval $[0, x]$ into M equal parts and the step length $h = \frac{x}{M}$. The approximate formula for the α order Riesz fractional derivative of $g(x)$ with second-order accuracy is as follows.

$$\frac{\partial^\alpha g(x)}{\partial |x|^\alpha} = -h^{-\alpha} \sum_{k=-\infty}^{+\infty} \omega_k^* g(x - kh) + O\left(h^2\right) \tag{4}$$

where $\omega_k^* = \frac{(-1)^k \Gamma(\alpha+1)}{\Gamma\left(\frac{\alpha}{2}-k+1\right)\Gamma\left(\frac{\alpha}{2}+k+1\right)}$, $k = 0, \pm 1, \pm 2, \ldots$, ω_k^* are the coefficients of the formula of Riesz's fractional derivative. Γ is *Gamma* function. Omit high order items, Eq. (4) can be written as.

$$\frac{\partial^\alpha g(x)}{\partial |x|^\alpha} \approx -h^{-\alpha} \left[\sum_{k=0}^{+\infty} \omega_k g(x - kh) + \sum_{k=-\infty}^{0} \omega_k g(x - kh)\right] \tag{5}$$

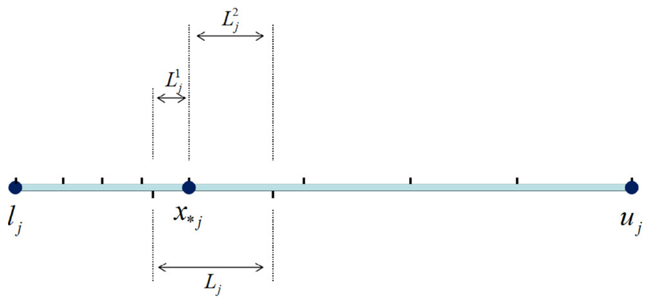

Fig. 2. Interval division.

where $\omega_0 = \frac{\omega_0^*}{2}, \omega_k = \omega_k^*, k = \pm 1, \pm 2, \ldots, \alpha \in [0, 1]$ indicates the rate of change. In this paper we take $\alpha \in [0, 1]$. When it gives the cutoff on $[-M, M]$, the Eq. (5) can be written as.

$$\frac{\partial^\alpha g(x)}{\partial |x|^\alpha} \approx -h^{-\alpha} \left[\sum_{k=0}^{M} \omega_k g(x - kh) + \sum_{k=M}^{0} \omega_k g(x - kh) \right] \tag{6}$$

where the parameters h, α, ω_k are the same as Eq. (5). Eq. (6) evidence the rate of change at point x utilizes the contribution of $2M$ points that are symmetrically equidistant from x. In this paper, we take $M = 2$. In order to enhance the local search ability of SCA, we select randomly the jth dimension from the optimal solution $\boldsymbol{x}_* = (x_{*1}, x_{*2}, \ldots, x_{*n})$ in the SCA evolutionary population G_1^{t+1} to perform mutation of fractioned derivatives to generate a new solution in the optimal solution neighborhood. As the element of the optimal solution x_*, the neighborhood of x_{*j} cannot be too large, so the range of neighborhood x_{*j} decreases as the number of iterations increase. Since the distance from x_{*j} to the two endpoints of interval $[l_j, u_j]$ is not equal. Therefore, we think about a quarter of the distance between x_{*j} and the lower l_j and upper u_j bounds as the acceptance range. We consider the adaptive interval partition according to the number of iterations. Let L_j^1 and L_j^2 represent the adaptive length of x_{*j} from the left and right endpoints, respectively, they can be automatically adjusted as follows

$$L_j^1 = \left(1 - \frac{t}{t_{\max}}\right) \cdot \frac{(x_{*j} - l_j)}{4} \tag{7}$$

$$L_j^2 = \left(1 - \frac{t}{t_{\max}}\right) \cdot \frac{(u_j - x_{*j})}{4} \tag{8}$$

then

$$L_j = L_j^1 + L_j^2 = \frac{(u_j - l_j)}{4} \cdot \left(1 - \frac{t}{t_{\max}}\right) \tag{9}$$

$$x = \left(x_{*j} + \frac{u_j - x_{*j}}{4}\right) - \frac{L_j}{2} \tag{10}$$

L_j indicates the length of the required segmentation interval, where $t, t_{\max}$ are the current number of iterations and the maximum number of iterations, and u_j, l_j are the upper and lower bounds of the jth dimension. The midpoint of the interval is x, The illustration of the process is shown in Fig. 2.

Taking x as the center and dividing L_j into $2M$, the step size h is

$$h = \frac{L_j}{2M + 1} \tag{11}$$

The $2M+1$ endpoints are $x_{*j}^k = x \pm kh, k = 0, 1, 2, \ldots, M$. According to the points, a correction point of $2M + 1\boldsymbol{x}_*$ is generated respectively. Let $\boldsymbol{y}_*^{k-} = (x_{*1}, x_{*2}, \ldots, x_{*j-1}, x - kh, x_{*j+1}, \ldots, x_{*n})$, $\boldsymbol{y}_*^{k+} = (x_{*1}, x_{*2}, \ldots, x_{*j-1}, x + kh, x_{*j+1}, \ldots, x_{*n})$, $k = 0, 1, 2, \ldots, M$, the $\boldsymbol{y}_*^{k-}, \boldsymbol{y}_*^{k+}, k = 0, \pm 1, \pm 2, \ldots, M$ are only different in the jth dimension. Then with regard to problem (1) the jth dimension of $\boldsymbol{x}_*$ corrected based on the Riesz fractional derivative is

$$\widetilde{x}_{*j} = x_{*j} \cdot \left[1 + r \cdot h^{-\alpha} \left(\sum_{k=0}^{M} \omega_k f\left(\boldsymbol{y}_*^{k+}\right) + \sum_{k=-M}^{0} \omega_k f\left(\boldsymbol{y}_*^{k-}\right) \right) \right] \tag{12}$$

where r is a uniformly distributed random number of $[0, 1]$, α, ω_k are same as Eq. (6), h is calculated by Eq. (11). Then a greedy search between the update point $\boldsymbol{y} = (x_{*1}, x_{*2}, \ldots, x_{*j-1}, \widetilde{x}_{*j}, x_{*j+1}, \ldots, x_{*n})$ and the best solution $\boldsymbol{x}_*$.

3.2. Probabilistic learning strategy

The quasi-opposition learning strategy (QOL) [12,13] and opposition-based learning (OBL) [14,15] are effective methods to enhance population diversity, improve solution performance, and speed up the convergence of the algorithm.

After considering the Riesz fractional derivatives in updating the optimal individuals, the QOL strategy and the OBL strategy are updated according to the probability p to improve the exploration ability of the algorithm.

For the current population, when the QOL strategy is used, the quasi opposition point $\boldsymbol{x}_i^{qo}$ of the ith individual $\boldsymbol{x}_i^t$ is $\boldsymbol{x}_i^{qo} = (x_{i1}^{qo}, x_{i2}^{qo}, \ldots, x_{in}^{qo})$, where

$$x_{ij}^{qo} = rand\left(\frac{l_j + u_j}{2}, l_j + u_j - x_{ij}^t\right), i = 1, 2, \ldots, N, j = 1, 2, \ldots, n \tag{13}$$

where u_j, l_j and x_{ij}^t are the upper and lower bounds and element of the jth dimension of $\boldsymbol{x}_i^t$, respectively. $rand(a, b)$ denotes the uniform distribution on the interval (a, b). If the current population is implemented the OBL strategy, the opposition point is $\boldsymbol{x}_i^{ob} = (x_{i1}^{ob}, x_{i2}^{ob}, \ldots, x_{in}^{ob})$, where

$$x_{ij}^{ob} = l_j + u_j - x_{ij}^t, i = 1, 2, \ldots, N, j = 1, 2, \ldots, n \tag{14}$$

for all individuals in the population, the QOL strategy and OBL strategy are executed according to a certain probability p, which takes into account the advantages of the two methods and obtains a high-quality population. The ith new individual $\boldsymbol{ux}_i^t$ can be obtained from Eq. (15)

$$\boldsymbol{ux}_i^t = \begin{cases} \boldsymbol{x}_i^{qo}, r > p \\ \boldsymbol{x}_i^{ob}, r \le p \end{cases}, i = 1, 2, \ldots, N \tag{15}$$

where r is a random number distributed uniformly over interval $[0, 1]$. p is a control parameter which denotes the given probability. In this paper, $p = 0.5$. Individuals are ranked according to the fitness value, and a better individual is selected as a new generation of population.

3.3. The proposed algorithm

Based on the two modification which are Riesz fractional derivative mutation strategy and probability learning strategies for improving SCA, we propose a sine cosine algorithm based on Riesz fractional derivative mutation which named RFSCA. The steps of the RFSCA are as follows:

Step1: Let $t = 0$, initialize a set of search agents using a random function(solutions x) and use the QOL strategy to generate a new agents ($\boldsymbol{x}^{qo}$) from x. Then pick the best N individuals from $\boldsymbol{x}^{qo} \cup \boldsymbol{x}$ as initial population G^t. Record the position of the optimal individual $\boldsymbol{x}_*^t$;

Step2: Using the Eq. (2) to update individuals in G^t to produce population G_1^{t+1}, and update the best individual $\boldsymbol{x}_*^t$;

Table 1
The parameters of algorithm and their values.

Algorithms	Parameters	Value
SCA	a	2
OBSCA	a	2
PSO	$v_{\min}, v_{\max}$	−0.6,0.6
	$w_{\min}, w_{\max}$	0.2,0.9
	c_1, c_2	1.496,1.496
OPSO	$v_{\min}, v_{\max}$	−0.6,0.6
	$w_{\min}, w_{\max}$	0.2,0.9
	c_1, c_2	1.496,1.496
	p_0	0.3
GWO	$a_{\min}, a_{\max}$	0,2
MFO	l	[−1,1]
	b	1

Step3: Generate random number r ($r \in [0, 1]$). If the random number r is greater than the probability p, the quasi-opposition population G_2^{t+1} of population G_1^{t+1} are calculated by Eq. (13). Greedy search for $2N$ individuals in $G_1^{t+1} \cup G_2^{t+1}$ and choose the first N individuals with better fitness as a new generation of population G^{t+1}. Otherwise, a opposition-based learning strategy is performed to update the population G_1^{t+1} to obtained the opposition population G_3^{t+1} of the population G_1^{t+1} by Eq. (14). Greedy search for $2N$ individuals in $G_1^{t+1} \cup G_3^{t+1}$ and choose the first N individuals with better fitness as a new generation of population G^{t+1}. Then update the best individual $\boldsymbol{x}_*^t$;

Step4: Update the best individual $\boldsymbol{x}_*^t$ using Eq. (12)

Step5: Determine whether the algorithm satisfies the termination condition. If not, let $G^t = G^{t+1}$, $t = t + 1$, return Step2; otherwise, output the global optimal value $f\left(\boldsymbol{x}_*^t\right)$.

The pseudo code and flow chart of RFSCA is shown in Figs. 3 and 4.

4. Experiments and discussion

For the stochastic optimization algorithm, select the appropriate test set to test its performance to show that the superiority of the algorithm is necessary.

Therefore, in this section, two different benchmark test set, classical benchmark of 23 problems [16–19] and standard IEEE CEC 2017 [20] have been chosen. The experiment of the test set (classical benchmark of 23 problems) uses iterations as the evaluation criteria of the algorithm, and the test set (standard IEEE CEC 2017) uses FES as the evaluation standard in numerical experiments.

In order to test the performance of the proposed RFSCA, the comparisons among the RFSCA and SCA [1] and its improved algorithms OBSCA [2], PSO [21] and its improved algorithms OPSO [22], GWO [23], MFO [24], SSA [25] are carried out to verification the effectiveness of RFSCA. In RFSCA, $r_2 \in [0, 2\pi]$, $r_3 \in [0, 2]$, $r_4 \in [0, 1]$ are three random parameters. The value of r_1 can be obtained from Eq. (3). The parameters of the comparison algorithm are shown in Table 1.

4.1. Experimental analysis on classical benchmark of 23 problems

In order to verify the validity of the proposed RFSCA, classical benchmark of 23 problems are used in this section to perform a minimum simulation experiment. The expression, dimension, upper and lower bounds of variables, and theoretical optimal values are shown in [16–19].

The experimental parameters in the following comparative experiments are the same, which are list as follows: population size $N = 30$, the number of iterations $t_{\max} = 500$, the dimension $Dim = 30$(Except Fixed-dimension multimodal benchmark functions). All of algorithms are independently executed 20 times, which calculate the average result (Ave) and standard deviation (STD). The Wilcoxon rank-sum-test [26] has been chosen for statistical comparison because it is necessary to distinguish the difference between the two sets of data.

To analyze the performed of algorithm, average result (Ave), Wilcoxon test results and standard deviation (STD) in each functions is shown in Table 2. The "+/=/−" indicates the number of functions of the comparison algorithm "better/equivalent/inferior" to RFSCA, and the "CPUtimes" indicates the sum of the CPU times taken by the SCA, OBSCA, and RFSCA algorithms to run 20 times in all functions, the bold data indicates the best result of problems. In addition, Table 3 show the Wilcoxon rank-sum-test *p-value* corresponding to Table 2, the bold data indicates the results of the current algorithm are similar to RFSCA results.

From the Tables 2 and 3, it can be analyzed that the result of RFSCA is better as compared to classical SCA and OBSCA. The classical SCA performance is similar to RFSCA in F14, F17 and inferior to RFSCA in the result of 21 test functions. OBSCA performs satisfactorily in a Multimodal functions, some of its results (F14, F16, F19, F21, F23) are similar to RFSCA, while OBSCA lost to RFSCA in the remaining 15 functions.

To analyze the convergence performance of algorithm, convergence curve of algorithms from each function is plotted in Fig. 5 (unimodal functions), Fig. 6 (multimodal functions), Fig. 7 (fixed-dimension multimodal functions). From the figures, RFSCA converges faster and convergence accuracy is higher. We can also clearly see from some convergence graphs (F1–F4, F9–F10) that RFSCA is still converging when the number of iterations reaches the maximum, while SCA and OBSCA have stopped converging.

From Table 3, it can be analyzed that the Wilcoxon test results of GWO, PSO, OPSO, MFO, SSA are 9/1/13, 8/3/12, 9/1/13, 6/3/14, 7/2/14. And the Wilcoxon test result of GWO at function F14 is 0.4570, indicating that there was no significant difference between results of the GWO and RFSCA at a significance level of 0.5.

From the Tables 2 and 3, it can be observed that regardless of the Unimodal, Multimodal and Fixed-dimension multimodal functions, RFSCA has the best performance compared to other algorithms. For unimodal functions, the convergence accuracy of RFSCA on F1, F2, F3, F4, F7 is much higher than that of the comparison algorithms, SSA performs best at F6 and all algorithms performance seem to be very weak in the F5, even so, RFSCA also performs better than most comparison algorithms in function F5.

From the result of Multimodal functions it can be observed also that the RFSCA shows very promising results in terms of convergence result as compared to other Algorithms in F9, F10, F11, and RFSCA converges to the theoretical optimal solution on F9 and F11. The performance of MFO is similar to RFSCA and better than other algorithm on F8, the result of OPSO in F12 and PSO in F13, are better than other. For fixed dimension functions, RFSCA converges to the satisfactory results on F15, F21. However, the performance of the results on F14, F17, F20 is general, indicating that the performance of the RFSCA algorithm in solving fixed dimension multimodal optimization problems still needs to be strengthened.

To analyze the distribution of the results, the box-plot (Unimodal functions F1, F3, Multimodal functions F8, F10, Fixed-dimension multimodal functions F21, F23) in each algorithm is plotted in Fig. 8, it is observed that the variance of RFSCA is significantly smaller than the comparison algorithm and there are few outliers. As above, the performance of RFSCA is better than other algorithm, and RFSCA has the characteristics of fast convergence, high convergence precision, small computational complexity and strong robustness. It also effectively coordinates the algorithm exploration and exploitation capabilities and can handle most problems well.

Algorithm 2:Riesz fractional derivative elite-guided Sine Cosine Algorithm

Initialize a set of search agents using a random function(solutions $\boldsymbol{x}$)
Use the QOL strategy to generate a new agents ($\boldsymbol{x}^{qo}$) from $\boldsymbol{x}$, pick the best N individuals from $\boldsymbol{x}^{qo} \cup \boldsymbol{x}$ as initial population
Do
    Evaluate each of the search agents by the objective function
    Update the best solution obtained so far ($\boldsymbol{x}_*$)
    Update r_1, r_2, r_3, r_4
    Update the position of search agents using Eq.(2)
    Generate random numbers r ($r \in [0,1]$)
    If ($r > p$)
        Update the position of search agents using Eq.(13)
    else
        Update the position of search agents using Eq.(14)
    endif
    Update the best solution $\boldsymbol{x}_*$ using Eq.(12)
While(t < maximum number of iterations)
Return the best solution obtained so far as the global optimum

Fig. 3. The pseudo-code of RFSCA.

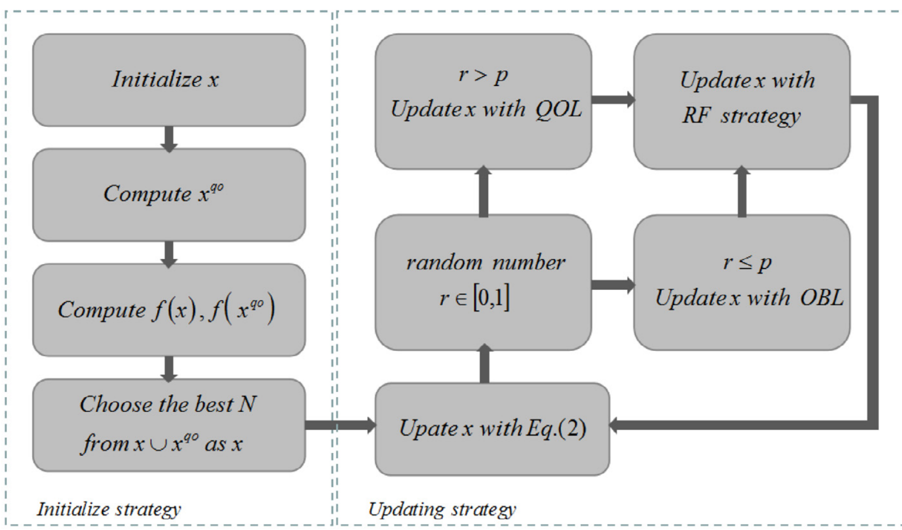

Fig. 4. The flow chart of proposed RFSCA algorithm.

4.2. Analysis of parameter α

The value of parameter α is the key to the Riesz fractional derivative mutation strategy. Since the value of α is unknown, we need to employ simulation experiments to detect the effect from α value on the performance of the algorithm. This section selects functions F1–F23 for experimentation, α takes 11 values at step of 0.1 within the interval [0,1]. The learning strategy uses quasi-inverse learning, and other parameters remain unchanged. Through the difference of experimental results of Friedman test [27], and the results of simulation experiments are sorted. Finally, the order corresponding to different α values is added to obtain total rank. The effect of parameter α on the performance of RFSCA is presented in Table 4.

Table 4 shows that there are seven functions (F6, F8, F12, F13, F17, F18, F20) in the 23 kinds of test functions which are (the Friedman detection value P are lower than 0.05). It indicates that there are a significant difference between the results of 11 kinds of α corresponding values in these functions. However, the results of F17 and F18 converge to the theoretical optimal value, and the results of five other functions are on the same order of magnitude. Therefore, the elite Riesz strategy is not sensitive to the value of α. The rank of 11 kinds of α values are corresponding to the range [23, 253] of the total rank. When the value of α is 0.2, the corresponding result is the best, and the total rank is 123. Therefore, we set α is 0.2 in this paper.

4.3. Analysis of control parameter p

The Eqs. (13) and (14) show that the opposition point obtained by the OBL strategy is relatively fixed and the diversity of the population is enhanced. The QOL strategy has a strong randomness, which accelerates the convergence speed of the algorithm. However, compared with OBL, the QOL strategy is more likely to fall into a local optimum when dealing with multimodal functions. Therefore, the reasonable value of parameter p can make good use of the advantages of OBL strategy and QOL strategy. Supposing 0.1 as the step size, take 11 values of p in the [0, 1]. Where $p = 0$ represents the QOL strategy is only performed and $p = 1$ represents the OBL strategy is only performed. Table 5 demonstrates the ranked mean and standard deviation of Friedman-p for 23 test functions under the same experimental conditions.

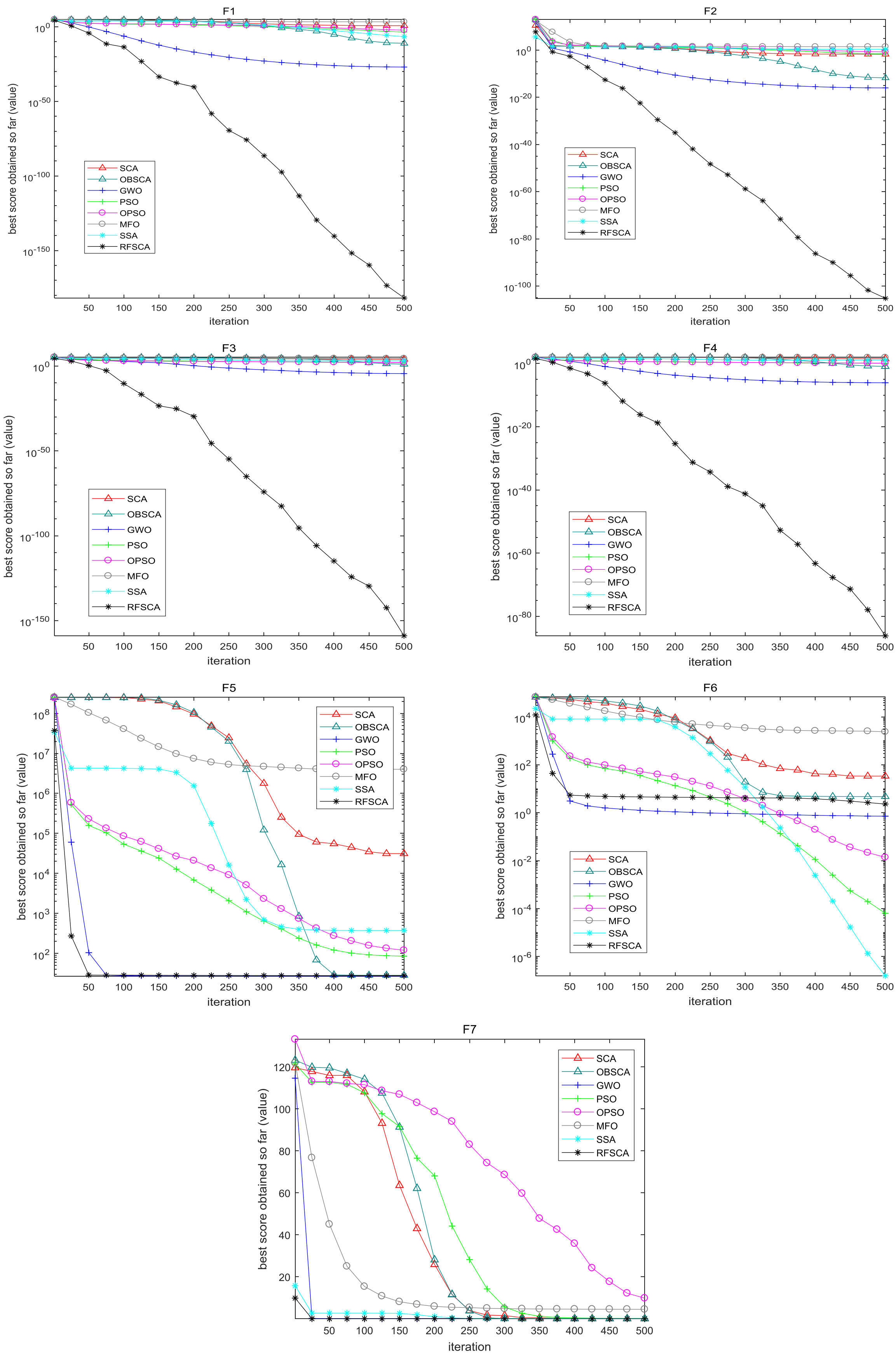

Fig. 5. Convergence graphs for unimodal problems.

Table 2
The statistical comparison proposed method and some swarm intelligence algorithms in F1–F23.

Functions		Algorithms							
		SCA	OBSCA	GWO	PSO	OPSO	MFO	SSA	RFSCA
F1	Ave	5.7639	7.9564e−12	9.7204e−28	2.9459e−04	0.0050	2.0041e+03	2.4355e−07	**1.4624e−182**
	STD	8.9942	3.3105e−11	7.4628e−28	5.0009e−04	0.0034	5.2319e+03	3.6187e−07	**0**
F2	Ave	0.0195	1.8926e−12	1.0204e−16	0.0285	0.1807	31.1313	2.5007	**6.1389e−106**
	STD	0.0210	4.9289e−12	6.7560e−17	0.0196	0.1384	20.9700	2.2936	**2.7378e−105**
F3	Ave	9.8663e+03	11.0920	3.1874e−05	79.2047	120.0785	2.2564e+04	1.3519e+03	**1.2042e−159**
	STD	6.2449e+03	24.0185	1.2770e−04	27.9250	39.0972	8.7717e+03	689.0420	**5.3135e−159**
F4	Ave	30.9892	0.1177	7.9485e−07	1.1296	1.2753	68.4859	10.8288	**6.8596e−87**
	STD	10.7659	0.3084	8.1718e−07	0.2110	0.2344	7.0105	2.9539	**3.0182e−86**
F5	Ave	3.1334e+04	28.5487	**27.0457**	86.1010	120.7562	4.0010e+06	376.6455	27.8192
	STD	4.3495e+04	0.2270	**0.8275**	40.9061	81.3156	1.7876e+07	658.6158	0.6267
F6	Ave	33.8302	4.7451	0.7205	6.4206e−05	0.0138	2.3930e+03	**1.5667e−07**	2.3155
	STD	48.5564	0.2887	0.2817	5.7934e−05	0.0294	5.3531e+03	**1.7485e−07**	0.4577
F7	Ave	0.0826	0.0029	0.0020	0.1758	9.8244	4.5024	0.1463	**1.0733e−04**
	STD	0.1566	0.0016	0.0010	0.0559	13.7695	10.9812	0.0538	**8.4294e−05**
F8	Ave	−3.6943e+03	−3.8259e+03	−5.9648e+03	−5.1881e+03	−8.1482e+03	**−8.6291e+03**	−7.6359e+03	−7.2990e+03
	STD	379.0686	258.1734	1.1317e+03	1.2373e+03	1.2218e+03	**713.5406**	827.2185	994.5160
F9	Ave	38.5256	5.7769e−10	2.5220	64.5113	91.5816	159.4464	49.4991	**0**
	STD	33.7324	1.5948e−09	3.8079	14.2834	27.2653	37.6656	23.9549	**0**
F10	Ave	12.9014	0.8696	1.0179e−13	0.2555	0.4815	16.0860	2.5458	**8.8818e−16**
	STD	9.2276	1.5010	1.5134e−14	0.5119	0.5085	6.7368	0.5851	**0**
F11	Ave	0.9033	3.9831e−06	0.0080	0.0095	0.0059	14.5469	0.0183	**0**
	STD	0.3885	1.7341e−05	0.0150	0.0078	0.0064	44.0100	0.0129	**0**
F12	Ave	6.7709e+03	0.5513	0.0408	0.0052	**9.5742e−05**	10.7130	6.4836	0.3252
	STD	2.3930e+04	0.0647	0.0195	0.0232	**8.3209e−05**	8.8272	2.4986	0.1131
F13	Ave	4.6733e+05	2.5815	0.5900	**0.0071**	0.0097	2.0503e+07	17.6341	1.5103
	STD	1.2004e+06	0.1742	0.2228	**0.0108**	0.0085	9.1693e+07	16.2260	0.3078
F14	Ave	1.6946	1.7036	3.9391	2.5786	2.4705	2.9135	**1.2463**	2.3650
	STD	0.9693	0.9636	3.8845	2.0284	2.9700	3.1500	**0.5462**	3.5762
F15	Ave	0.0011	8.2053e−04	0.0034	9.0799e−04	8.9057e−04	0.0011	0.0039	**6.5150e−04**
	STD	3.6216e−04	2.0977e−04	0.0073	1.8888e−04	1.0426e−04	4.5797e−04	0.0071	**2.3519e−04**
F16	Ave	−1.0316	−1.0316	−1.0316	**−1.0316**	−1.0316	−1.0316	−1.0316	−1.0316
	STD	4.4812e−05	5.3698e−06	9.9032e−09	**1.9729e−16**	1.7646e−16	2.2781e−16	1.6802e−14	1.2090e−05
F17	Ave	0.3994	0.3988	0.3979	**0.3979**	0.3979	0.3979	0.3979	0.3985
	STD	0.0016	6.1889e−04	8.5810e−07	**0**	0	0	1.0757e−14	4.7389e−04
F18	Ave	3.0001	3.0000	3.0000	**3.0000**	3.0000	3.0000	3.0000	3.0000
	STD	1.1938e−04	2.6448e−05	2.6677e−05	**1.4372e−15**	1.6803e−15	2.0325e−15	1.7017e−13	1.1482e−05
F19	Ave	−3.8540	−3.8583	−3.8622	**−3.8628**	−3.8628	−3.8628	−3.8628	−3.8587
	STD	0.0031	0.0020	0.0014	**2.2367e−15**	2.1946e−15	2.2781e−15	4.2277e−11	0.0030
F20	Ave	−2.7706	−3.1068	−3.2521	−3.2566	**−3.2804**	−3.2134	−3.2250	−3.1432
	STD	0.5299	0.0391	0.0771	0.0607	**0.0582**	0.0510	0.0585	0.0570
F21	Ave	−2.6319	−8.7161	−9.0131	−8.1355	−8.3908	−5.7582	−8.1464	**−9.5979**
	STD	2.2686	2.0800	2.3847	2.9007	2.8363	3.1285	3.2099	**0.3594**
F22	Ave	−3.3189	**−9.8822**	−9.7537	−9.0714	−9.2272	−7.1064	−9.1590	−9.7974
	STD	1.9381	**0.3629**	2.0281	2.7724	2.4578	3.4904	2.5938	0.4378
F23	Ave	−3.7434	−10.0400	**−10.5344**	−8.2896	−10.2684	−6.8237	−7.5267	−9.7438
	STD	1.5133	0.3268	**8.8917e−04**	3.2270	1.1987	3.8584	3.5199	0.5246
+/=/−		0/2/21	0/8/15	8/3/12	9/1/13	9/1/13	6/3/14	7/2/14	
CPUtime (s)		19.4469	32.1976						34.0926

From Table 5, we can see that the results of F17–F20 only have a small difference, except for function F14. The RFSCA corresponding to different values of parameter p has a significant difference among the remaining 18 test functions. For the unimodal functions F1–F4 and F6, the results of the QOL strategy have the best performance, indicating that the QOL strategy has a greater contribution to the improvement of the accuracy of the unimodal function. For multimodal functions and fixed-dimensional multimodal functions, the use of probabilistic selection strategy has the best performance, and the QOL strategy has a significant impact. However, the results of only applying QOL strategy is the worst (Total rank is 204, the maximum rank in Table 5). These results indicate the use of QOL strategies for individuals cannot improve algorithm performance effectively. The range of total ranks corresponding of 11 kinds of p values is [23,253]. When the value of p is 0.5, the corresponding algorithms have the best performance (Total rank is 109.5, the minimum rank in Table 5). Therefore, $p = 0.5$ is used as the RFSCA algorithm parameter in this paper.

4.4. Experimental analysis on standard IEEE CEC 2017

To further verify the effectiveness of the proposed algorithm, in this section, challenging standard benchmark test set IEEE CEC 2017 [20] is used to evaluate the performance of proposed algorithm. This test set contains 4 different types of functions

Table 3
p-value obtained from Wilcoxon rank sum test (RFSCA algorithm for comparative samples, $n = 30$).

Functions	SCA		OBSCA		GWO		PSO		OPSO		MFO		SSA	
	p-value		*p-value*		*p-value*		*p-value*		*p-value*		*p-value*		*p-value*	
F1	6.7956e−08	−	6.7956e−08	−	6.7956e−08	−	6.7956e−08	−	6.7956e−08	−	6.7956e−08	−	6.7956e−08	−
F2	6.7956e−08	−	6.7956e−08	−	6.7956e−08	−	6.7956e−08	−	6.7956e−08	−	6.7956e−08	−	6.7956e−08	−
F3	6.7956e−08	−	6.7956e−08	−	6.7956e−08	−	6.7956e−08	−	6.7956e−08	−	6.7956e−08	−	6.7956e−08	−
F4	6.7956e−08	−	6.7956e−08	−	6.7956e−08	−	6.7956e−08	−	6.7956e−08	−	6.7956e−08	−	6.7956e−08	−
F5	6.7956e−08	−	6.7956e−08	−	0.0023	+	4.6804e−05	−	6.7956e−08	−	6.7956e−08	−	4.6804e−05	−
F6	6.7956e−08	−	6.7956e−08	−	6.7956e−08	+	6.7956e−08	+	6.7956e−08	+	0.0033	−	6.7956e−08	+
F7	6.7956e−08	−	6.7956e−08	−	6.7956e−08	−	6.7956e−08	−	6.7956e−08	−	6.7956e−08	−	6.7956e−08	−
F8	6.7956e−08	−	6.7956e−08	−	5.0907e−04	−	5.1658e−06	−	0.0071	+	1.0373e−04	+	**0.2393**	=
F9	6.7956e−08	−	1.6699e−04	−	7.8754e−09	−	8.0065e−09	−	8.0065e−09	−	8.0065e−09	−	8.0065e−09	−
F10	6.7956e−08	−	8.0065e−09	−	7.1806e−09	−	8.0065e−09	−	8.0065e−09	−	8.0065e−09	−	8.0065e−09	−
F11	6.7956e−08	−	2.9920e−08	−	0.0096	−	8.0065e−09	−	8.0065e−09	−	8.0065e−09	−	8.0065e−09	−
F12	6.7956e−08	−	1.2009e−06	−	6.7956e−08	+	6.7956e−08	+	6.7956e−08	+	6.7956e−08	−	6.7956e−08	−
F13	6.7956e−08	−	6.7956e−08	−	1.0646e−07	+	6.7956e−08	+	6.7956e−08	+	6.7956e−08	−	0.0060	+
F14	**0.0531**	=	**0.0810**	=	**0.4570**	=	**0.7141**	=	0.0092	−	**0.1028**	=	2.2214e−04	−
F15	3.7499e−04	−	0.0275	−	0.0468	−	2.4706e−04	−	1.1590e−04	−	0.0040	−	0.0021	−
F16	1.8074e−05	−	**0.5609**	=	2.2178e−07	+	2.8636e−08	+	**0.0758**	=	8.0065e−09	+	6.7956e−08	+
F17	**0.1806**	=	**0.1264**	=	6.7956e−08	+	8.0065e−09	+	8.0065e−09	+	8.0065e−09	+	6.4125e−08	+
F18	0.0123	−	**0.4735**	=	0.0315	−	6.2414e−08	+	8.0065e−09	+	6.3761e−08	+	6.7956e−08	+
F19	1.0373e−04	−	**0.3507**	=	4.5401e−06	+	1.5124e−08	+	2.4037e−08	+	8.0065e−09	+	6.7956e−08	+
F20	9.1266e−07	−	0.0499	−	8.2924e−05	+	4.7072e−06	+	1.1096e−06	+	2.5073e−04	+	4.6804e−05	+
F21	6.7956e−08	−	**0.7972**	=	0.0012	−	**0.1059**	=	0.0302	−	0.0311	−	0.0315	−
F22	6.7956e−08	−	**0.5792**	=	1.5997e−05	−	8.9731e−04	−	9.5287e−04	−	**1**	=	0.0012	−
F23	6.7956e−08	−	**0.0859**	=	6.7956e−08	+	**0.1057**	=	6.0608e−07	+	**1**	=	**0.5979**	=

and the number is 30, Unimodal Functions (F1–F3), Simple Multimodal Functions (F4–F10), Hybrid Functions (F11–120), Composition Functions (F21–F30).

For the sake of fairness, the experimental parameters in the following comparative experiments are the same, which are list as follows: population size $N = 50$, and the number of functional evaluations ($FES = 10000 \cdot Dim$). All of algorithms are independently executed 20 times, which calculate the average result (Ave) and standard deviation (STD), this section of the experiment uses a higher dimension $Dim = 50$. The Wilcoxon rank-sum-test has been chosen for statistical comparison.

Table 6 shows the average result and standard deviation of the eight algorithms running independently 20 times at dimension $Dim = 50$. Table 7 shows Wilcoxon rank sum test of SCA and OBSCA for the 50-dimensional benchmark set. The Wilcoxon rank-sum-test results of SCA and OBSCA are 2/17/11 and 5/13/12. Although those algorithms perform similarly most test functions, it can be seen that RFSCA is significantly better than the classical SCA and OBSCA.

Unimodal functions evaluate the strength of exploitation of algorithm. RFSCA performs better than the improved algorithm OBSCA in unimodal functions, which is similar to the classical SCA algorithm. The problems from F4 to F10 are simple multimodal functions and in these functions RFSCA performs better than classical SCA, and OBSCA outperforms RFSCA only in F5, F10. Hybrid Functions (F11–F20) and Composition Functions (F21–F30) evaluate the stability of exploration and exploitation, it is shows that RFSCA performs better in terms of equalization exploration and exploitation. In addition, RFSCA can obtain better results than SCA and OBSCA in a shorter time because the CPU times of SCA, OBSCA, and RFSCA are 6.0808e+03 s, 5.0692e+03 s, and 5.0473e+03 s.

To analyze the diversity behavior of solutions and exploitation capability, average distance between the solutions in each generation is plotted in Fig. 9. Fig. 10 reflects the impact of the Riesz fractional derivative mutation strategy on the optimal individual, and the ordinate is the coordinate Euclidean distance between current optimal individual and previous optimal individual. In Figs. 9 and 10, the maximum number of iterations set 500, the population size is 50, the dimension $Dim = 50$, and the six test functions are selected are the unimodal function (F1), the simple multimode function (F10), the mixing function (F15, F20), and Combination function (F25, F30).

From the Fig. 9 it can be observed that the RFSCA shows good results in terms of diversity behavior as compared to classical SCA. The diversity behavior curve of RFSCA changes more obviously with the number of iterations. This is because RFSCA has a learning strategy and a Riesz fractional derivative variation strategy, which makes RFSCA conduct local exploitation while taking into account global exploration. If the Riesz fractional derivative variation strategy of the tth iteration is not improved for the optimal individual in Fig. 10, the distance of the tth iteration is zero. From Fig. 9 it can be seen that the update of the optimal individual is frequent, which proves the validity of the Riesz fractional derivative mutation strategy.

4.5. Complexity

The time complexity of the swarm intelligent optimization algorithm is very important, and it depends on the structure of the algorithm and the size of the population. Therefore, the time complexity of RFSCA is affected by population size, number of iterations, and algorithmic strategies, RFSCA consists of three parts: classical SCA update strategy, Learning strategy and Riesz fractional derivative mutation strategy, thus the complexity of the proposed algorithm can be calculated as follows:

$$\begin{aligned} O(RFSCA) &= O\left(positions\quad update\ through\ equations\,of\ SCA\right) \\ &\quad + O\left(OBL\,or\ \ QOL\right) \\ &\quad O\left(best\ \ positions\quad update\ through\ equations\,of\ RF\right) \\ &= O\left(\left((2N+1)\times n\right)\times t_{\max}\right) \end{aligned}$$

Similarly

$$O(SCA) = O\left(N \times n \times t_{\max}\right)$$

$$O(OBSCA) = O\left((2N \times n) \times t_{\max}\right)$$

where N is the population size, n is the dimension, $t_{\max}$ is the maximum number of iterations. Only the global optimal individual is updated in the Riesz fractional derivative variation strategy. The last row of Table 2 shows the sum of CPU times of SCA, OBSCA, RFSCA running 20 times in 23 functions. The CPU times of

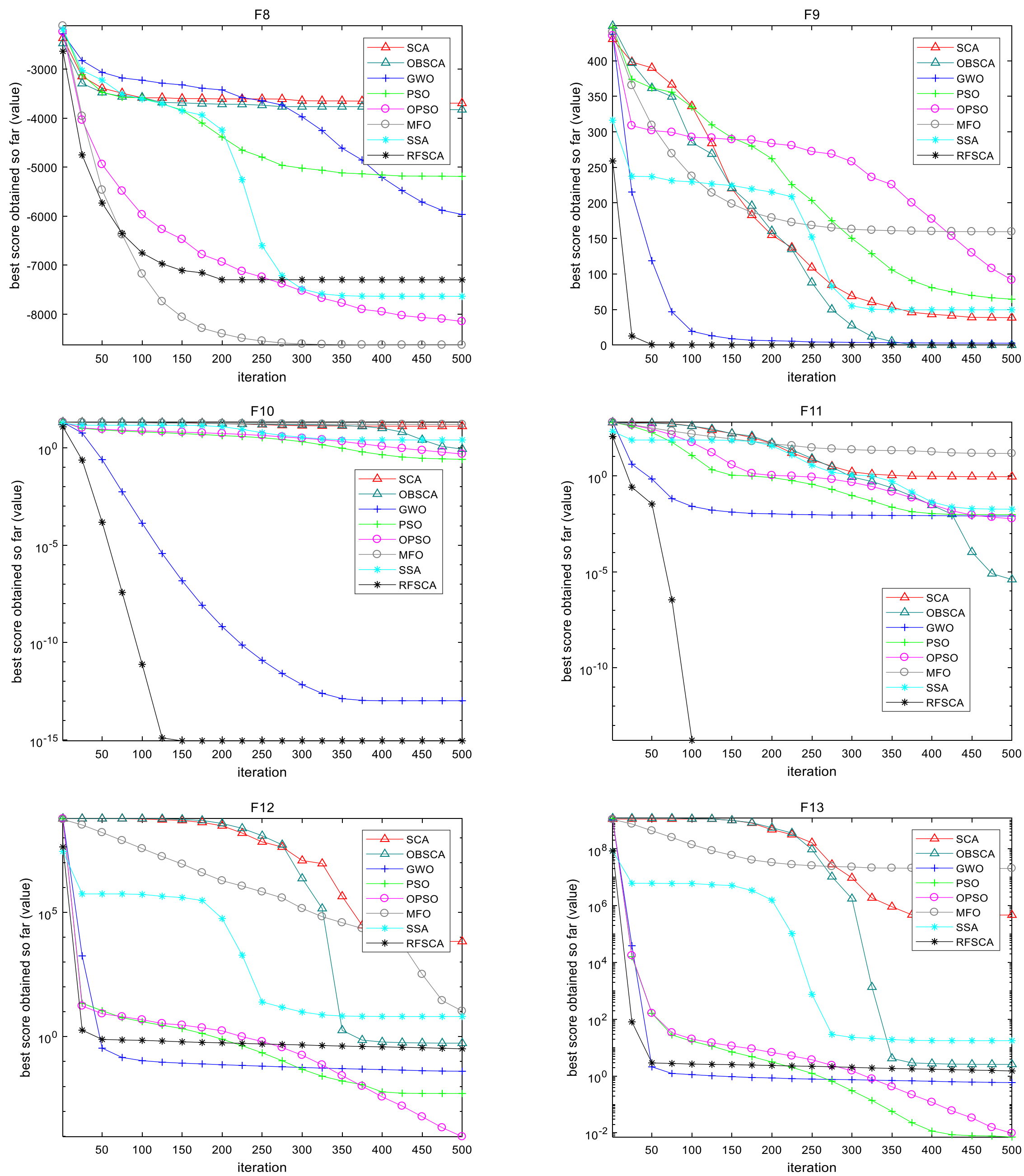

Fig. 6. Convergence graphs for multimodal problems.

OBSCA and RFSCA is 32.1976 s and 34.0926, respectively. So it can be seen that the time complexity of RFSCA is similar to OBSCA.

Section 4.4 gives the experimental results of the IEEE CEC 2017 standard test set, where the number of functional evaluations ($FES = 10000 \cdot Dim$) is used as the algorithm termination condition, and the number of functional evaluations can reflect the algorithm runtime. The last row of Table 6 shows the sum of CPU times of SCA, OBSCA, RFSCA running 20 times in 30 functions, and the CPU times of SCA, OBSCA, RFSCA are 6.0808e+03 s,5.0692e+03 s,5.0473e+03 s. In summary, RFSCA can use less time to calculate better results than SCA, OBSCA.

4.6. Welded beam design

This section tests the ability of RFSCA to solve constrained optimization problems through two engineering instances: Welded beam design [28], pressure vessel design [18,19]. The performance of RFSCA in two problems can reflect the practical application performance of the algorithm. These two engineering problems have different constraints, and the penalty function method is one of the constraint processing techniques which is most commonly used. The basic idea of the penalty function is to add a penalty item that can reflect whether the constraint is satisfied in the objective function, and then use the optimization algorithm to solve the objective function, so that the algorithm finds the optimal solution of the objective function under the action of the penalty term. In this paper, we apply the penalty function in [29] to solve the problem. The population size of the algorithm is set as 30 and the maximum number of iterations is 500 in all experiments. The algorithm runs 20 times to obtain the average value of the objective function.

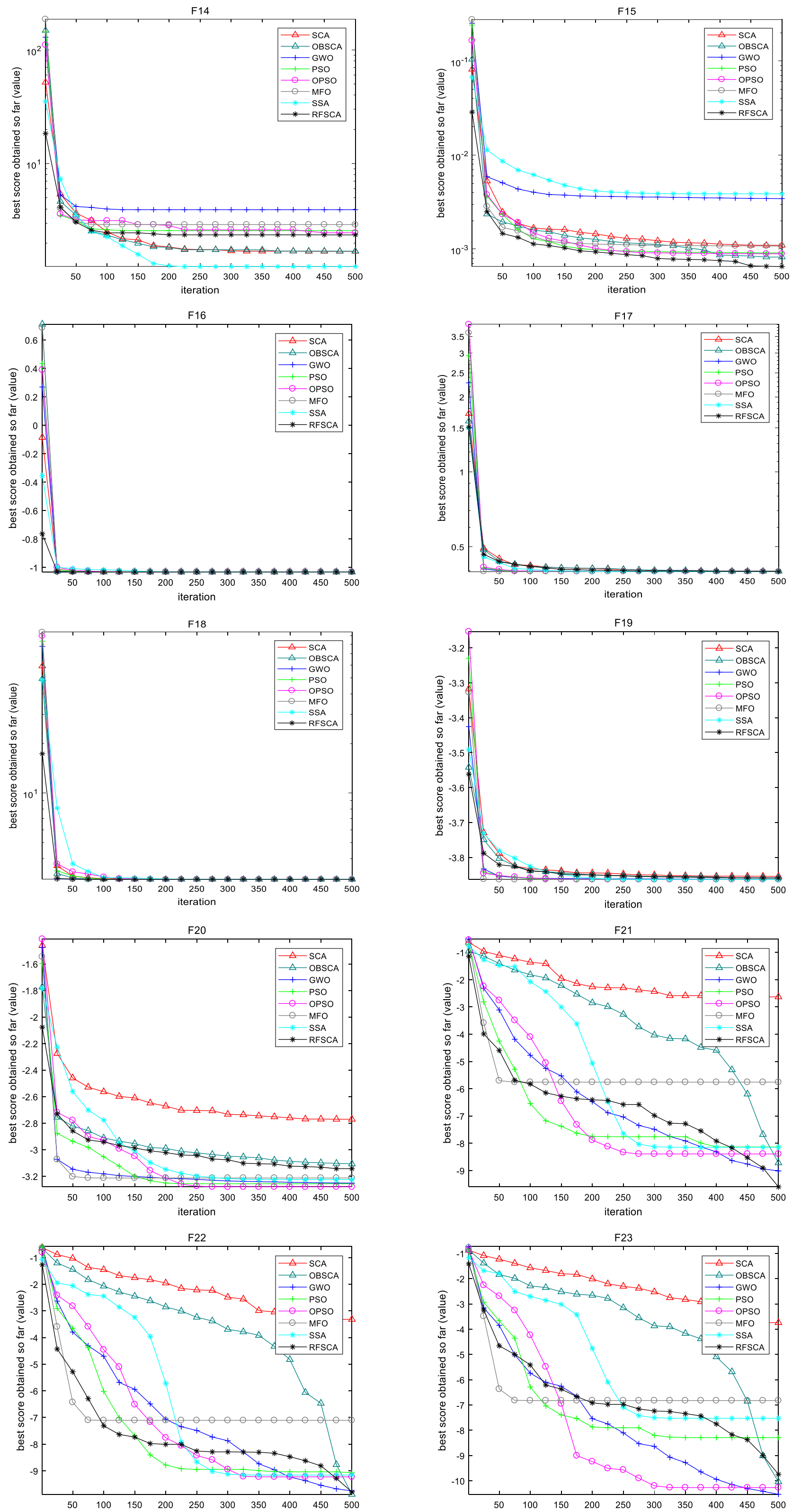

Fig. 7. Convergence graphs for Fixed-dimension multimodal benchmark functions.

Welded beam design problem (as Fig. 11) is to minimize manufacturing costs. There are four consecutive design variables in this question: welding thickness h, weld joint length l, beam width t and beam thickness b.

Let $\boldsymbol{x} = [x_1, x_2, x_3, x_4] = [h, l, t, b]$, and the mathematical description of the problem is as follows.
The objective function is

$$\min f(\boldsymbol{x}) = 1.1047x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2)$$

Table 4
The result of the control parameter α taken on the benchmark function (mean, standard deviation, Friedman-P, rank, $n = 30$).

α	F1(P = 1)			F2(P = 0.2857)			F3(P = 1)			F4(P = 0.3792)		
	Ave	STD	Rank	Ave	STD	Rank	Ave	STD	Rank	Ave	STD	Rank
0	0	0	6	4.4e−207	0	6	0	0	6	1.13e−188	0	6
0.1	0	0	6	8.8e−209	0	6	0	0	6	5.26e−184	0	6
0.2	0	0	6	7.5e−209	0	6	0	0	6	1.22e−176	0	6
0.3	0	0	6	1.8e−215	0	6	0	0	6	3.22e−184	0	6
0.4	0	0	6	8.5e−199	0	6	0	0	6	1.56e−190	0	6
0.5	0	0	6	1.9e−206	0	6	0	0	6	1.22e−189	0	6
0.6	0	0	6	4.4e−211	0	6	0	0	6	1.78e−189	0	6
0.7	0	0	6	2.3e−211	0	6	0	0	6	1.35e−184	0	6
0.8	0	0	6	1.2e−209	0	6	0	0	6	1.32e−184	0	6
0.9	0	0	6	1.2e−209	0	6	0	0	6	1.59e−185	0	6
1	0	0	6	1.1e−210	0	6	0	0	6	1.12e−187	0	6

α	F5(p = 0.0764)			F6(p = 5.3173e−13)			F7(p = 0.7408)			F8(p = 0.0042)		
	Ave	STD	Rank	Ave	STD	Rank	Ave	STD	Rank	Ave	STD	Rank
0	27.9655	0.4278	6	3.1233	0.5099	8	5.70e−05	5.71e−05	6	−6.41e+03	745.6081	11
0.1	27.8290	0.5173	6	2.2869	0.5788	2	5.26e−05	3.98e−05	6	−6.71e+03	1.02e+03	9
0.2	27.7598	0.4594	6	2.1622	0.6143	1	5.42e−05	5.40e−05	6	−6.86e+03	1.27e+03	8
0.3	27.9698	0.4213	6	2.4747	0.3760	3	4.37e−05	3.54e−05	6	−6.92e+03	769.9945	7
0.4	27.7635	0.4504	6	2.6556	0.2031	5	4.58e−05	4.31e−05	6	−6.52e+03	807.6112	10
0.5	27.8772	0.5162	6	2.4786	0.4952	4	4.67e−05	3.80e−05	6	−7.17e+03	959.0727	5
0.6	28.0599	0.5062	6	2.8634	0.3553	7	3.48e−05	3.40e−05	6	−7.31e+03	1.09e+03	3
0.7	28.2131	0.4289	6	2.8063	0.5867	6	4.46e−05	4.12e−05	6	−7.28e+03	894.7391	4
0.8	27.9742	0.4821	6	3.1803	0.5732	10	4.70e−05	3.51e−05	6	−7.06e+03	823.7323	6
0.9	28.0799	0.4858	6	3.1363	0.3813	9	7.44e−05	6.83e−05	6	**−7.59e+03**	**934.7570**	**1**
1	28.1339	0.4224	6	3.1936	0.6535	11	6.21e−05	6.51e−05	6	−7.47e+03	771.0374	2

α	F9(p = 1)			F10(p = 1)			F11(p = 1)			F12(p = 1.5058e−05)		
	Ave	STD	Rank	Ave	STD	Rank	Ave	STD	Rank	Ave	STD	Rank
0	0	0	6	8.88e−16	0	6	0	0	6	0.3370	0.1096	5
0.1	0	0	6	8.88e−16	0	6	0	0	6	0.2656	0.0570	1
0.2	0	0	6	8.88e−16	0	6	0	0	6	0.2818	0.0463	2
0.3	0	0	6	8.88e−16	0	6	0	0	6	0.3234	0.0807	4
0.4	0	0	6	8.88e−16	0	6	0	0	6	0.3169	0.0519	3
0.5	0	0	6	8.88e−16	0	6	0	0	6	0.3410	0.0670	6
0.6	0	0	6	8.88e−16	0	6	0	0	6	0.3418	0.1099	7
0.7	0	0	6	8.88e−16	0	6	0	0	6	0.3740	0.1208	9
0.8	0	0	6	8.88e−16	0	6	0	0	6	0.3691	0.1168	8
0.9	0	0	6	8.88e−16	0	6	0	0	6	0.3765	0.0941	10
1	0	0	6	8.88e−16	0	6	0	0	6	0.3830	0.0810	11

α	F13(p = 9.6151e−09)			F14(p = 0.2701)			F15(p = 0.1399)			F16(p = 0.5382)		
	Ave	STD	Rank	Ave	STD	Rank	Ave	STD	Rank	Ave	STD	Rank
0	2.5436	0.3410	11	3.4456	3.9780	6	4.37e−04	1.23e−04	6	−1.0316	1.86e−05	6
0.1	1.7955	0.3537	6	4.1186	4.7822	6	5.16e−04	1.90e−04	6	−1.0316	2.02e−05	6
0.2	1.5083	0.3881	1	4.1268	4.3629	6	6.01e−04	1.94e−04	6	−1.0316	1.75e−05	6
0.3	1.6148	0.3487	2	6.7607	5.0391	6	4.86e−04	1.37e−04	6	−1.0316	1.09e−05	6
0.4	1.7679	0.2749	5	6.0717	5.0841	6	5.35e−04	2.96e−04	6	−1.0316	2.07e−05	6
0.5	1.6345	0.2941	3	3.8362	4.0599	6	5.89e−04	1.90e−04	6	−1.0316	1.32e−05	6
0.6	1.8224	0.2531	7	5.8810	4.8640	6	5.79e−04	2.47e−04	6	−1.0316	1.20e−05	6
0.7	1.7515	0.2607	4	4.4129	4.8357	6	5.22e−04	1.66e−04	6	−1.0316	2.50e−05	6
0.8	1.8395	0.3043	10	3.2990	3.7853	6	5.21e−04	1.73e−04	6	−1.0316	2.96e−05	6
0.9	1.8287	0.3588	8	6.5522	5.4042	6	5.78e−04	2.18e−04	6	−1.0316	1.86e−05	6
1	1.8300	0.2732	9	4.0764	4.1647	6	5.42e−04	2.78e−04	6	−1.0316	1.10e−05	6

α	F17(p = 2.7525e−07)			F18(p = 0.0019)			F19(p = 0.3740)			F20(p = 0.0122)		
	Ave	STD	Rank	Ave	STD	Rank	Ave	STD	Rank	Ave	STD	Rank
0	0.3987	0.0011	6	3.0000	2.43e−05	6	−3.8594	0.0019	6	−3.1567	0.0633	2
0.1	0.3988	9.08e−04	8	3.0000	4.67e−05	6	−3.8602	0.0018	6	−3.1484	0.0539	4
0.2	0.3988	0.0010	8	3.0000	6.05e−06	6	−3.8590	0.0026	6	−3.1853	0.0446	1
0.3	0.3985	6.48e−04	3	3.0000	1.25e−05	6	−3.8593	0.0023	6	−3.1530	0.0512	3
0.4	0.3989	0.0011	10	3.0000	6.11e−06	6	−3.8580	0.0030	6	−3.1316	0.0621	9
0.5	0.3985	4.95e−04	3	3.0000	3.69e−06	6	−3.8598	0.0018	6	−3.1441	0.0541	5
0.6	0.3988	0.0011	8	3.0000	2.35e−05	6	−3.8589	0.0027	6	−3.1387	0.0464	7
0.7	0.3986	9.66e−04	5	3.0000	6.71e−06	6	−3.8590	0.0030	6	−3.1393	0.0578	6
0.8	0.3985	5.44e−04	3	3.0000	1.59e−05	6	−3.8597	0.0023	6	−3.1293	0.0432	10
0.9	0.3989	0.0012	11	3.0000	1.75e−05	6	−3.8583	0.0032	6	−3.1341	0.0431	8
1	**0.3979**	**2.36e−05**	**1**	3.0000	2.50e−05	6	−3.8587	0.0021	6	−3.1149	0.0613	11

(*continued on next page*)

Table 4 (*continued*).

α	F21(p = 0.6368)			F22(p = 0.1898)			F23(p = 0.1278)			Total rank
	Ave	STD	Rank	Ave	STD	Rank	Ave	STD	Rank	
0	−8.7928	1.0088	6	−8.7308	1.1228	6	−9.2730	**0.8361**	6	145
0.1	−8.8379	0.7412	6	−8.4041	1.0372	6	−8.5102	0.8606	6	132
0.2	−8.7790	0.8402	6	−8.7753	0.8192	6	−9.0773	1.1141	6	**123**
0.3	−8.6430	0.9305	6	−8.8948	0.9607	6	−8.8518	0.9258	6	124
0.4	−8.4131	0.8949	6	−9.0763	0.7988	6	−8.6049	1.0977	6	144
0.5	−8.4723	0.7307	6	−9.0132	0.8797	6	−8.5353	0.9565	6	128
0.6	−8.5162	0.9304	6	−8.7834	1.2256	6	−8.5986	1.0924	6	141
0.7	−8.4461	0.9803	6	−9.0577	0.8499	6	−8.9145	1.1112	6	136
0.8	−8.6816	0.8473	6	−8.2817	1.0365	6	−8.4102	0.9184	6	149
0.9	−8.0694	1.3100	6	−8.5910	1.1203	6	−8.5425	1.0180	6	149
1	−8.2050	1.3928	6	−8.4671	1.1518	6	−8.7049	1.2393	6	147

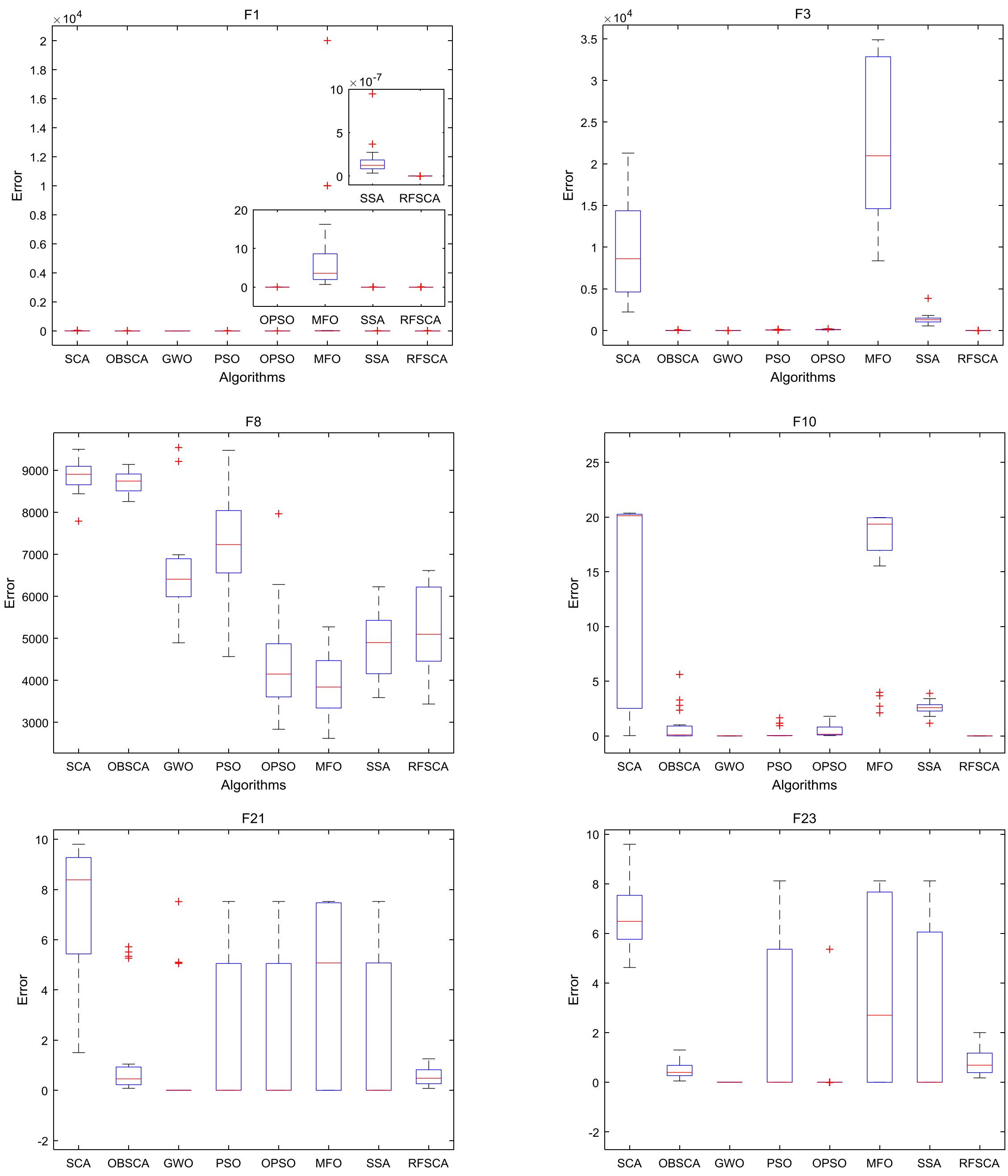

Fig. 8. Box plot for functions (F1, F3, F8, F10, F21, F23).

Table 5
The results of the parameter p taken on the benchmark function (mean, standard deviation, Friedman-P, rank, $n = 30$).

p	F1(p = 5.2885E−37)			F2(p = 3.1334E−37)			F3(p = 3.4369E−37)			F4(p = 3.1334E−37)		
	Ave	STD	Rank	Ave	STD	Rank	Ave	STD	Rank	Ave	STD	Rank
0	0	0	1	4.20e−209	0	1	0	0	1	3.53e−188	0	1
0.1	0	0	2	6.51e−188	0	2	5.10e−314	0	2	7.35e−162	3.20e−161	2
0.2	3.78e−315	0	3	3.12e−165	0	3	2.88e−265	0	3	2.33e−147	1.01e−146	3
0.3	3.79e−251	0	4	2.07e−145	1.58e−123	4	1.83e−221	0	4	1.80e−130	7.60e−130	4
0.4	3.03e−206	0	5	3.70e−124	1.18e−101	5	9.41e−199	0	5	1.22e−109	5.21e−109	5
0.5	8.39e−193	0	6	2.72e−102	2.42e−80	6	2.36e−155	1.02e−154	6	5.88e−86	2.40e−85	6
0.6	6.42e−147	2.71e−146	7	5.56e−81	3.84e−62	7	2.91e−99	1.27e−98	7	1.08e−69	4.53e−69	7
0.7	1.51e−115	6.12e−115	8	8.92e−63	3.12e−40	8	1.28e−77	5.59e−77	8	2.02e−51	6.15e−51	8
0.8	3.23e−76	1.40e−75	9	7.16e−41	5.63e−28	9	1.50e−43	6.57e−43	9	2.60e−30	1.04e−29	9
0.9	4.42e−38	1.92e−37	10	1.55e−28	5.15e−12	10	5.34e−21	1.38e−20	10	9.48e−15	4.12e−14	10
1	1.96e−10	4.78e−10	11	2.25e−12	9.04e−145	11	25.3840	46.8074	11	0.2122	0.5215	11
p	F5(p = 3.6569e−04)			F6(p = 0.0028)			F7(p = 2.2392e−10)			F8(p = 1.2464e−06)		
	Ave	STD	Rank	Ave	STD	Rank	Ave	STD	Rank	Ave	STD	Rank
0	27.8982	0.6084	8	2.1670	0.5301	1	6.46e−05	7.35e−05	5	−7.00e+03	872.205	6
0.1	27.9926	0.4304	10	2.2848	0.6225	3	5.80e−05	5.65e−05	2	−6.97e+03	778.752	8
0.2	27.8652	0.4592	5	2.3365	0.4368	5	5.49e−05	3.42e−05	1	−7.32e+03	995.620	2
0.3	27.7493	0.4878	2	2.2897	0.8274	4	6.06e−05	4.54e−05	4	−7.24e+03	947.762	4
0.4	27.8244	0.5234	4	2.5635	0.4378	10	7.79e−05	7.49e−05	6	−7.18e+03	977.437	5
0.5	27.9540	0.4381	9	2.1735	0.5918	2	5.87e−05	3.73e−05	3	−7.61e+03	995.569	1
0.6	27.6370	0.5050	1	2.4225	0.5095	6	1.11e−04	1.03e−04	8	−7.31e+03	1.09e+0	3
0.7	27.7968	0.4494	3	2.4815	0.5051	7	1.03e−04	6.30e−05	7	−6.86e+03	1.15e+0	9
0.8	27.8882	0.4563	7	2.5300	0.4770	9	1.77e−04	1.49e−04	10	−6.98e+03	1.53e+0	7
0.9	27.8877	0.4225	6	2.4903	0.5702	8	1.35e−04	1.04e−04	9	−6.50e+03	1.27e+0	10
1	28.5583	0.1697	11	3.0948	0.7508	11	0.0037	0.0026	11	−5.45e+03	381.673	11
p	F9(p = 2.3430e−33)			F10(p = 1.6139e−37)			F11(p = 1.6139e−37)			F12(p = 8.2300e−06)		
	Ave	STD	Rank	Ave	STD	Rank	Ave	STD	Rank	Ave	STD	Rank
0	0	0	5.5	8.881e−16	0	5.5	0	0	5.5	0.2433	0.0704	1
0.1	0	0	5.5	8.881e−16	0	5.5	0	0	5.5	0.3263	0.1223	4
0.2	0	0	5.5	8.881e−16	0	5.5	0	0	5.5	0.3120	0.1565	3
0.3	0	0	5.5	8.881e−16	0	5.5	0	0	5.5	0.3443	0.1056	6
0.4	0	0	5.5	8.881e−16	0	5.5	0	0	5.5	0.2915	0.1103	2
0.5	0	0	5.5	8.881e−16	0	5.5	0	0	5.5	0.3319	0.1604	5
0.6	0	0	5.5	8.881e−16	0	5.5	0	0	5.5	0.3488	0.1418	9
0.7	0	0	5.5	8.881e−16	0	5.5	0	0	5.5	0.3724	0.1724	10
0.8	0	0	5.5	8.881e−16	0	5.5	0	0	5.5	0.3463	0.0733	8
0.9	0	0	5.5	8.881e−16	0	5.5	0	0	5.5	0.3450	0.0720	**7**
1	0.3309	1.4423	11	0.3525	1.5365	11	7.124e−05	1.98e−04	11	0.4581	0.1003	11
p	F13(p = 1.4897E−06)			F14(p = 0.4054)			F15(p = 1.8070E−06)			F16(p = 0.0021)		
	Ave	STD	Rank	Ave	STD	Rank	Ave	STD	Rank	Ave	STD	Rank
0	1.5918	0.3606	7	6.5483	5.5764	6	4.76e−04	1.69e−04	1	−1.0316	2.98e−05	6
0.1	1.7709	0.3212	10	4.1262	4.3633	6	5.13e−04	2.52e−04	2	−1.0316	1.77e−05	6
0.2	1.6148	0.2645	9	4.7988	5.1980	6	5.38e−04	1.87e−04	3	−1.0316	1.20e−05	6
0.3	1.6113	0.5029	8	5.1140	4.8329	6	6.28e−04	2.63e−04	5	−1.0316	6.96e−06	6
0.4	1.4607	0.3202	5	2.7802	2.8173	6	5.71e−04	1.88e−04	4	−1.0316	9.96e−06	6
0.5	1.3587	0.3659	1	3.2531	3.5554	6	6.58e−04	2.88e−04	6	−1.0316	1.43e−05	6
0.6	1.4167	0.3335	3	5.7822	5.1249	6	6.78e−04	2.43e−04	8	−1.0316	8.18e−06	6
0.7	1.5135	0.2557	6	2.7616	3.4199	6	6.74e−04	2.87e−04	7	−1.0316	9.35e−06	6
0.8	1.4098	0.3215	2	3.1474	4.0749	6	7.92e−04	2.00e−04	11	−1.0316	3.81e−06	6
0.9	1.4189	0.3580	4	3.5328	4.6063	6	7.64e−04	2.01e−04	9	−1.0316	3.42e−06	6
1	2.1833	0.3260	11	2.6543	3.9559	6	7.75e−04	1.93e−04	10	−1.0316	5.57e−06	6
p	F17(p = 0.1063)			F18(p = 0.8412)			F19(p = 0.6951)			F20(p = 0.7551)		
	Ave	STD	Rank	Ave	STD	Rank	Ave	STD	Rank	Ave	STD	Rank
0	0.3986	7.77e−04	6	3.0000	1.62e−05	6	−3.8586	0.0040	6	−3.1458	0.0526	6
0.1	0.3991	0.0014	6	3.0000	1.87e−05	6	−3.8588	0.0026	6	−3.1507	0.0594	6
0.2	0.3986	5.45e−04	6	3.0000	9.07e−06	6	−3.8600	0.0024	6	−3.1486	0.0470	6
0.3	0.3988	0.0013	6	3.0000	1.18e−05	6	−3.8598	0.0025	6	−3.1662	0.0543	6
0.4	0.3985	5.98e−04	6	3.0000	1.93e−05	6	−3.8595	0.0019	6	−3.1439	0.0364	6
0.5	0.3983	4.67e−04	6	3.0000	1.56e−05	6	−3.8589	0.0028	6	−3.1639	0.0438	6
0.6	0.3986	5.37e−04	6	3.0000	1.82e−05	6	−3.8586	0.0031	6	−3.1536	0.0543	6
0.7	0.3986	6.62e−04	6	3.0000	1.07e−05	6	−3.8591	0.0031	6	−3.1590	0.0497	6
0.8	0.3984	6.05e−04	6	3.0000	2.17e−05	6	−3.8586	0.0025	6	−3.1587	0.0502	6
0.9	0.3992	0.0014	6	3.0000	3.86e−05	6	−3.8589	0.0033	6	−3.1352	0.0310	6
1	0.3992	0.0011	6	3.0000	3.11e−05	6	−3.8596	0.0020	6	−3.1400	0.0334	6

(*continued on next page*)

Table 5 (*continued*).

p	F21(p = 2.5747e−04)			F22(p = 8.4551e−07)			F23(p = 1.2333e−05)			Total rank
	Ave	STD	Rank	Ave	STD	Rank	Ave	STD	Rank	
0	−8.6162	1.1218	11	−8.6889	1.0888	11	−8.6592	1.2678	11	118.5
0.1	−9.0185	0.8009	10	−9.2094	0.8111	10	−9.5513	0.5668	9	129.5
0.2	−9.4169	0.5977	8	−9.4215	0.7519	8	−9.3848	0.5826	10	116.5
0.3	−9.2619	0.5460	9	−9.4165	0.5789	9	−9.6492	0.6401	8	128.5
0.4	−9.4315	0.5177	7	−9.6386	0.5653	7	−9.7439	0.6621	7	129.5
0.5	−9.7037	0.3893	2	−9.8942	0.2949	5	−9.8815	0.4191	6	**109.5**
0.6	−9.4832	0.4251	6	−9.6859	0.3949	6	−9.9419	0.3531	5	109.5
0.7	−9.6094	0.3807	4	−9.9873	0.2515	1	−10.1377	0.3245	1	145.5
0.8	−9.6498	0.2801	3	−9.9212	0.3397	4	−10.0098	0.3870	3	151.5
0.9	−9.7431	0.2949	1	−9.9576	0.3066	2	−10.0504	0.3434	2	148.5
1	−9.5220	1.1480	5	−9.9346	0.1706	3	−9.9872	0.3120	4	204

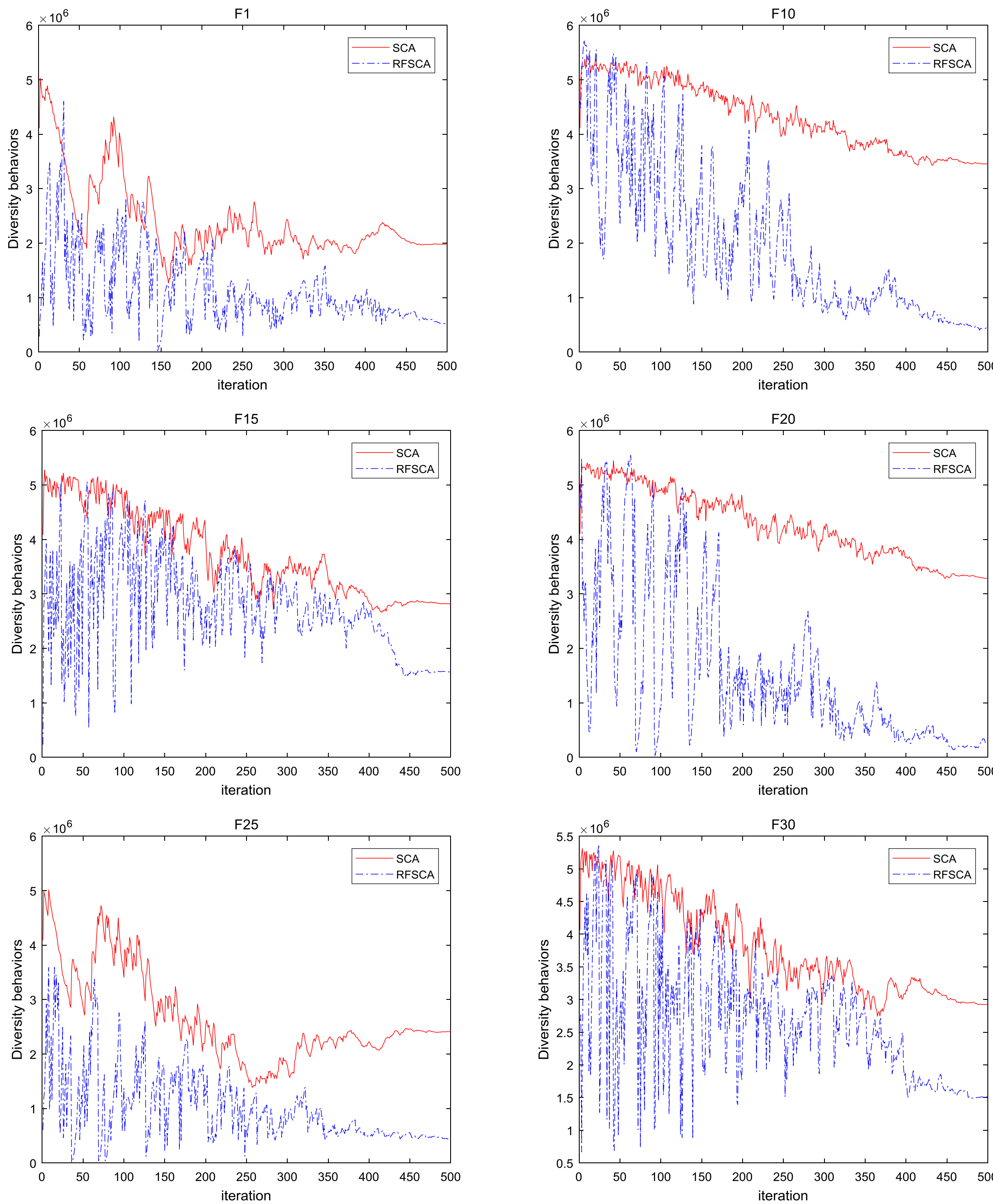

Fig. 9. Diversity behavior for functions (F1, F10, F15, F20, F25, F30).

Table 6
The statistical comparison proposed method and some swarm intelligence algorithms in standard IEEE CEC 2017.

Functions		Algorithm							
		SCA	OBSCA	GWO	PSO	OPSO	SSA	MFO	RFSCA
F1	Ave	7.1865e+10	9.0695e+10	9.3144e+09	6.7224e+06	1.6148e+08	8.5441e+03	3.9203e+10	7.4840e+10
	STD	8.5017e+09	9.7471e+09	3.6136e+09	3.0018e+07	6.4934e+07	1.0345e+04	2.1383e+10	7.6484e+09
F2	Ave	2.8749e+77	1.0693e+82	1.4386e+51	5.6296e+27	6.1801e+28	1.1911e+06	4.2048e+72	1.5132e+80
	STD	6.4942e+77	4.6685e+82	3.4132e+51	2.5176e+28	9.5355e+28	3.0709e+06	1.4909e+73	5.2227e+80
F3	Ave	6.6887e+08	1.7532e+07	7.6534e+04	5.4563	2.8749e+03	1.5276e−07	2.0053e+05	3.7773e+06
	STD	1.8237e+09	2.9966e+07	1.1883e+04	4.3623	1.1171e+03	3.5020e−08	1.0990e+05	8.8163e+06
F4	Ave	1.7921e+04	2.4979e+04	896.5962	155.2622	258.7275	198.5172	4.0918e+03	1.9380e+04
	STD	4.8497e+03	4.7272e+03	428.1702	57.7847	45.7992	38.2625	2.2371e+03	3.2752e+03
F5	Ave	711.4990	591.3619	221.4273	263.2647	356.1589	269.0431	455.9413	617.0501
	STD	44.7997	20.3877	35.5283	35.1935	37.1614	48.7124	69.9795	14.9112
F6	Ave	94.0523	94.4079	16.3465	45.3093	74.4817	44.6200	55.7683	96.6642
	STD	6.8745	3.4446	4.6633	6.0346	9.6730	9.5049	11.0270	4.3058
F7	Ave	1.5380e+03	1.2982e+03	328.5100	242.8791	509.3825	345.1304	1.0650e+03	1.1804e+03
	STD	159.4168	54.1741	51.6524	33.5793	45.5237	74.7048	454.4803	54.5719
F8	Ave	696.9358	657.5234	219.0074	274.3585	366.1817	258.7872	439.1819	657.3519
	STD	41.0094	20.3943	45.5867	43.1880	48.8620	56.2548	69.1237	14.7742
F9	Ave	3.7282e+04	3.9555e+04	5.8019e+03	8.7062e+03	2.1103e+04	9.1257e+03	1.4675e+04	3.5656e+04
	STD	6.6660e+03	3.7318e+03	3.8060e+03	1.3823e+03	2.0284e+03	3.9943e+03	3.0586e+03	3.3024e+03
F10	Ave	1.6097e+04	1.1117e+04	6.0483e+03	6.1480e+03	7.9163e+03	8.2728e+03	7.8980e+03	1.1634e+04
	STD	724.4347	425.9723	698.8297	871.3161	1.3903e+03	3.0110e+03	978.1511	497.4955
F11	Ave	2.9427e+04	2.5667e+04	4.2232e+03	144.1353	417.0242	279.9908	8.2199e+03	1.9781e+04
	STD	1.4393e+04	5.8796e+03	2.1935e+03	33.9505	60.5519	55.2791	6.3508e+03	1.8142e+03
F12	Ave	2.9866e+10	3.0224e+10	1.0747e+09	1.2515e+07	1.1626e+08	1.0472e+07	6.4009e+09	3.6143e+10
	STD	6.5831e+09	7.4115e+09	1.9115e+09	5.2095e+07	5.2525e+07	5.8445e+06	4.5101e+09	6.5407e+09
F13	Ave	1.3055e+10	1.3740e+10	1.2453e+08	5.3510e+03	1.1013e+07	1.3519e+05	6.2357e+08	1.4915e+10
	STD	4.0080e+09	3.5254e+09	1.1640e+08	5.2045e+03	7.8594e+06	7.1043e+04	1.0145e+09	4.8024e+09
F14	Ave	5.0592e+07	6.0261e+07	6.9681e+05	3.9478e+04	1.0427e+05	6.7153e+05	6.9706e+05	4.8398e+07
	STD	2.6935e+07	3.5678e+07	9.3096e+05	2.8741e+04	5.8703e+04	2.8148e+06	8.5539e+05	1.8985e+07
F15	Ave	3.3026e+09	3.2351e+09	1.9074e+07	7.9092e+03	2.7331e+06	3.7655e+04	3.1706e+07	3.0410e+09
	STD	1.0438e+09	1.1573e+09	2.5862e+07	6.9705e+03	2.7392e+06	2.1837e+04	9.4626e+07	9.2698e+08
F16	Ave	6.1561e+03	6.0312e+03	1.6183e+03	1.7064e+03	1.9317e+03	1.8577e+03	2.6781e+03	6.1948e+03
	STD	512.4725	500.3005	351.4790	520.4002	490.8616	574.9992	415.3409	678.6378
F17	Ave	4.5783e+03	4.5956e+03	973.0288	1.3433e+03	1.1793e+03	1.3304e+03	2.1882e+03	4.7858e+03
	STD	1.1197e+03	590.2002	227.0177	281.9043	250.3352	472.7411	443.5808	958.0007
F18	Ave	3.0599e+08	1.7708e+08	2.9589e+06	2.0751e+05	9.4965e+05	2.5770e+05	8.9527e+06	1.4264e+08
	STD	1.7969e+08	9.1053e+07	2.3298e+06	1.2249e+05	5.3210e+05	1.4500e+05	1.5686e+07	6.8400e+07
F19	Ave	1.6641e+09	2.1194e+09	2.2529e+06	1.2977e+04	1.3498e+06	6.1197e+05	1.2468e+07	1.7903e+09
	STD	7.6501e+08	7.5462e+08	6.3283e+06	9.4921e+03	1.4151e+06	2.5556e+05	3.9794e+07	5.0769e+08
F20	Ave	2.7580e+03	1.8296e+03	789.7615	1.1303e+03	1.1504e+03	1.0949e+03	1.5422e+03	2.9724e+03
	STD	405.6047	147.1946	296.7960	206.5977	221.3535	356.4621	453.4683	445.6113
F21	Ave	926.5924	895.0859	415.3762	547.0316	562.4932	426.4403	641.6914	853.9346
	STD	49.1621	103.1074	33.4888	49.1373	84.9341	39.6424	69.0967	109.5580
F22	Ave	1.6370e+04	7.7278e+03	6.7012e+03	6.8337e+03	8.3060e+03	9.2480e+03	8.0804e+03	7.9531e+03
	STD	744.0607	955.3102	931.8424	893.7672	2.8914e+03	4.8428e+03	858.2767	704.6667
F23	Ave	1.7025e+03	1.6953e+03	721.0690	1.2169e+03	1.2558e+03	675.0804	869.6087	1.6259e+03
	STD	131.6014	94.8857	113.6030	170.0205	152.5121	51.3039	77.8164	130.6163
F24	Ave	1.8478e+03	1.8516e+03	863.1938	1.2009e+03	1.1697e+03	749.3175	808.9487	1.8663e+03
	STD	129.6578	109.1144	123.8392	175.6401	191.7634	108.5309	46.1625	157.6832
F25	Ave	9.4063e+03	1.0725e+04	1.2044e+03	565.8681	617.5086	546.9053	2.8216e+03	8.6215e+03
	STD	1.4732e+03	1.3260e+03	302.1256	37.9801	32.5347	38.6364	1.9849e+03	1.0277e+03
F26	Ave	1.3185e+04	8.2214e+03	4.0309e+03	6.2700e+03	1.5526e+03	2.8205e+03	5.7124e+03	1.1565e+04
	STD	803.1621	781.2141	606.4577	2.5218e+03	2.7729e+03	2.1051e+03	781.4916	490.6770
F27	Ave	3.2510e+03	3.6426e+03	1.0608e+03	1.3048e+03	1.3105e+03	979.2142	864.6682	3.2704e+03
	STD	563.7377	410.7277	86.5221	373.6430	269.1831	126.9422	86.7233	369.9304
F28	Ave	6.8585e+03	8.2021e+03	1.6542e+03	534.2461	551.1416	1.3547e+03	5.4649e+03	7.1223e+03
	STD	817.0022	1.1044e+03	518.4023	34.4643	44.6155	328.5891	994.9079	938.4836
F29	Ave	1.0699e+04	1.4729e+04	1.7212e+03	2.0465e+03	2.5500e+03	1.8571e+03	2.5318e+03	1.5163e+04
	STD	4.8051e+03	7.4566e+03	367.5296	320.1930	472.7169	242.0193	519.1372	6.3270e+03
F30	Ave	3.1732e+09	3.4218e+09	9.3511e+07	8.2298e+05	7.0002e+07	2.3591e+07	9.4440e+07	2.6091e+09
	STD	6.3872e+08	1.1142e+09	3.3320e+07	1.4547e+05	2.4039e+07	5.0572e+06	3.4273e+08	7.8233e+08
CPUtime		6.0808e+03	5.0692e+03						5.0473e+03

Table 7
Statistical decision based on Wilcoxon signed rank test at 5% level of significance in standard IEEE CEC 2017.

Function	SCA		OBSCA		Function	SCA		OBSCA	
	p-value	Decision	*p-value*	Decision		*p-value*	Decision	*p-value*	Decision
F1	0.4407	=	1.5997e−05	−	F16	0.7557	=	0.2977	=
F2	0.8604	=	0.0040	−	F17	0.5075	=	0.8817	=
F3	6.2200e−04	−	5.0907e−04	−	F18	5.0907e−04	−	0.3942	=
F4	0.0565	=	3.0480e−04	−	F19	0.2287	=	0.1478	=
F5	6.9166e−07	−	3.0480e−04	+	F20	0.0962	=	6.7956e−08	+
F6	0.2085	=	0.1404	=	F21	0.0207	−	0.1136	=
F7	1.4309e−07	−	5.1658e−06	−	F22	6.7956e−08	−	0.2733	=
F8	0.0013	−	0.7557	=	F23	0.0531	=	0.0256	−
F9	0.5075	=	0.0028	−	F24	0.7150	=	0.8604	=
F10	6.7956e−08	−	0.0011	+	F25	0.0720	=	2.0407e−05	−
F11	0.0016	−	2.7451e−04	−	F26	2.9598e−07	−	6.7956e−08	+
F12	0.0047	+	0.0090	+	F27	0.9461	=	0.0060	−
F13	0.2085	=	0.5979	=	F28	0.3648	=	0.0043	−
F14	0.9246	=	0.2733	=	F29	0.0060	+	0.6359	=
F15	0.4570	=	0.6554	=	F30	0.0223	−	0.0193	−
+/= /−						2/17/11		5/13/12	

7 constraints are

$$g_1(\boldsymbol{x}) = \tau(\boldsymbol{x}) - \tau_{\max} \le 0,$$

$$g_2(\boldsymbol{x}) = \sigma(\boldsymbol{x}) - \sigma_{\max} \le 0,$$

$$g_3(\boldsymbol{x}) = \delta(\boldsymbol{x}) - \delta_{\max} \le 0,$$

$$g_4(\boldsymbol{x}) = x_1 - x_4 \le 0,$$

$$g_5(\boldsymbol{x}) = P - Pc(\boldsymbol{x}) \le 0,$$

$$g_6(\boldsymbol{x}) = 0.125 - x_1 \le 0,$$

$$g_7(\boldsymbol{x}) = 1.1047x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \le 0$$

Variables range are

$$0.1 \le x_1 \le 2,$$
$$0.1 \le x_2 \le 10,$$
$$0.1 \le x_3 \le 10,$$
$$0.1 \le x_4 \le 2$$

where

$$\tau(\boldsymbol{x}) = \sqrt{(\tau')^2 + \tau'\tau''\frac{x_2}{R} + (\tau'')^2},$$

$$\tau' = \frac{P}{\sqrt{2}x_1x_2}, \tau'' = \frac{MR}{J}, M = P\left(L + \frac{x_2}{2}\right),$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2},$$

$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\},$$

$$\sigma(\boldsymbol{x}) = \frac{6PL}{x_3^2x_4}, \delta(\vec{\boldsymbol{x}}) = \frac{6PL^3}{Ex_3^2x_4},$$

$$Pc(\boldsymbol{x}) = \frac{4.013\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right),$$

$$P = 6000 \text{ lb}, L = 14 \text{ in.}, \delta_{\max} = 0.25 \text{ in},$$

$$E = 30 \times 10^6 \text{ psi}, G = 12 \times 10^6 \text{ psi},$$

$$\tau_{\max} = 13{,}600 \text{ psi}, \sigma_{\max} = 30{,}000 \text{ psi}$$

Table 8
The optimization results of RFSCA and other algorithms for solving the welded beam design problem.

Algorithm	Optimal values for variables				Optimal cost
	h	*l*	*t*	*b*	
RFSCA	**0.212258**	**1.760352**	**9.415076**	**0.212258**	**1.6029**
OBSCA	0.230824	3.069152	8.988479	0.208795	1.722315
WOA	0.205396	3.484293	9.037426	0.206276	1.730499
RO	0.203687	3.528467	9.004233	0.207241	1.735344
MVO	0.205463	3.473193	9.044502	0.205695	1.72645
CPSO	0.202369	3.544214	9.04821	0.205723	1.72802
GSA	0.182129	3.856979	10	0.202376	1.87995
GA(1991)	0.2489	6.173	8.1789	0.2533	2.43
HS	0.2442	6.2231	8.2915	0.24	2.3807
SIMPLEX	0.2792	5.6256	7.7512	0.2796	2.5307
DAVID	0.2434	6.2552	8.2915	0.2444	2.3841
APPROX	0.2444	6.2189	8.2915	0.2444	2.3815

where the E is the Young's modulus of the bar, G is the shear modulus of the bar, P is the loading condition, and L is the overhang length of the beam.

In this section, we apply RFSCA to solve the problem of structural design of welded beam, and compares the results with other algorithms which include the Opposition-Based Sine Cosine Algorithm (OBSCA) [2] (MA Elaziz et al. 2017), Coevolutionary Particle Swarm Optimization (CPSO) [30] (Krohling & dos Santos Coelho, 2006), Harmony Search (HS) [31] (Lee & Geem, 2005), Genetic Algorithm (GA) [32] (Deb, 1991), Gravitational Search Algorithm (GSA) [33] (Rashedi, Nezamabadi-Pour, & Saryazdi, 2009), Whale Optimization Algorithm (WOA) [34] (Mirjalili & Lewis, 2016b), Griffith and Stewart's successive linear approximation (APPROX) [35] (Ragsdell & Phillips, 1976) and simplex method (SIMPLEX) [35] (Ragsdell & Phillips, 1976), Davidon–Fletcher–Powell (DAVID) [35] (Ragsdell & Phillips, 1976), MVO [36] (Mirjalili et al. 2016) and Ray Optimization (RO) [37] (Kaveh & Khayatazad, 2012). The comparison results are shown in Table 8.

From Table 8, it can be seen that RFSCA has the lowest optimal cost and best performance among all algorithms.

4.7. Pressure vessel design problem

Fig. 12 shows the design of the pressure piping. There are 4 parameters in this design problem: Head thickness T_h, shell thickness T_s, pipe length L, pipe inner radius R.

Let $\boldsymbol{x} = [x_1, x_2, x_3, x_4] = [T_h, T_s, L, R]$, the mathematical description of the problem is as follows.

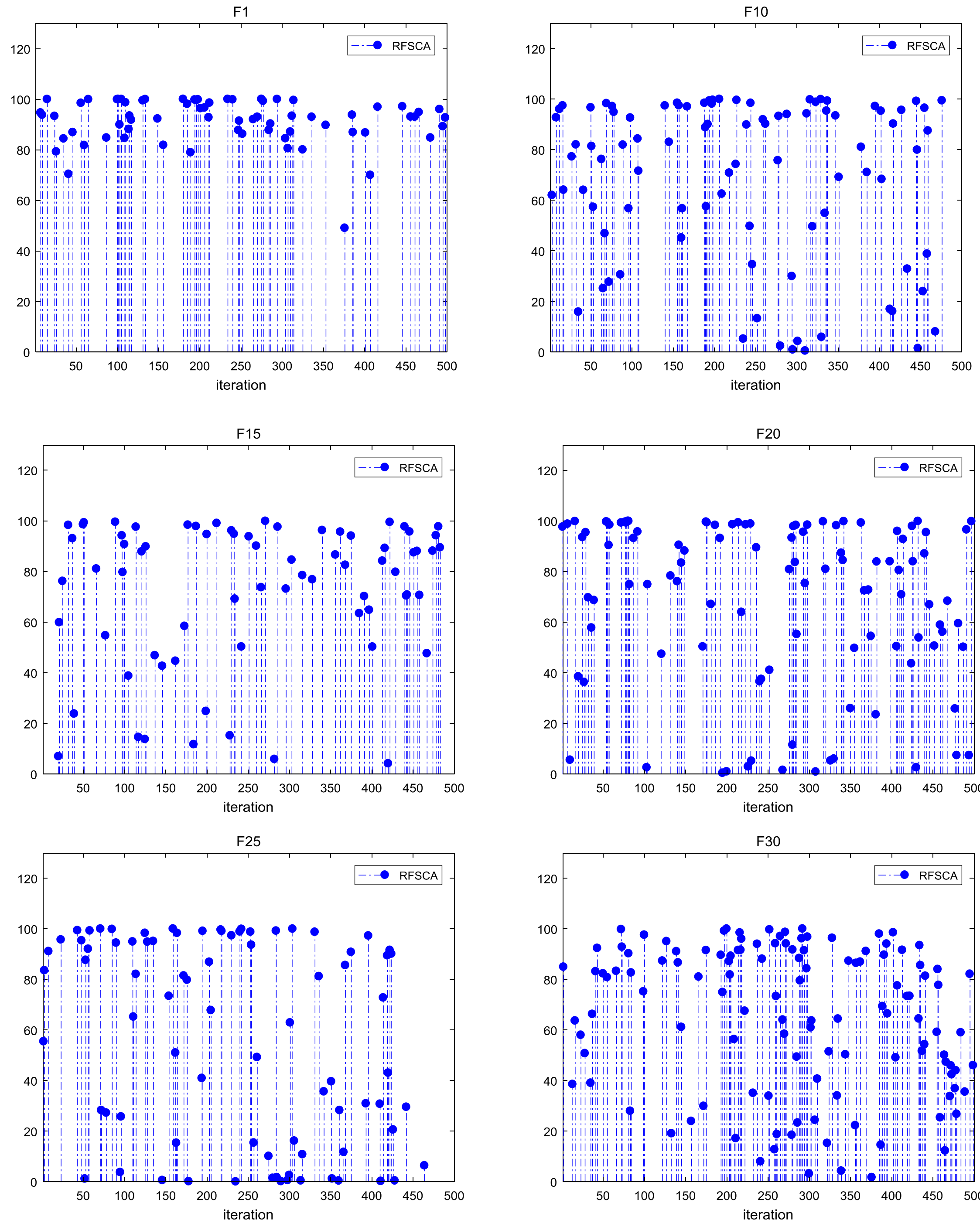

Fig. 10. Euclidean distance of RF strategy (F1, F10, F15, F20, F25, F30).

The objective function is

$$\min f(\boldsymbol{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.4661x_1^2x_4 + 19.84x_1^2x_3$$

4 constraints are

$$g_1(\boldsymbol{x}) = -x_1 - 0.0193x_3 \leq 0,$$

$$g_2(\boldsymbol{x}) = -x_3 + 0.00954x_3 \leq 0,$$

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

$$g_4(\boldsymbol{x}) = x_4 - 240 \leq 0$$

Variables range are

$0 \leq x_1 \leq 99,$
$0 \leq x_2 \leq 99,$
$10 \leq x_3 \leq 200,$
$10 \leq x_4 \leq 200$

The RFSCA is applied for solving this problem, which finds the minimum value of $f(\boldsymbol{x})$ under the constraints. We compare it with thirteen optimization algorithms which are OB-SCA [2] (MA Elaziz et al. 2017); CPSO [30] (Krohling & dos Santos Coelho, 2006); GA [32] (Deb,1991) GA [38] (Coello,2002); GSA [33] (Rashedi, Nezamabadi-Pour, & Saryazdi, 2009); WOA (Mirjalili & Lewis, 2016b); MVO [36] (MA Elaziz et al. 2016);

Table 9
The optimization results of RFSCA and other literature for pressure vessel design problem.

Algorithm	Optimal values for variables				Optimal cost
	T_s	T_h	R	L	
RFSCA	**0.06827**	**0.0625**	**56.58046**	**128.9269**	**4982.57106**
OBSCA	1.2500	0.0625	59.1593	70.8437	5833.9892
WOA	0.812500	0.437500	42.0982699	176.638998	6059.7410
MVO	0.8125	0.4375	42.090738	176.73869	6060.8066
PSO-SCA	0.8125	0.4375	42.098446	176.6366	6059.71433
HPSO	0.8125	0.4375	42.0984	176.6366	6059.7143
ACO	0.812500	0.437500	42.098353	176.637751	6059.7258
ES	0.8125	0.4375	42.098087	176.640518	6059.74560
GA(2002)	0.81250	0.43750	42.097398	176.65405	6059.94634
CPSO	0.8125	0.4375	42.091266	176.7465	6061.0777
GA(1997)	0.9375	0.5000	48.3290	112.6790	6410.3811
Lagrangian Multiplier	1.125	0.625	58.291	43.690	7198.200
Branch-bound	1.125	0.625	48.97	106.72	7982.5
GSA	1.125	0.625	55.9886598	84.4542025	8538.8359

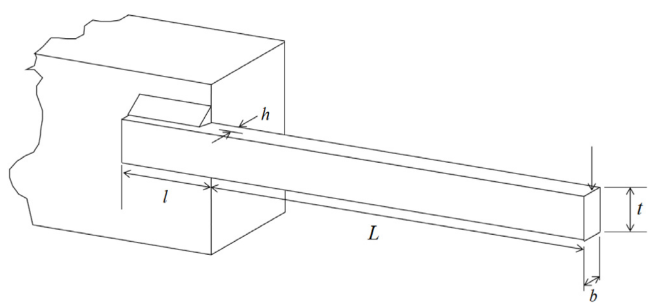

Fig. 11. Welded beam design.

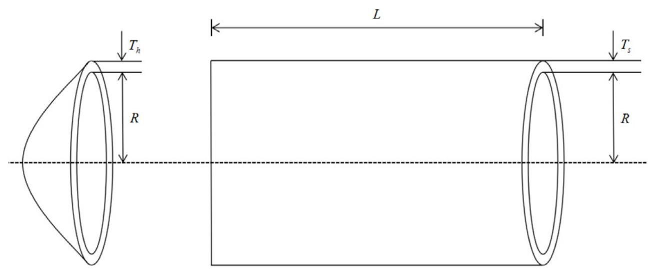

Fig. 12. Pressure vessel design problem.

RO [37] (Kaveh & Khayatazad, 2012); ES [39] (MezuraMontes & Coello,2008). PSO-SCA [40] (Liu, Cai, & Wang, 2010), HPSO [41]; (He & Wang, 2007b), ACO [42]; (Kaveh & Talatahari, 2010), Lagrangian Multiplier [43]; (Kannan & Kramer, 1994), Branch-bound [44]; (Sandgren, 1990). The comparison results are shown in Table 9.

Table 9 shows that the result of RFSCA is better than OBSCA, and the result of OBSCA outperform other algorithms, the results of OBSCA and RFSCA outperform other algorithms.

5. Conclusion and future works

In this study, the Riesz fractional derivative is firstly introduced into the swarm intelligence optimization algorithm, and a sine cosine algorithm based on the Riesz fractional mutation strategy is proposed. The proposed algorithm adopts random function and QOL strategy to initialize the population, and integrates QOL strategy and OBL strategy to enhance the global exploration ability of the algorithm. A new strategy with smoothness and memory preservation is designed for the optimal solution, so that the optimal solution has an independent update formula, which is the biggest difference between RFSCA and other algorithms, and it is also the root cause of RFSCA performs well. The Riesz fractional order mutation strategy improves the convergence speed of the algorithm. The global exploration and local exploitation capability of the algorithm are effectively coordinated. RFSCA adopts a greedy selection strategy, which theoretically ensuring that the RFSCA approaches the global optimal solution by probability.

Sections 4.2 and 4.3 show the improvement of RFSCA by the two strategies respectively. The experimental results show that the RFSCA is superior to the classical SCA and OBSCA in convergence accuracy and stability. And the RFSCA complexity is similar to OBSCA, this shows that RFSCA has better performance than similar algorithms. Compared with GWO, PSO, OPSO, SSA, MFO, the RFSCA has better convergence ability. However, the cons of RFSCA are also obvious — the exploration capability still needs to be strengthened, such as the lack of RFSCA performance in some test functions (mixed function F11–F20 in Standard IEEE CEC 2017, F16–F20 in 23 benchmark functions). In addition, the results of two constrained engineering problems (Welded beam design, compression spring, and pressure vessel) verify the effectiveness of RFSCA.

Future works include classifying populations and updating them with different improve rules, and apply the proposed algorithm to solve more practical optimization issues.

Acknowledgments

The authors thank all the editors and the anonymous referees for their hard works. This work is supported by the National Key research and development program, China and National Natural Science Foundation of china under grant number 2017 YFC0403605, 61772416 and 11601419.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.asoc.2019.04.044.

References

[1] S. Mirjalili, SCA: A sine cosine algorithm for solving optimization problems, Knowl. Based Syst. 96 (2016) 120–133.
[2] M.A. Elaziz, D. Oliva, S. Xiong, An improved opposition-based sine cosine algorithm for global optimization, Expert Syst. Appl. 90 (2017) 484–500.
[3] H. Nenavath, R.K. Jatoth, Hybridizing sine cosine algorithm with differential evolution for global optimization and object tracking, Appl. Soft Comput. 62 (2018) 1019–1043.
[4] R.M. Rizk-Allah, Hybridizing sine cosine algorithm with multi-orthogonal search strategy for engineering design problems, J. Comput. Des. Eng. 5 (2) (2018) 249–273.
[5] K. Chellapilla, Combining mutation operators in evolutionary programming, IEEE Trans. Evol. Comput. 2 (3) (1998) 91–96.
[6] H. Wang, H. Li, C. Li, et al., Opposition-based particle swarm algorithm with cauchy mutation, in: 2007 IEEE CEC, Singapore, 2007, pp. 4750–4756.
[7] Q. Wu, Hybrid forecasting model based on support vector machine and particle swarm optimization with adaptive and cauchy mutation, Expert Syst. Appl. 38 (8) (2011) 9070–9075.
[8] F. Jiang, H. Xia, Q. Tran, et al., A new binary hybrid particle swarm optimization with wavelet mutation, Knowl. Based Syst. 130 (2017) 90–101.
[9] A.A. Kilbas, H.M. Srivastava, J.J. Trujillo, Theory and applications of fractional differential equations, 204, 2006.
[10] F. Riewe, Mechanics with fractional derivatives, Phys. Rev. E. 55 (3) (1997) 3581–3592.

[11] V.V. Anh, N.N. Leonenko, A. Sikorskii, Stochastic representation of fractional bessel-riesz motion, Chaos Solitons Fractals 102 (2017) 135–139.
[12] C.K. Shiva, V. Mukherjee, Automatic generation control of interconnected power system for robust decentralized random load disturbances using a novel quasi-oppositional harmony search algorithm, Int. J. Electr. Power Energy Syst. 73 (2015) 991–1001.
[13] C.K. Shiva, G. Shankar, V. Mukherjee, Automatic generation control of powersystem using a novel quasi-oppositional harmony search algorithm, Int. J. Electr. Power Energy Syst. 73 (2015) 787–804.
[14] H.R. Tizhoosh, Opposition-based learning: a new scheme for machine intelligence, in: CIMCA-IAWTIC, IEEE, Vienna, 2005, pp. 695–701.
[15] X.Z. Gao, X. Wang, S.J. Ovaska, K. Zenger, A hybrid optimization method of harmony search and opposition-based learning, Eng. Optim. 44 (8) (2012) 895–914.
[16] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput. 3 (2) (1999) 82–102.
[17] J. Digalakis, K. Margaritis, On benchmarking functions for genetic algorithms, Int. J. Comput. Math. 77 (4) (2001) 481–506.
[18] S. Mirjalili, A. Lewis, S-shaped versus v-shaped transfer functions for binary particle swarm optimization, Swarm Evol. Comput. 9 (2013) 1–14.
[19] S. Mirjalili, S.M. Mirjalili, X.S. Yang, Binary bat algorithm, Neural Comput. Appl. 25 (3–4) (2014) 663–681.
[20] N. Awad, M. Ali, J. Liang, B. Qu, P. Suganthan, Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective real-parameter numerical optimization.
[21] J. Kenndy, R. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, Vol. 4, Southern California. USA, 1995, pp. 1942–1948.
[22] H. Wang, H. Liu, C. Li, S.Y. Zeng, Opposition-based particle swarm algorithm with cauchy mutation, in: 2007 IEEE Congress on Evolutionary Computation, Singapore, 2007, pp. 4750–4756.
[23] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61.
[24] S. Mirjalili, Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm, Knowl. Based Syst. 89 (2015) 228–249.
[25] S. Mirjalili, A.H. Gandomi, S.Z. Mirjalili, S. Saremi, H. Faris, S.M. Mirjalili, Salp swarm algorithm: a bio-inspired optimizer for engineering design problems, Adv. Eng. Softw. 114 (2017) 163–191.
[26] J. Zhou, X. Yao, Multi-population parallel self-adaptive differential artificial bee colony algorithm with application in large-scale service composition for cloud manufacturing, Appl. Soft Comput. 56 (2017) 379–397.
[27] S.M.A. Pahnehkolaei, A. Alfi, A. Sadollah, J.H. Kim, Gradient-based water cycle algorithm with evaporation rate applied to chaos suppression, Appl. Soft Comput. 53 (2016) 420–440.
[28] T. Yokota, T. Taguchi, M. Gen, A solution method for optimal cost problem of welded beam by using genetic algorithms, Comput. Ind. Eng. 37 (1–2) (1999) 379–382.
[29] K.E. Parsopoulos, M.N. Vrahatis, Particle swarm optimization method for constrained optimization problem, Front. AI. Appl. 76 (1) (2002) 214–220.
[30] R.A. Krohling, L.S. Coelho, Coevolutionary particle swarm optimization using gaussian distribution for solving constrained optimization problems, IEEE Trans. Syst. 36 (6) (2006) 1407–1416.
[31] K.S. Lee, Z.W. Geem, A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice, Comput. Methods Appl. Mech. Eng. 194 (36–38) (2005) 3902–3933.
[32] K. Deb, Optimal design of a welded beam via genetic algorithms, Aiaa J. 29 (1991) 2013–2015.
[33] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, GSA: a gravitational search algorithm, Inf. Sci. 179 (13) (2009) 2232–2248.
[34] S. Mirjalili, A. Lewis, The whale optimization algorithm, Adv. Eng. Softw. 95 (2016) 51–67.
[35] K. Ragsdell, D. Phillips, Optimal design of a class of welded structures using geometric programming, ASME J. Eng. Ind. 98 (1976) 1021–1025.
[36] S. Mirjalili, S.M. Mirjalili, A. Hatamlou, Multi-verse optimizer: a nature-inspired algorithm for global optimization, Neural Comput. Appl. 27 (2016) 495–513.
[37] A. Kaveh, M. Khayatazad, A new meta-heuristic method: ray optimization, Comput. Struct. 112–113 (2012) 283–294.
[38] C.A.C. Coello, E.M. Montes, Constraint-handling in genetic algorithmsthrough the use of dominance-based tournament selection, Adv. Eng. Inf. 16 (2002) 193–203.
[39] E. Mezura-Montes, C.A.C. Coello, An empirical study about the usefulness of evolution strategies to solve constrained optimization problems, Int. J. Gen. Syst. 37 (2008) 443–473.
[40] H. Liu, Z. Cai, Y. Wang, Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization, Appl. Soft Comput. 10 (2010) 629–640.
[41] Q. He, L. Wang, A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization, Appl. Math. Comput. 186 (2007) 1407–1422.
[42] A. Kaveh, S. Talatahari, An improved ant colony optimization for constrained engineering design problems, Eng. Comput. 27 (2010) 155–182.
[43] B. Kannan, S.N. Kramer, An augmented lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design, J. Mech. Des. 116 (1994) 405–411.
[44] E. Sandgren, Nonlinear integer and discrete programming in mechanical design optimization, J. Mech. Des. 112 (1990) 223–229.