Hindawi
Wireless Communications and Mobile Computing
Volume 2019, Article ID 2965061, 18 pages
https://doi.org/10.1155/2019/2965061

Research Article

SFDE: Shuffled Frog-Leaping Differential Evolution and Its Application on Cognitive Radio Throughput

Hongbo Wang[1,2] **Xiaoxiao Zhen,**[1] **and Xuyan Tu**[1]

[1]*School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China*
[2]*Beijing Key Lab of Knowledge Engineering for Materials Science, No. 30 Xueyuan Road, Haidian Zone, Beijing 100083, China*

Correspondence should be addressed to Hongbo Wang; foreverwhb@126.com

Received 2 July 2018; Revised 11 January 2019; Accepted 10 February 2019; Published 4 March 2019

Guest Editor: Charles Yaacoub

Differential Evolution (abbreviation for DE) is showing many advantages in solving optimization problems, such as fast convergence, strong robustness, and so on. However, when DE faces a complex target space, the diversity of its population will degenerate in a small scope; even sometimes it is premature to fall into the local minimum. All things contend in beauty in the world; a Shuffled Frog Leaping Algorithm (abbreviation for SFLA) has a strong global ability; unfortunately, its convergence speed is also slow. In order to overcome the shortcoming, this article suggests a Shuffled Frog-leaping Differential Evolution (abbreviation for SFDE) algorithm in a cognitive radio network, which combines Differential Evolution with Shuffled Frog Leaping Algorithm. This proposed method hikes its local searching for a certain number of subgroups, and their individuals join together and share their mutual information among different subgroups, which improves the population diversity and achieves the purpose of fast global search during the whole Differential Evolution. The SFDE is examined by 20 well-known numerical benchmark functions, and those obtained results are compared with four other related algorithms. The experimental simulation in solving the problem of effective throughput optimization for cognitive users shows that the proposed SFDE is effective.

1. Introduction

The convergence of the basic Differential Evolution (abbreviation for DE) is closely related to its control parameters. Many researchers have worked on improving its performance in various ways and developed many variants. In [1], a method of Self-adaptive Differential Evolution algorithm with discrete mutation control parameters (DMPSADE) is proposed. In DMPSADE, each vector has its own mutation, cross parameters, and control strategies. Those original unified control parameters and mutation strategies are divided into each individual, so that the evolutionary granularity becomes smaller and the performance improves. In [2], a fuzzy system is introduced to dynamically adaptive control parameters to improve DE. An improved DE named ADE is proposed in [3], where the author suggests DE with two-level parameter adaptation. The first level of control parameters F_P and CR_P is used to control the global optimization ability, and the second level of control parameters F_i and CR_i corresponds with each individual to improve the local search of the algorithm. The main characteristic of DE is its mutation strategy. Price and Storn proposed more than ten different differential strategies to implement mutation operation such as *DE/rand/1/bin*, *DE/rand/2/bin*, *DE/best/1/bin*, *DE/rand-to-best/bin,* and so on. Among them, *DE/rand/1/bin* and *DE/best/2/bin* are the most widely used and most successful differential strategies. The former can maintain the diversity of the population, and the latter pays more attention to the speed of its convergence.

Reference [4] introduces a fitness Euclidean-distance ratio for multimodal optimization. In [5], a novel mutation strategy, named *elite/rand/1*, which divides a whole population into two subpopulations on the basis of the fitness and then extracts the maximum information from an elite individual. MDE in [6] includes three new steps: Opposition-Based Learning (OBL), tournament method for mutation, and a single population structure. A parameter adaptive selection can dynamically evaluate their performance during evolution and guide the direction of its development in the next iteration process [7]. Based on the abstract convex theory, a dynamic adaptive differential evolution algorithm

(DADE) is proposed in [8]. Shuffled Frog Leaping Algorithm (SFLA) is proposed in [9], which is inspired by the individuals learning from each other during the foraging behaviour of frogs. In [10], a multiphase hybrid algorithm implements a local depth search using SFLA for every cluster and readjusts its global solutions. An improved hybrid bifurcation algorithm based on cloud model is proposed in [11], which focuses on the transformation between qualitative and quantitative computing. Reference [12] proposed crossing and variation frog leaping algorithm (KSFLA), where the individuals of the subpopulations vary from the ranking list before producing new individuals instead of the poor ones. Reference [13] proposed a Self-adaptive Differential Evolution algorithm with Improved Mutation Mode (IMMSADE), which improves the mutation model of DE and introduces some new control parameters.

Based on the above discussion, SFLA has slow the convergence, low efficiency, and precision in the high dimension continuous optimization problems, in that its core strategy is to continuously update the location of the worst individual. But DE does not do anything with the worst individual. A hybrid algorithm based on frog leaping algorithm and Differential Evolution is suggested in this article, which combines the advantages of SFLA and DE, in order to obtain a significant performance on convergence speed and the accuracy.

The rest of this paper is organized as follows. In order to ensure the readability, the related mathematics description of Differential Evolution and Shuffled Frog Leaping Algorithm is summarized in Section 2. Section 3 formally defines models of an improvement in a Shuffled Frog-leaping Differential Evolutionary algorithm (SFDE). Section 4 designs the comparative experiments of well-known public benchmark functions and analyses the related T-test and F-test results for proof of the effective SFDE. The analysis of convergence in SFDE is described in detail in Section 4. Simulation experiments on the problem of effective throughput optimization for cognitive users are given in Section 5. Section 6 concludes this article with a summary and future direction.

2. The Related Work

2.1. About Differential Evolution. DE is a random search based on population and also a heuristic swarm intelligence algorithm. It mainly uses mutation, crossover, and selection operations to perform an intelligent evolutionary search. DE has five operations, which are described as follows.

(1) Initialization Operation. Initializing a population randomly in the search space, the *j-th* parameter of the *i-th* individual in the *0-th* generation is initialized by formula (1):

$$x_{j,i}(0) = x_{j,i}^{Low} + rand(0,1) \cdot \left(x_{j,i}^{Up} - x_{j,i}^{Low}\right), \quad (1)$$

In formula (1), x^{Up} and x^{Low} are the upper and lower boundary of the *j-th* parameter of the *i-th* individual, respectively. $rand(0,1)$ denotes a random number, which is limited between 0 to 1.

(2) Mutation Operation. In DE, the most common in mutation operations is *DE/rand/1,* which selects two individuals at random in a population. Then the vectors are combined with the target vector for mutating into an intermediate vector $v_i(g)$. The process can be expressed as follows in formula (2):

$$v_i(g) = x_{r1}(g) + F \cdot (x_{r2}(g) - x_{r3}(g)) \quad (2)$$

Obviously, the smaller the difference between $x_{r2}(g)$ and $x_{r3}(g)$ is, the smaller the difference between $x_{r1}(g)$ and $v_i(g)$ is. This means that, in the initial stage, the disturbances are large because of the far distance between individuals, and the search range of the algorithm is large. But in the later period of the iteration, the detective range will be a shrink. Several common mutation strategies are shown in the following formulas (3), (4), and (5):

① *DE/rand/2*

$$v_i(g) = x_{r1}(g) + F(x_{r2}(g) - x_{r3}(g)) + F(x_{r4}(g) - x_{r5}(g)) \quad (3)$$

② *DE/current-to-best/1*

$$v_i(g) = x_i(g) + F(x_{best}(g) - x_i(g)) + F(x_{r1}(g) - x_{r2}(g)) \quad (4)$$

③ *DE/best/1*

$$v_i(g) = x_{best}(g) + F(x_{r2}(g) - x_{r3}(g)) \quad (5)$$

Among them, the subscript variables r_1, r_2, r_3, r_4, and r_5 represent different individuals and $x_{best}(g)$ represents the optimal vector in the g-*th* generation.

(3) Correcting Operation. After the mutation operation, the algorithm must ensure that each component (gene) of the intermediate vector satisfies the predefined boundary constraints. If a component (gene) of an intermediate vector exceeds the scope, the vector individual must be corrected. The following formula (6) is a simple but most frequently used correction strategy:

$$v_i(g) = \begin{cases} \min\left\{x_{j,i}^{Up}, 2x_{j,i}^{Low} - v_i(g)\right\}, & if\ v_i(g) < x_{j,i}^{Low} \\ \max\left\{x_{j,i}^{Low}, 2x_{j,i}^{Up} - v_i(g)\right\}, & if\ v_i(g) > x_{j,i}^{Up}. \end{cases} \quad (6)$$

(4) Crossover. Crossover is to mixture the $x_{j,i}(g)$ and $v_{j,i}(g)$ in the g-*th* generation, producing test vectors. The process can be expressed as follows in formula (7):

$$u_{j,i}(g) = \begin{cases} v_{j,i}(g), & if\ rand(0,1) \le CR\ or\ j = j_{rand} \\ x_{j,i}(g), & otherwise \end{cases} \quad (7)$$

In formula (7), CR is the cross probability and j_{rand} is a random integer of $[1, 2 \ldots D]$.

(5) Selecting Operation. By comparing the function values of the target vectors in the test function $f(x_i(g))$ with the test function values of the corresponding test vectors $f(u_i(g))$, which is determined whether the test vectors can be put into the next generation or not, the standard Differential Evolution algorithm employs a greedy algorithm for the selection operation. The process can be expressed as follows in formula (8):

$$x_i(g+1) = \begin{cases} u_i(g), & if\ f(u_i(g)) \le f(x_i(g)) \\ x_i(g), & otherwise. \end{cases} \tag{8}$$

2.2. Shuffled Frog Leaping Algorithm. Suppose that there are N frogs in the n-dimensional space catching their food. The basic Shuffled Frog Leaping Algorithm sorts all the individuals in descending order according to the fitness of the objective function and then puts the first frog individual into the first group, the second frog individual into the second group, and so forth. The first m frogs entry own group in order, respectively; thus each group contains a frog. Beginning with the $m+1st$ frog, the remaining frogs join their own group according to the rule of Round Robin (namely, *Mod m*).

In each group, $X_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ represents the current position of the $i-th$ frog. $pbest_i = \min(pbest_{i1}, pbest_{i2}, \ldots, pbest_{io})$ is the optimal location in the $i-th$ group, called the local optimal position. $pworst_i = \max(pworst_{i1}, pworst_{i2}, \ldots, pworst_{iM})$ is the worst in the $i-th$ group, called the local worst position.

Let $f(X)$ be a minimal objective function, and use formula (9) to calculate the local optimal frog location for the $i-th$ group.

$$pbest_i(t+1) = \begin{cases} pbest_i(t), & if\ f(X_i(t+1)) \ge f(pbest_i(t)) \\ X_i(t+1), & if\ f(X_i(t+1)) < f(pbest_i(t)) \end{cases} \tag{9}$$

Assume that the number of the frogs is N, and it is divided into m groups according to the grouping operator. The optimal position $gbest(t)$ for each group is the global optimal location, as shown in formula (10).

$$gbest(t) = \min\{f(pbest_1(t)), f(pbest_2(t)), \ldots, f(pbest_m(t))\} \tag{10}$$

According to the local location update operator, adjusting the position of the worst frog in each group, the concrete adjustment method is as follows.

The frog moving distance is as shown in formula (11):

$$pmove_i = rand * (pbest_i - pworst_i) \tag{11}$$

Update the worst frog individuals as follows:

$$pworst_i = pworst_i + pmove_i\ (-pmove_{max} \ll pmove_i \ll pmove_{max}) \tag{12}$$

In formula (11), *rand* is a random number between 0 and 1. $pmove_{max}$ in formula (12) represents the maximum distance that the frog can move during its foraging for food. If the positions obtained from formulas (11) and (12) are closer to the food source than the position of the worst frog in the group, the worst frog leaps to this new position. The position of the worst frog $pworst_i$ is further closer to the food source. Otherwise, $pbest_i$ is replaced by $gbest_i$, and the worst frog continues to search for a new leaping position as follows:

$$pmove_i = \text{rand} * (gbest_i - pworst_i) \tag{13}$$

$$pworst_i = pworst_i + pmove_i,\quad (-pmove_{max} \ll pmove_i \ll pmove_{max}) \tag{14}$$

If the position obtained from formulas (13) and (14) is still not closer to the food source than the worst frog in the group, or the worst frog individual needs to skip more than the maximum step size, the worst frog leaps to a new position at random. Otherwise, the worst frog leaps to the new position generated from formula (14). SFLA performs an iterative evolution of the frogs within each group via adjusting and updating of the worst frog.

In the evolutionary process of SFLA, each iteration involves the local optimal solution $pbest_i$ and the global optimal $gbest$. In the whole population, there exist m groups, and therefore there are m local optimal solutions, and there is only one global optimal. SFLA uses a grouping operator and a mechanism of individual integration into groups for transmitting messages and sharing information. The grouping operator divides all the frogs in the population into m groups, and then the worst frog in each group updates itself local search strategy. When all groups finish searching, a shuffle share information and interact with each other. This mechanism makes SFLA have better global search ability. The procedure of basic SFLA is shown as follows.

Step 1. Initialize the positions of all the frogs in the population at random, to make the frogs randomly distributed in the search space, and initialize the population size N, the number of frogs in a group Q, the number of evolution of the *inner_iteration*, the maximum step size of the frogs allowed to move $pmove_{max}$, and maximum number of iterations of population *max_iteration*.

Step 2. Calculate the fitness of all frogs in the population and sorting them in descending order. *gbest* is initialized with the best frog position, and the frogs are sorted into m groups according to a grouping operator.

Step 3. Adjust the worst frog in the group according to formulas (11) and (12).

Step 4. If the position of the frog in Step 3 is improved, replace the worst frog position in the group with the new position, then skip to Step 6, or readjust the worst frog position of the group according to formulas (13) and (14).

Step 5. If a position of the frog from Step 4 is improved, replace the worst frog position in the group with the new position, then skip to Step 6, or generate a new random position instead of the worst frog position in the current group.

Step 6. Calculate the fitness of all frogs in the group, and sort them in descending order to determine if the internal iteration is over. If not, skip to Step 3.

Step 7. Share shuffle information among all the frogs in the same group.

Step 8. Judge whether the end condition meets or not. If it does, output the optimal solution directly. If not, skip to Step 2 and it continues.

The end condition of SFLA can achieve the maximum number of iterations or meet an error criterion, which may be a combination of the above two. In the case where the optimal value of the objective function and the allowable error are determined in advance, the end condition can be set to satisfy a certain error range. When the difference between the objective value and the optimum is less than the setting error, output the optimal value; otherwise it continues.

3. The Shuffled Frog-Leaping Differential Evolutionary (SFDE)

Although combining a new algorithm with simple assembly can improve the accuracy or convergence of the algorithm to a certain degree, it is still far from the theoretical value of the test function, so how to integrate the two algorithms is the most important problem.

3.1. Integration Strategy between SFLA and DE. There are three main steps in the integration strategy between SFLA and DE. Step (1): initialize populations. Step (2): divide the whole population into several groups, and integrate within a group. Step (3): shuffle the whole population after iterations in each group. We use the idea of dividing the whole population into several subpopulations and considering each subpopulation as a group. So before dividing the whole population into groups, individuals are sorted by the fitness value of the test function, then a new group with the strategy is shown as in Figure 1.

As shown in Figure 1, SFLA helps to adjust the worst individuals in the groups. If the fitness value of the individuals generated from formulas (11) and (12) is improved, the worst one in the original group will be replaced. Otherwise, its position of the group should be readjusted according to formulas (13) and (14). If it is worse than original one; then discard it and randomly generate a new one to replace it. Then

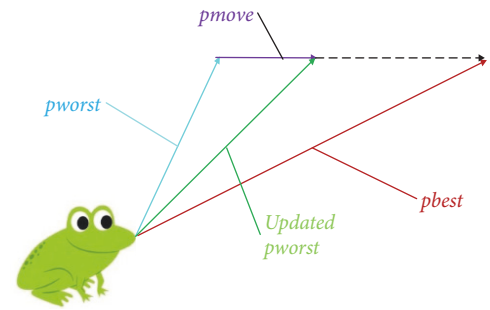


Figure 1: The worst individual position adjustment process in the SFDE.

we use DE to deal with the new population. The operations include mutation, crossover, and selection.

The evolutionary process of the individual population is shown in Figure 2, where the sequence (1, 2, 3, 4) is the stage number of each group evolution. (1) The worst individual position is adjusted for each subpopulation according to Figure 1, then the operation of the Differential Evolution for all the individuals in the subpopulation, that is, the variation of the evolution and the selection operation. (2) The whole population run mutation and cross operation of Differential Evolution stage, thus resulting in the middle of the vector of individuals (represented in Figure 2 with a red circle of individuals) for the latter part of the selection operation. (3) Select the operation for the generation of intermediate individuals and original target individuals.

In Figure 2, the next generation has the intermediate vector individuals (individual with red virtual coil) and the original target individuals. In SFDE, the mutation strategy uses *DE/best/1/bin*.

3.2. The Basic Process of Shuffled Frog-Leaping Differential Evolutionary (SFDE). The basic SFDE can be described in Algorithm 1.

In Algorithm 1, T is the number of subpopulation; $gbest_i$ denotes the best individual in population; $pworst_i$ is the worst individual in subpopulation; $pbest_i$ is the best individual in subpopulation; $v_{i,j}$ denotes the current mutated vectors.

4. Experimentation

This section compares the proposed SFDE with twenty classic well-known test functions.

4.1. Experimental Setting and Parameterization. Twenty test functions with 30 high dimensions check the performance of SFDE. The detailed definitions of them are shown in Table 1.

Experimental environment configuration: operation system is Windows 7. Minimum memory is 4G; processor type is Intel (R) Core (TM) i5-2370M @ 2.40GHz; development tools and version are Matlab-R2012a.

To be fair, the initial conditions of each algorithm are consistent. (1) Population size of all Differential Evolution

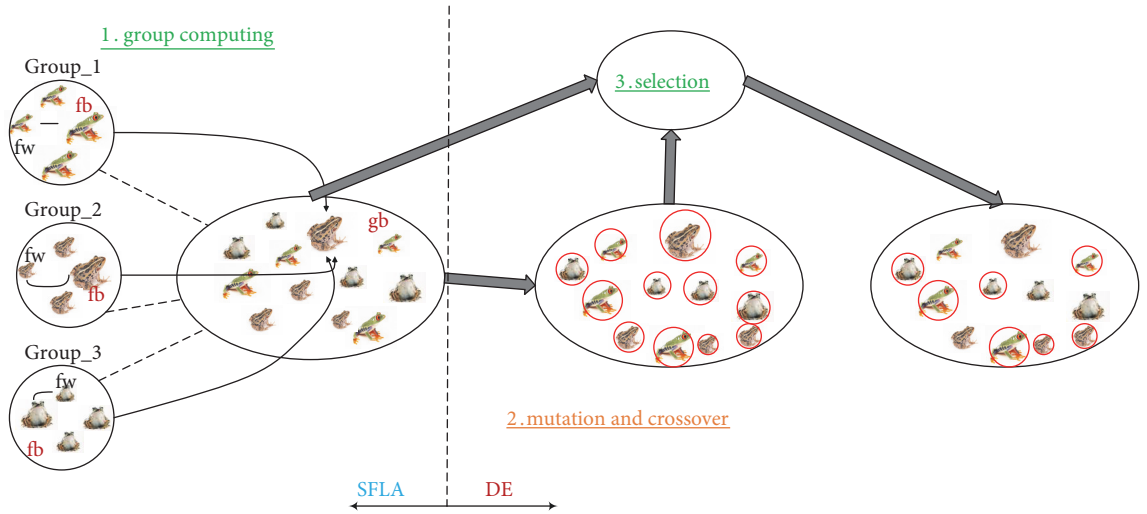


Figure 2: The integration strategy between *SFLA* and *DE*.

algorithms is set to be 200. (2) The number of subpopulations of SFDE is set to 10. (3) The number of iterations is set according to the complexity of the problem, but the number of iterations of all Differential Evolution algorithms is equal every time.

4.2. Computational Results and Discussion. We record the optimal value obtained at the end of every computing in 30 runs. Then, the average optimal value and the standard deviation of the optimal value are calculated by using the 30 dependently runs. And the standard deviation of the best average value and the optimal value is inputted into the table to facilitate the analysis of the experimental results. In order to highlight the experimental data close to the theoretical value of the function, we test each test function in the table closest to the theoretical value of the experimental data bold. The output images reflect the variation curve of the average optimal value for each test function in 30 separate experiments.

The convergence curves plots of five algorithms for 9 benchmark functions with 30 dimensions are drawn in Figure 3 (Ackle, Michalwicz Function and Schwefel's Problem 1.2), Figure 4 (Perm Function, Rastrigin, Sphere Function), and Figure 5 (Rotated Hyper-Ellipsoid, Shifted and Zakharov Function, Dixon-Price Function), respectively. The 9 results confirm the good performance of SFDE. The optimal value of the function curve is shown in Figure 3 and the full optimal value of the function is shown in Table 2.

Schwefel's Problem 1.2 function's algorithm is difficult to obtain the theoretical value on 30 dimensions (iterations: 2000 times). From the experimental results, it can be seen that the convergence rate of SFDE is faster than that of other algorithms, and the final accuracy is also better than other algorithms, and even 34 units of magnitude are higher than the accuracy of EPSDE algorithm.

Rotated Hyper-Ellipsoid is a continuous, convex single-peak, which is an extension of the axis-parallel super-ellipsoid function and also becomes a square sum function. From the experimental results, SFDE is better than other algorithms, and with the iteration of the algorithm, the accuracy of the algorithm is getting better.

The Perm Function is characterized by a large number of local optimal value of the single-peak function, and it is a well test function which can test the function of the algorithm. It can be seen from the experimental results that SFDE has the highest precision and the fastest convergence rate for Zakharov Function with 30 dimensions (iterations: 2000 times).

The convergence curves of five algorithms for 9 benchmark functions with 30 dimensions are drawn in Figure 3, respectively. For each benchmark function, there are two curves, the left one is plotted with iterations, and average function values are plotted with NFEs at right. Figures 3(a)–3(c) and 4(a) present multimodal benchmark functions and Figures 4(b)-4(c), 5(a)–5(c) present unimodal benchmark functions. Figure 3(a) is the curves of average optimal of F1. In Figure 3(b), SFDE and CODE almost have the same best value on F2. The convergence curves of compared algorithms for F3 function and F13 function are given in Figures 3(c) and 4(a). Figure 4(b) present convergence curves of test results for F10 function. In Figure 4(c), 5(a)-5(b), plots of SFDE are much better than other variants for F4, F6, and F8 functions. In Figure 5(c), SFDE demonstrates best average and convergence speed for 30 dimensions on F9 function and MDE is the second best on F9.

4.3. Results Analysis. Table 2 shows the comparison between SFDE and classic DE. It can be seen from the above experimental results that SFDE does better in most of the base functions compared with other differential evolutionary algorithms, and the convergence speed is also accelerated.

In Table 2, the eight test functions, $f_4(x)$, $f_6(x)$, $f_8(x)$, $f_9(x)$, $f_{14}(x)$, and $f_{19}(x)$, are unimodal functions, and the solving accuracy of SFDE is better than that of other algorithms. In the results of the multimodal test function, the individual test function is poor performance, such as $f_2(x)$

SFDE: Shuffled Frog-leaping Differential Evolutionary Algorithm
Input: An initial population $\{x_{i,j} \mid x_{i,j} \in [LowBoundery, UpBoundery]; i = 1, \ldots NP; j = 1, 2, \ldots D.\}$;
Benchmark $f(x)$ or object function $g(x)$;
Terminal conditions: Maximum iterations $\aleph$ or Acceptable error ε_0.
Output: optimal *gbest*.
Begin
(1) Initialization;
(2) Dividing $x_{i,j}$ into T subpopulations at random;
(3) **While** $((t < \aleph)$ *or* $(\varepsilon > \varepsilon_0))$
(4) $pbest_i(t+1) = \begin{cases} pbest_i(t), & if\ f(X_i(t+1)) \geq f(pbest_i(t)) \\ X_i(t+1), & if\ f(X_i(t+1)) < f(pbest_i(t)) \end{cases}$ /∗ computing $pbest_i$ according to Eq (9); ∗/
(5) $gbest(t) = \min\{f(pbest_1(t)), f(pbest_2(t)), \ldots, f(pbest_m(t))\}$ /∗ finding the $gbest_i$ according to Eq (10); ∗/
/∗ updating the position of the $pworst_i$ in each subpopulation according to Formulas (11) and (12); ∗/
(6) $pmove_i = \text{rand} * (pbest_i - pworst_i)$;
(7) $pworst_i = pworst_i + pmove_i$;
(8) **While** p_{worsti} is not increased
/∗ updating the position of the p_{worsti} in each subpopulation according to Formulas (13) and (14); ∗/
(9) $pmove_i = rand * (gbest_i - pworst_i)$;
(10) $pworst_i = pworst_i + pmove_i$;
(11) **End While**
(12) **For** i=1 to *NP*
(13) $v_{i,j} = x_{r1,j} + F * (x_{r2,j} - x_{r3,j})$; $v_{i,j} = x_{best,j} + F * (x_{r1,j} - x_{r2,j})$; /∗*Mutation operation;* ∗/
(14) Correction(); /∗ *Correction operation*∗/
(15) Crossover(); /∗ *Crossover operation*∗/
(16) Selection(); /∗ *Selection operation* ∗/
(17) **End For**
(18) $\varepsilon = |(f(x) - f(x^*)|$;
(19) $\boldsymbol{t} = \boldsymbol{t} + \mathbf{1}$;
(20) **End While**
(21) **Return** t, ε and *gbest*.
(22) **End**

Algorithm 1: Pseudocode of Shuffled Frog-leaping Differential Evolutionary (SFDE).

and $f_7(x)$. However, the solving accuracy of SFDE is much higher than that of other algorithms and SFDE is more stable. More importantly, SFDE algorithm itself is very simple and fast and has an absolute advantage.

In order to determine whether SFDE is more effective than other methods, statistical methods need to detect the results of CPU running time; this paper selected the paired *T-test* and the paired *F-test*. The *T-test* is an important test of the mean difference between the two samples, with the aim of checking the system error. The *F-test* is a significant test of the variance of the two samples, with the aim of testing the accident error. Usually we compare the value of 0.05. If the test result is less than 0.05, the cell will mark the result with '+', instead, the cell will mark the result with '-'. The percentage of '+' in all results is the likelihood that SFDE will handle the reliability and advantage of the test problem. Specific test results as shown in Table 3, each column is test function, efficiency, detection methods, and a variety of intelligent optimization.

To compare SFDE with other algorithms, we select advanced variants of DE, namely, JADE, DMPSADE, and IMMSADE, which is introduced in [12]. The comparison results are as shown as follows in Table 4.

Table 4 shows a clear comparison of SFDE-SaDE, SFDE-JADE, SFDE-DMPSADE, and SFDE-IMMASDE. From Table 4, it is clear to see that the convergence speed of SFDE is faster among the considered algorithms for most of test functions.

5. Simulation of Spectrum Throughput Optimization

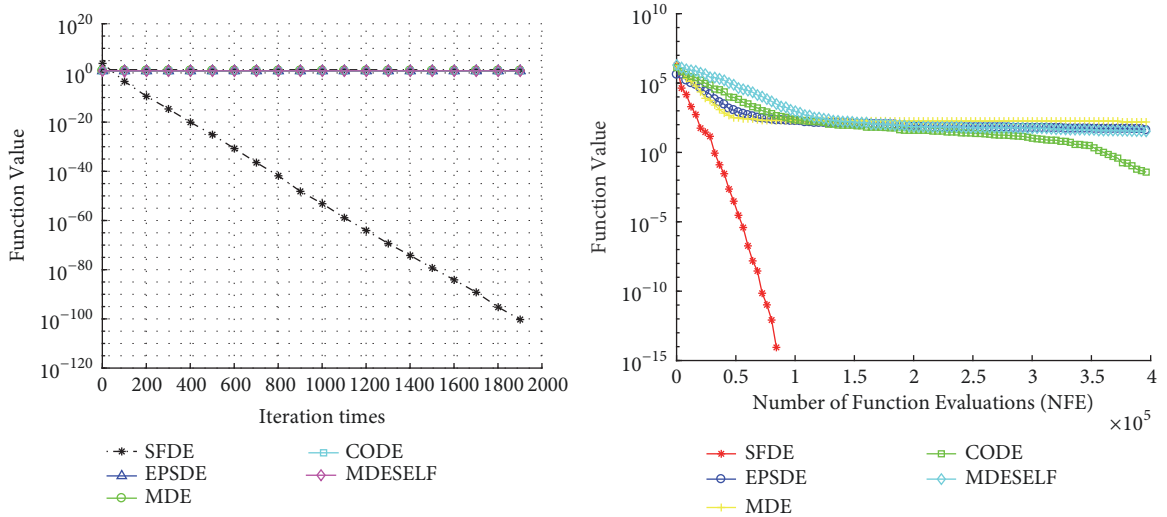
There are three ways of spectrum sensing: energy detection, matched filter detection, and cyclone-stationary property detection, the most commonly used of which is energy detection. In this way, if the length of each frame is fixed, its sensing time synchronizes with its energy accumulation. At the same time, if authorized users have good protection, the false alarm probability will be smaller and the detection rate will be larger, but the data transmission time will be shortened [14]. That is to say, the idle time of the cognitive user's usage spectrum becomes shorter, resulting in lower throughput of the cognitive user. Therefore, how to arrange an effective schedule to maximize the throughput of cognitive users (i.e., authorized users have a right of first class service) is a hot research topic.

Table 1: Detailed definitions of the test function.

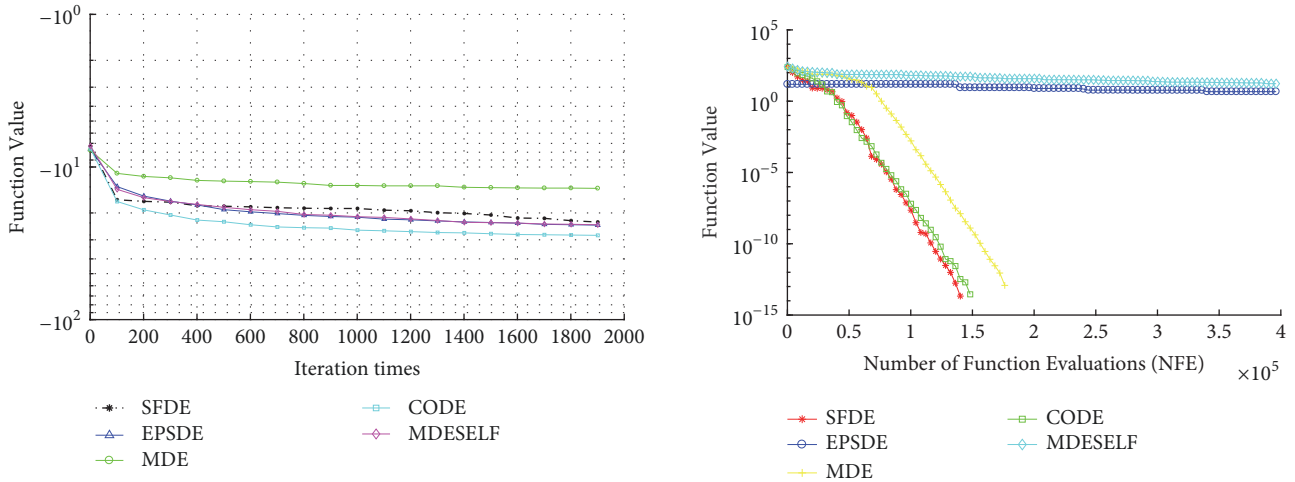
F#	Name	Description	Dim.	Range	Opt.
$f_1(x)$	Ackle	$f_1(x) = -20\exp\left(-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N}x_i^2}\right) - \exp\frac{1}{N}\sum_{i=1}^{N}\cos(2\pi x_i) + 20 + e$	30	$[-600, 600]$	0
$f_2(x)$	Michalewicz	$f_2(x) = -\sum_{i=1}^{N}\sin(x_i)\sin^{2m}\left(\frac{ix_i^2}{\pi}\right),\ m = 10$	30	$[0, \pi]$	-9.66
$f_3(x)$	Schwefel's problem 1.2	$f_3(x) = \sum_{i=1}^{N}\left(\sum_{j}^{i}x_i\right)^2$	30	$[-500, 500]$	0
$f_4(x)$	Sphere	$f_4(x) = \sum_{i=1}^{N}x_i^2$	30	$[-500, 500]$	0
$f_5(x)$	Rosenbrock	$f_5(x) = \sum_{i=1}^{N-1}\left(100\left(x_i^2 - x_{i+1}\right)^2 + \left(x_i - 1\right)^2\right)$	30	$[-100, 100]$	0
$f_6(x)$	Rotated Hyper-Ellipsoid	$f_6(x) = \sum_{i=1}^{n}\sum_{j=1}^{i}x_j^2$	30	$[-65.53, 65.53]$	0
$f_7(x)$	Shifted and Rotated Schwefel's Function	$f_7(x) = 500 + 418.9829 \times n - \sum_{i=1}^{n}g(y_i)$, $y_i = z_i + 4.20968736e + 002$, $g(y_i) =$ $\begin{cases} y_i\sin\left(\lvert y_i\rvert^{1/2}\right) & if\ \lvert y_i\rvert \le 500 \\ \left((500 - \mathrm{mod}(y_i, 500))\sin\left(\sqrt{\lvert 500 - \mathrm{mod}(y_i, 500)\rvert}\right)\right) - \frac{(y_i - 500)^2}{10000n} & if\ y_i > 500 \\ \left((\mathrm{mod}(\lvert y_i\rvert, 500) - 500)\sin\left(\sqrt{\lvert 500 - \mathrm{mod}(\lvert y_i\rvert, 500)\rvert}\right)\right) - \frac{(y_i + 500)^2}{10000n} & if\ y_i < -500 \end{cases}$ $z = M_5\left(\frac{1000(x - o_5)}{100}\right)$, $o_3 = [o_{31}, o_{32}, \ldots o_{3n}]$	30	$[-100, 100]$	500

Table 1: Continued.

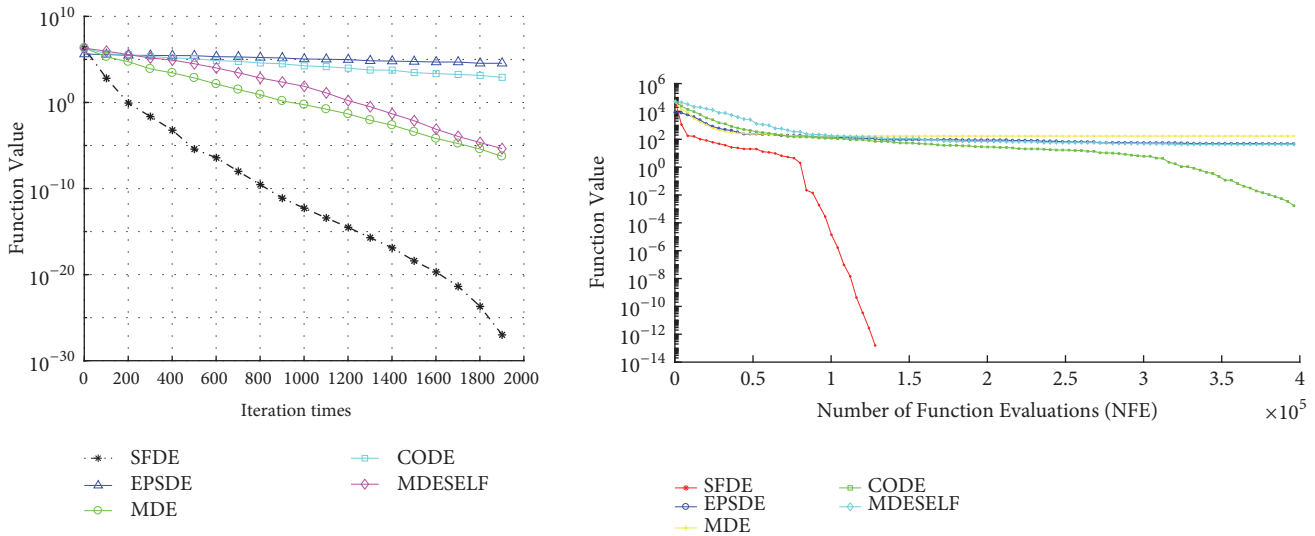
F#	Name	Description	Dim.	Range	Opt.
$f_8(x)$	Zakharov	$f_8 = \sum_{i=1}^{N} x_i^2 + \left(\sum_{i=1}^{N} 0.5i \times x_i^2\right)^2 + \left(\sum_{i=1}^{N} 0.5i \times x_i^2\right)^4$	30	$[-10, 10]$	0
$f_9(x)$	Dixon-Price	$f_9 = \sum_{i=2}^{N} i\left(2x_i^2 - x_{i-1}\right)^2 + \left(x_i - 1\right)^2$	30	$[-10, 10]$	0
$f_{10}(x)$	Perm	$f_{10} = \sum_{i=1}^{d}\left(\sum_{j=1}^{d}\left(j^i + \beta\right)\left(\left(\frac{x_j}{j}\right)^i - 1\right)\right)^2$	30	$[0, \pi]$	-9.66
$f_{11}(x)$	Griewank	$f_{11} = \frac{1}{4000}\sum_{i=1}^{N} x^2 - \prod_{i=1}^{N}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	$[-600, 600]$	0
$f_{12}(x)$	Schwefel	$f_{12} = \sum_{i=1}^{N}\left[-x_i \sin\left(\sqrt{\lvert x_i\rvert}\right)\right]$	30	$[-500, 500]$	-418.982d
$f_{13}(x)$	Rastrigin	$f_{13} = \sum_{i=1}^{N}\left(x_i^2 - 10\cos\left(2\pi x_i\right) + 10\right)$	30	$[-5.12, 5.12]$	0
$f_{14}(x)$	Sum of different powers	$f_{14} = \left(\sum_{i=1}^{N}\lvert x_i\rvert^{i+1}\right)$	30	$[-1, 1]$	0
$f_{15}(x)$	Shubert	$f_{15} = \left(\sum_{i=1}^{5} i\cos\left(\left(i+1\right)x_1 + i\right)\right) + \left(\sum_{i=1}^{5} i\cos\left(\left(i+1\right)x_2 + i\right)\right)$	2	$[-10, 10]$	-186.7309
$f_{16}(x)$	Powell	$f_{16} = \sum_{i=1}^{d/4}\left[\left(x_{4i-3} + 10x_{4i-2}\right)^2 + 5\left(x_{4i-1} - x_{4i}\right)^2 + \left(x_{4i-2} - 2x_{4i-1}\right)^4 + 10\left(x_{4i-3} - 2x_{4i}\right)^4\right]$	30	$[-4, 5]$	0
$f_{17}(x)$	Drop-Wave	$f_{17} = \frac{1 + \cos\left(12\sqrt{x_1^2 + x_2^2}\right)}{0.5\left(x_1^2 + x_2^2\right) + 2}$	2	$[-5.12, 5.12]$	-1
$f_{18}(x)$	Levy	$f_{18} = \sin^2\left(\pi\omega_1\right) + \sum_{i=1}^{d-1}\left(\omega_1 - 1\right)^2\left[1 + 10\sin^2\left(\pi\omega_1 + 1\right)\right] + \left(\omega_d - 1\right)^2\left[1 + 10\sin^2\left(2\pi\omega_d\right)\right]$	30	$[-10, 10]$	0
$f_{19}(x)$	SumSquares	$f_{19} = \sum_{i=1}^{d} i x_i^2$	30	$[-10, 10]$	0
$f_{20}(x)$	Weierstrass	$f_{20} = \sum_{i=1}^{N}\left(\sum_{k=0}^{k\max}\left[a^k\cos\left(2\pi b^k\left(x_i + 0.5\right)\right)\right] - n\sum_{k=0}^{k\max}\left[a^k\cos\left(2\pi b^k \times 0.5\right)\right]\right)$	30	$[-0.5, 0.5]$	0



(a) Ackle Function

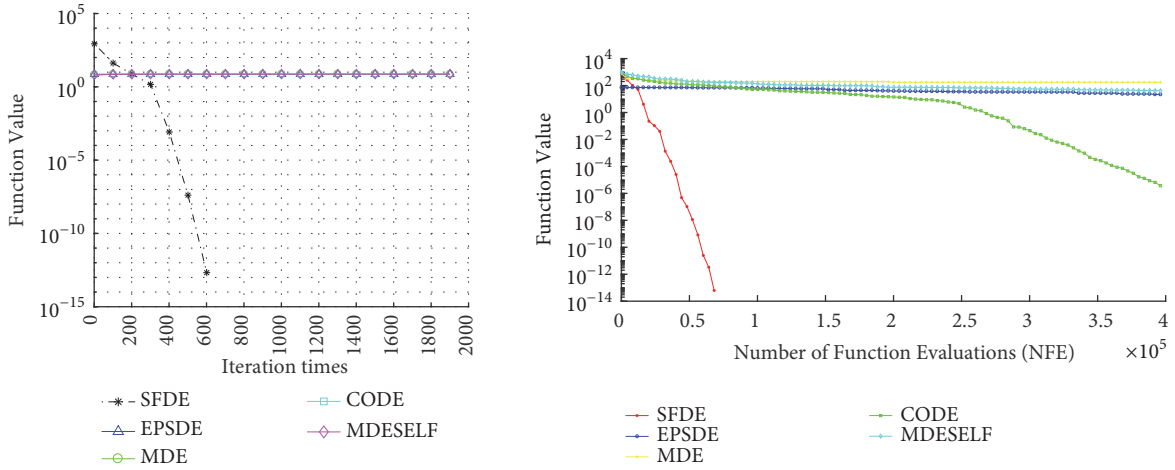


(b) Michalwicz Function

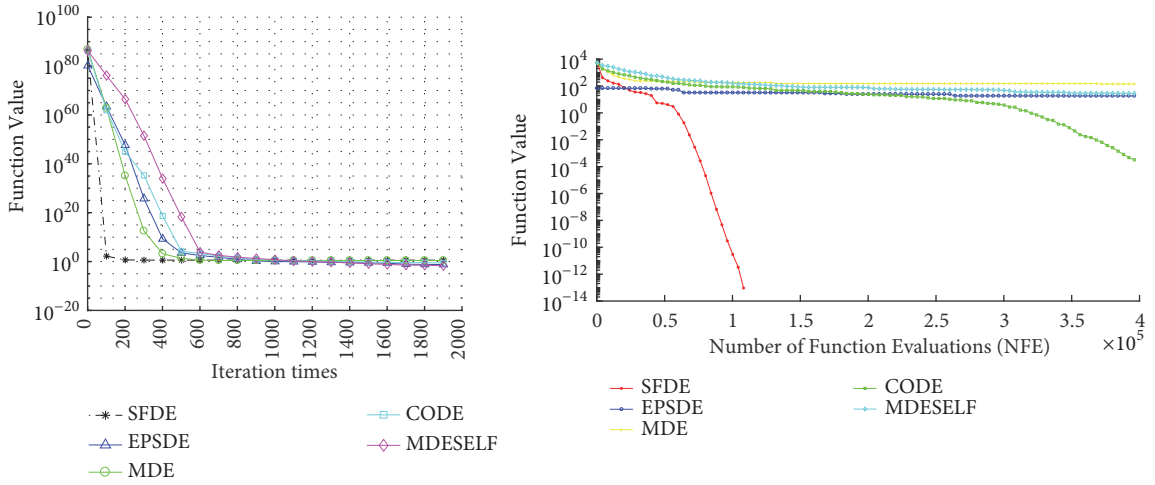


(c) Schwefel's Problem 1.2

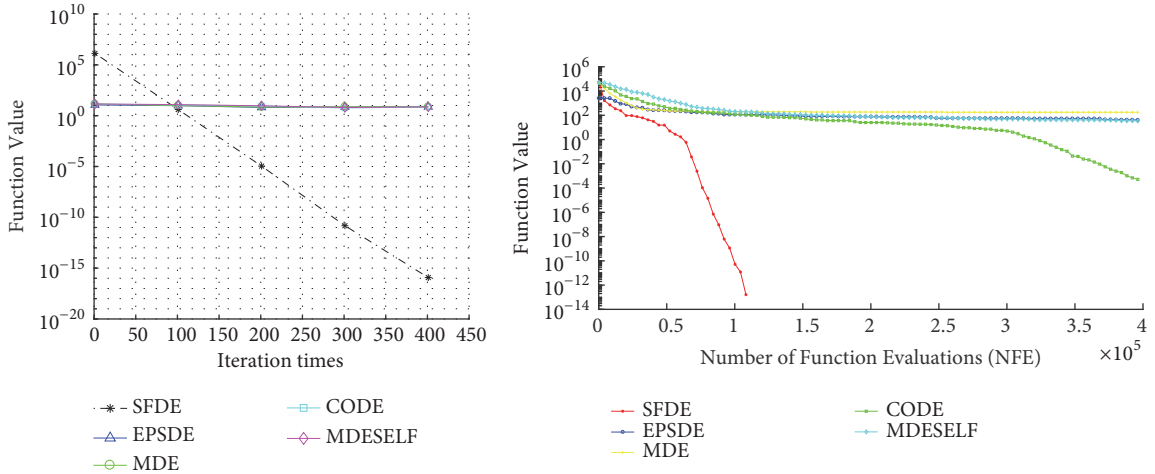
Figure 3: The 30 dimensions and curves convergence for 3 benchmark functions with 5 variants of DE.



(a) Rastrigin Function

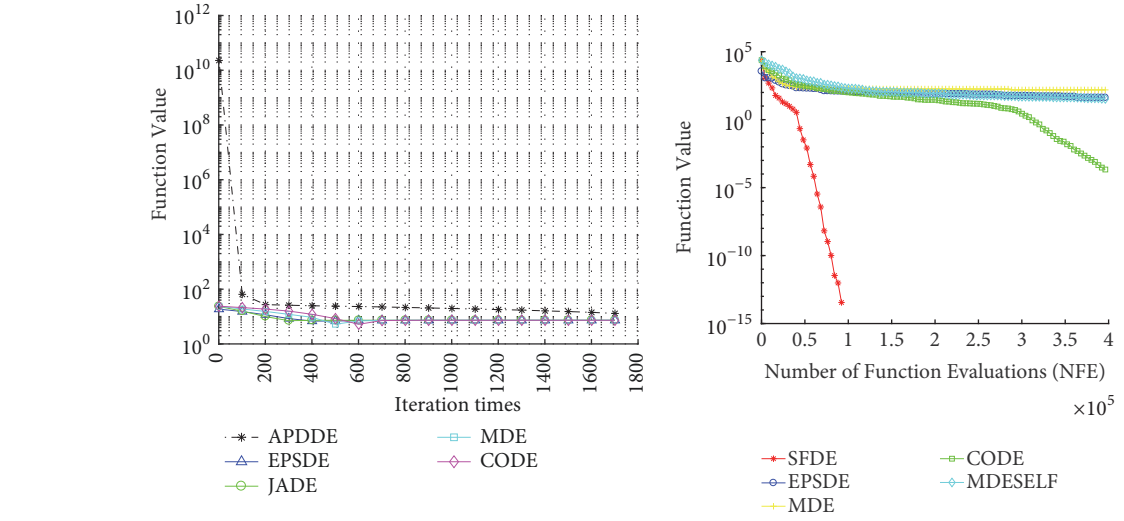


(b) Perm Function

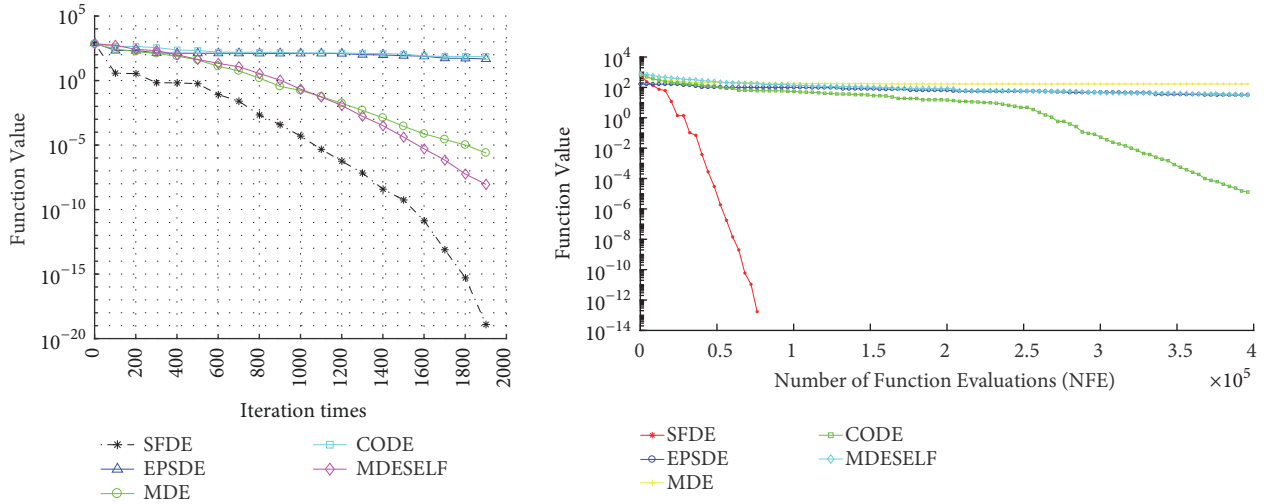


(c) Sphere Function

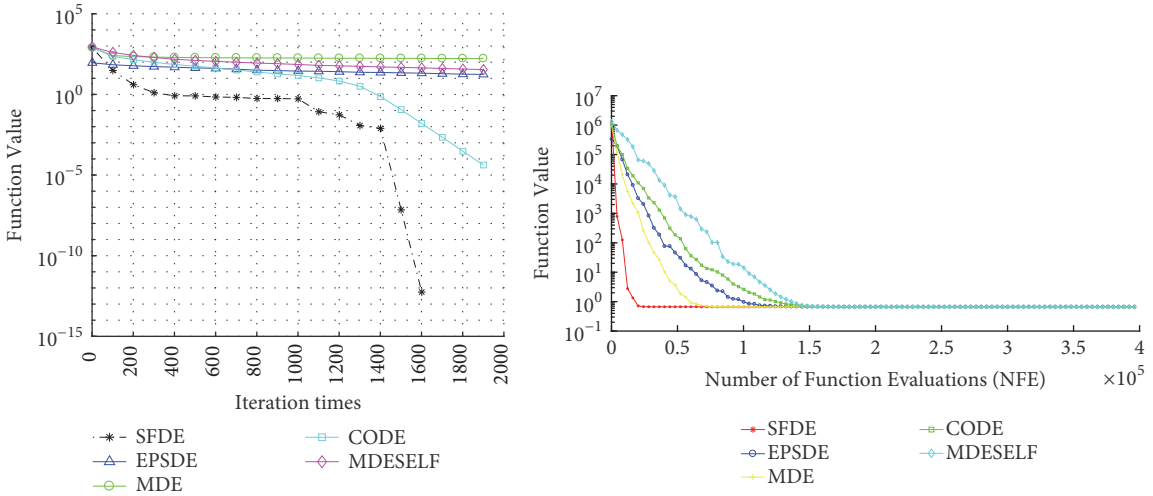
Figure 4: The 30 dimensions and curves convergence for 3 benchmark functions with 5 variants of DE.



(a) Rotated Hyper-Ellipsoid



(b) Zakharov Function



(c) Dixon-Price Function

Figure 5: The 30 dimensions and curves convergence for 3 benchmark functions with 5 variants of DE.

Table 2: Experimental data of various *DE* algorithms on the test functions (30d).

Func.	Rate	SFDE	EPSDE	MDE	CODE	MDESELF
$f_1(x)$	Avg.	**5.47E+00**	2.00E+01	2.07E+01	2.00E+01	2.00E+01
	S.D.	**2.12E-12**	1.29E-01	3.37E-02	6.49E-03	2.53E-02
$f_2(x)$	Avg.	-2.25E+01	-2.41E+01	-1.31E+01	**-2.78E+01**	-2.36E+01
	S.D.	4.99E-01	5.81E-01	4.29E-01	**1.10E-02**	1.43E-01
$f_3(x)$	Avg.	**4.16E-31**	1.94E+03	1.38E-07	4.42E+01	3.59E-07
	S.D.	**7.16E-31**	1.18E+03	1.04E-07	8.34E+00	6.22E-07
$f_4(x)$	Avg.	**4.76E-21**	2.93E-01	2.99E-03	3.26E+01	3.72E-02
	S.D.	**1.06E-20**	2.00E-01	8.92E-04	7.04E+00	4.82E-02
$f_5(x)$	Avg.	1.20E+01	**1.63E+01**	5.30E-07	1.19E+01	1.51E+01
	S.D.	**1.14E+00**	2.76E+00	5.52E-07	6.98E-01	5.80E+00
$f_6(x)$	Avg.	**9.98E-94**	3.73E-17	3.01E-25	1.18E-10	9.42E-24
	S.D.	**1.60E-93**	6.03E-17	2.89E-25	1.18E-10	1.63E-23
$f_7(x)$	Avg.	5.47E+03	5.13E+03	7.22E+03	**4.08E+03**	4.52E+03
	S.D.	3.14E+02	2.21E+02	3.25E+02	3.52E+02	**1.88E+02**
$f_8(x)$	Avg.	**5.34E-24**	4.92E+01	1.63E-06	6.86E+01	3.95E-11
	S.D.	**9.22E-24**	4.42E+01	1.96E-06	4.96E+00	6.07E-11
$f_9(x)$	Avg.	**0.00E+00**	1.65E+01	1.66E+02	5.71E-06	3.19E+01
	S.D.	**0.00E+00**	1.07E+01	1.05E+01	5.46E-06	5.06E+00
$f_{10}(x)$	Avg.	3.80E+00	5.06E-02	2.60E+00	2.66E-01	**1.72E-02**
	S.D.	5.34E+00	2.17E-02	5.80E+00	3.67E-01	**1.76E-02**
$f_{11}(x)$	Avg.	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.10E-11	**0.00E+00**
	S.D.	**0.00E+00**	**0.00E+00**	**0.00E+00**	7.40E-12	**0.00E+00**
$f_{12}(x)$	Avg.	**-1.26E+04**	**-1.26E+04**	**-1.26E+04**	**-1.26E+04**	-1.17E+04
	S.D.	1.47E-11	1.86E-05	**0.00E+00**	1.58E-12	1.99E+02
$f_{13}(x)$	Avg.	**0.00E+00**	2.31E+01	1.64E+02	3.44E-06	3.33E+01
	S.D.	**0.00E+00**	1.30E+01	8.78E+00	2.37E-06	4.99E+00
$f_{14}(x)$	Avg.	**0.00E+00**	3.47E-58	1.74E-86	5.11E-47	8.08E-67
	S.D.	**0.00E+00**	7.75E-58	3.80E-86	8.83E-47	1.81E-66
$f_{15}(x)$	Avg.	**-1.87E+02**	**-1.87E+02**	**-1.87E+02**	**-1.87E+02**	**-1.87E+02**
	S.D.	**0.00E+00**	4.02E-14	**0.00E+00**	**0.00E+00**	2.20E-07
$f_{16}(x)$	Avg.	**4.83E-21**	1.99E-03	3.92E-15	2.09E-06	2.64E-10
	S.D.	**9.38E-21**	1.04E-03	1.81E-15	7.44E-07	2.54E-10
$f_{17}(x)$	Avg.	**-1.00E+00**	**-1.00E+00**	**-1.00E+00**	**-1.00E+00**	**-1.00E+00**
	S.D.	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
$f_{18}(x)$	Avg.	1.79E-02	3.02E-20	**5.13E-31**	7.24E-15	3.25E-19
	S.D.	4.00E-02	4.60E-20	**2.90E-31**	3.22E-15	6.61E-19
$f_{19}(x)$	Avg.	**2.52E-105**	1.13E-21	1.39E-30	1.83E-14	1.44E-29
	S.D.	**3.58E-105**	1.05E-21	1.44E-30	1.02E-14	3.21E-29
$f_{20}(x)$	Avg.	**0.00E+00**	2.28E-05	2.13E-13	3.18E-04	1.88E-08
	S.D.	**0.00E+00**	3.17E-05	1.31E-13	6.10E-05	2.46E-08

At present, Monte Carlo is a common method, but its implementation is a complex and time-consuming task [15], and the more important point we have noticed is that it cannot also meet the real-time requirements. It is high time to improve the sensing time by using the improved Differential Evolution algorithm, such as good searching performance, fewer parameters, and faster convergence rate, so as to maximize the system throughput.

5.1. Mathematical Model Based on Energy Detection. Without interfering with the authorized users, it is necessary to determine if they are busy or idle by continuously detecting signals received in a certain frequency band, so as to determine whether these resources are accessible or not, which is a binary hypothesis testing problem, we assume that the authorized user accesses his reserved frequency for periodic T_1, and its disappearance takes

Table 3: The comparison results of function optimization.

F#/Rate	T/F	EPSDE	MDE_DE	CODE_DE◦	MDESELF_DE
F1/87.5%	T-test	7.6954E-03	3.3359E-02	3.5105E-02	3.4813E-02
	F-test	8.8978E-01	5.4921E-03	3.2283E-05	3.6439E-03
	T/F	+/+	+/+	+/+	+/+
F2/62.5%	T-test	2.2431E-02	3.3328E-06	3.1399E-04	5.8249E-02
	F-test	5.4126E-01	5.2737E-01	4.2237E-03	1.3638E-02
	T/F	+/-	+/-	+/+	-/+
F3/75%	T-test	1.9592E-01	2.8839E-02	5.5096E-03	4.2232E-01
	F-test	1.5253E-33	4.9760E-23	2.8585E-31	5.6612E-24
	T/F	-/+	+/+	+/+	-/+
F4/87.5%	T-test	4.4275E-02	6.7979E-04	2.8489E-03	9.0376E-02
	F-test	4.7045E-40	3.0269E-34	6.7452E-43	7.9601E-40
	T/F	+/+	+/+	+/+	-/+
F5/50%	T-test	1.6634E-02	9.8894E-06	4.9342E-01	1.4166E-01
	F-test	1.1582E-01	3.2658E-25	3.6254E-01	8.1762E-03
	T/F	+/-	+/+	-/-	-/+
F6/62.5%	T-test	9.0588E-02	6.7503E-02	6.7318E-03	2.1129E-01
	F-test	3.0342E-147	1.2115E-130	2.8959E-159	1.7384E-146
	T/F	-/+	-/+	+/+	-/+
F7/50%	T-test	8.0260E-03	5.0414E-04	8.2932E-04	1.4593E-02
	F-test	2.9739E-01	2.9593E-01	6.3363E-01	1.9485E-01
	T/F	+/-	+/-	+/-	+/-
F8/87.5%	T-test	1.2310E-02	7.9426E-03	7.5570E-05	7.5386E-02
	F-test	1.1154E-74	4.4393E-52	2.0763E-73	6.1861E-42
	T/F	+/+	+/+	+/+	-/+
F9/100%	T-test	2.3766E-02	4.7325E-08	1.1628E-02	4.0496E-07
	F-test	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	T/F	+/+	+/+	+/+	+/+
F10/37.5%	T-test	1.4552E-01	6.3556E-01	1.7863E-01	1.5229E-01
	F-test	1.6245E-09	8.7781E-01	1.3244E-04	7.0381E-10
	T/F	-/+	-/-	-/+	-/+
F11/100%	T-test	0.0000E+00	0.0000E+00	3.9290E-02	0.0000E+00
	F-test	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	T/F	+/+	+/+	+/+	+/+
F12/100%	T-test	0.0000E+00	0.0000E+00	0.0000E+00	6.8809E-06
	F-test	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	T/F	+/+	+/+	+/+	+/+
F13/100%	T-test	2.3766E-02	4.7325E-08	2.2266E-02	1.6671E-05
	F-test	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	T/F	+/+	+/+	+/+	+/+
F14/50.0%	T-test	1.2857E-01	2.9792E-01	6.8323E-02	4.2411E-01
	F-test	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	T/F	-/+	-/+	-/+	-/+
F15/100%	T-test	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	F-test	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	T/F	+/+	+/+	+/+	+/+
F16/100%	T-test	1.7538E-02	2.3093E-03	2.3093E-03	3.9836E-02
	F-test	2.2333E-51	6.1220E-16	6.2159E-42	2.4880E-31
	T/F	+/+	+/+	+/+	+/+
F17/100%	T-test	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	F-test	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	T/F	+/+	+/+	+/+	+/+

Table 3: Continued.

F#/Rate	T/F	EPSDE	MDE_DE	CODE_DE◦	MDESELF_DE
F18/50.0%	T-test	3.5592E-01	3.5592E-01	3.4659E-01	3.4659E-01
	F-test	1.0490E-71	1.6603E-116	2.5120E-52	4.4566E-67
	T/F	-/+	-/+	-/+	-/+
F19/100%	T-test	1.3200E-02	5.4837E-02	1.5650E-03	3.3401E-01
	F-test	0.0000E+00	5.7934E-298	0.0000E+00	1.3100E-303
	T/F	+/+	+/+	+/+	+/+
F20/100%	T-test	4.6049E-02	4.2513E-04	1.3872E-08	3.6213E-02
	F-test	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	T/F	+/+	+/+	+/+	+/+

Table 4: Comparison results of function optimization (30d).

Function	Dim	Algorithm	MEAN	STD
$f(x) = \sum_{i=1}^{N} x_i^2$	30	SFDE	0.00E+00	0.00E+00
		SaDE	0.00E+00	0.00E+00
		JADE	0.00E+00	0.00E+00
		DMPSADE	0.00E+00	0.00E+00
		IMMSADE	0.00E+00	0.00E+00
$f(x) = \sum_{i=1}^{N-1} \left(100\left(x_i^2 - x_{i+1}\right)^2 + \left(x_i - 1\right)^2\right)$	30	SFDE	5.62E+00	1.06E+00
		SaDE	4.67E+01	3.23E+01
		JADE	2.97E+00	8.17E+00
		DMPSADE	7.48E-06	2.24E-05
		IMMSADE	0.00E+00	0.00E+00
$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}\right) - \exp\frac{1}{N}\sum_{i=1}^{N}\cos\left(2\pi x_i\right) + 20 + e$	30	SFDE	5.47E+00	2.12E-12
		SaDE	2.09E+01	4.26E-03
		JADE	2.09E+01	1.67E-01
		DMPSADE	2.02E+01	1.60E-01
		IMMSADE	3.99E-15	0.00E+00

T_0 as a cycle; the energy detection model is as formula (15):

$$\begin{aligned} T_0 &: \mathrm{z}(\mathrm{n}) = \mu(\mathrm{n}) \\ T_1 &: z(\mathrm{n}) = s(\mathrm{n}) + \mu(\mathrm{n}) \end{aligned} \tag{15}$$

In the above model, $z(n)$ is the signal received by the cognitive user. $\mu(n)$ is a Gaussian noise with a mean of 0 and a variance of σ_μ^2. $s(n)$ is the signal belonged to authorized user, the mean value is 0, and the variance is σ^2.

Spectrum sensing concerns two parameters (i.e., detection probability and false alarm probability) [16]. Let τ be the duration of observation, and while the statistic index denotes as formula (16)

$$\mathrm{W} = \frac{1}{N}\sum_{n=1}^{N} |z(n)|^2 \tag{16}$$

where N is the amount of those perceived samples and f is the sampling frequency, it is obvious that $N = \tau f$. Suppose that probability density W is a random variable with distribution of χ^2 and a given threshold ε; the false alarm probability can be given by the following formula (17):

$$\begin{aligned} P_f(\varepsilon) &= P_\tau\left(\mathrm{W} > \varepsilon \mid \mathrm{T}_0\right) \\ &= \frac{1}{\sqrt{2\pi\sigma_0}}\int_N^\infty e^{-(\mathrm{W}-\mu_0)^2/2\sigma_0^2} dx \\ &= Q\left(\left(\frac{\varepsilon}{\sigma_\mu^2}\right) - \sqrt{\tau f}\right) \end{aligned} \tag{17}$$

where $P_f(\varepsilon)$ refers to the probability of misjudging an authorized user without his/her perceiving, δ_0^2 is the variance of W, and μ_0 is the mean of W, respectively.

However, assuming the given T_1 and a threshold ε, the probability density W will be a noncentral χ^2 distribution variable. Then the corresponding detection probability is shown in formula (18).

$$\begin{aligned} P_d(\varepsilon) &= P_\tau\left(\mathrm{W} > \varepsilon \mid \mathrm{T}_1\right) \\ &= \mathrm{Q}\left(\left(\frac{\varepsilon}{\sigma_\mu^2} - \gamma - 1\right)\sqrt{\frac{\tau f}{2\gamma + 1}}\right) \end{aligned} \tag{18}$$

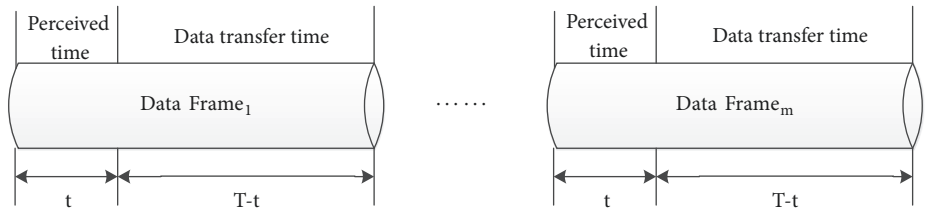


Figure 6: Period-aware frame structure.

where $P_d(\varepsilon)$ refers to the probability of correctly judging the existence of authorization and γ is the signal-to-noise ratio from cognitive users to authorized users. From formulas (17) and (18) combined, we infer that formulas (19) and (20) are correct.

$$P_f = Q\left(\sqrt{2\gamma+1}\mathrm{Q}^{-1}\left(\overline{P_d}\right)+\sqrt{\tau f\gamma}\right) \tag{19}$$

$$P_d = Q\left(\frac{1}{\sqrt{2\gamma+1}}\left(\mathrm{Q}^{-1}\left(\overline{P_f}\right)-\sqrt{\tau f\gamma}\right)\right) \tag{20}$$

When the timely requirement of an authorized user is detected, the related cognitive user needs to quickly give a way to the authorized user. Based on the above formulas (19) and (20), it is obvious that the higher the value of P_d is, the better the related protection is obtained. The lower the P_f is, the higher the reusability of wireless resource is. So a good algorithm should have high P_d and low P_f.

5.2. A Trade-Off between User Throughput and Perceived Time. The time cost of processing one data frame of cognitive radio includes two parts, i.e., frame perceived phase and data transmission phase. When the first perception becomes longer, the detection probability becomes larger, the probability of false alarm becomes smaller, but the cognitive user's data transmission time will be reduced [17]. Therefore, how to balance the sensing and transmission time under the precondition of authorizing users to operate safely and then improve the throughput of the cognitive radio system is an urgent problem to be solved. In this section, the periodic structure of the radio frame is the basis for research, as shown in Figure 6.

When the authorized user is not in the channel, there is no false alarm probability [18]; the cognitive user's throughput is $\mathrm{C}_0 = \log 2(1 + SNR_s)$; if an authorized user exists, the cognitive user throughput is $\mathrm{C}_1 = \log 2(1+SNR_s/(1+SNR_p))$. Obviously $\mathrm{C}_0 > \mathrm{C}_1$, and $SNRs = Ps/N_0$, where Ps is the average cognitive user power and N_0 is the noise power. Then $SNRp = P_p/N_0$, where P_p is the interference degree of the authorized user at the cognitive user acceptance point [19]. It can be found that, in order to protect authorized users, the probability of $\overline{P_d}$ is close to 1. At this time the system throughput is as formula (21):

$$R(\tau) = R_0(\varepsilon,\tau) + \mathrm{R}_1(\varepsilon,\tau) \tag{21}$$

Among them, $R_0(\varepsilon,\tau) = ((T-\tau)/T)C_0(1-P_f(\varepsilon,\tau))P(\mathrm{H}_0)$ $R_1(\varepsilon,\tau) = ((T-\tau)/T)C_1(1-P_d(\varepsilon,\tau))P(\mathrm{H}_1)$, $P(H_0)$ and $P(H_1)$ denote the probability that the first user will not appear and appear, respectively, and $P(H_0)+P(H_1) = 1$. Since $\mathrm{C}_0 > \mathrm{C}_1$ and $\overline{P_d}$ approximately equal 1, R_1 is close to zero. Then formula (21) can simplify the following formula (22):

$$R(\tau) \approx R_0(\varepsilon,\tau) = \frac{T-\tau}{T}C_0\left(1-P_f(\varepsilon,\tau)\right)P(\mathrm{H}_0) \tag{22}$$

Since the Q is a decreasing function, when the sensing time becomes larger, under the specified detection probability, the false alarm probability will be small, and the data transmission time will be smaller. That is to say, the value of formula (22) may become larger or smaller. The trade-off between perceived user throughput and perceived time is to find the optimal sensing time τ under the protection of the authorized user, so that the system arrives at the maximum effective throughput. Therefore, the optimization problem of looking for the optimal perception time is transformed into formula (23):

$$\begin{aligned} \max:\quad & \{R(\tau) = R_0(\varepsilon,\tau)\} \\ s.t\quad & P_d(\varepsilon,\tau) \geq \overline{P_d} \end{aligned} \tag{23}$$

To solve problem of the effective throughput of spectrum resource in wireless cognition network, we need to find the best service mode for the personalized users. This is a problem of function optimization. This section uses the SFDE to find the optimal perception time.

5.3. Specific Solution Based on SFDE. Specific implementation steps are as follows, which is based on SFDE.

Step (1) initializes a population. The entire range of variables is from 0.001 to 1 into 45 intervals. The upper limit of the *i-th* segment is *bounds* $(i, 2)$ and the lower bound is limited to bounds $(i, 1)$. The specific evaluation process is shown as formula (24):

$$\begin{aligned} bounds(\mathrm{i},2) &= 0.01 + 0.0002 * \mathrm{i} \\ bounds(\mathrm{i},1) &= 0.01 + 0.0002 * (\mathrm{i}-1) \end{aligned} \tag{24}$$

where each interval assumes as an individual and the whole has 45 individual vectors. The whole population divide themselves into the three subpopulations, each of which has 15 individuals. Then each individual is initialized as shown in formula (25):

$$x_{j,i}(0) = bounds(\mathrm{i},1) + 0.0002 * rand(0,1) \tag{25}$$

Table 5: The parameters used in the simulation experiment.

Parameter	value	Description
NP	45	Population size.
D	20	Population individual dimension.
$iMax$	20	The maximum number of iterations.
W	5	The maximum number of individuals that the archive set holds.
N	15	The number of individuals per sub population.
P	0.1	The step size of the parameters of the parameter pool.
$P(H0)$	0.2	The probability of occurrence of cognitive users.
$\overline{P_d}$	0.95	Detection of non-value of probability.
F	6MHz	Sampling frequency.
$SNRp$	-15dB	Authorized user's SNR.
$SNRs$	20dB	Cognitive user's SNR

where the 0.0002 is the length of itself individual gene in the population.

Step (2) finds the optimal individual X_{lbest} in each subpopulation. Each gene vector is brought into fitness formula (22). The X_{lbest} is based on the fitness and the maximum value of each gene for each individual. With randomly selecting the parameters F and CR for each individual from the parameter candidate pool, we can get the sniffing values F_ad and F_de of the parameter F corresponding to each individual are obtained and use the optimal individual to compute the variation vector.

Step (3) chooses sniffing values CR_ad and CR_de of the parameter CR corresponding to each individual. And we can combine the intermediate vector with the mutation vector of Step (2).

Step (4) lets the intermediate vector obtained in Step (3) participate in the selection operation, and we can compute the fitness values for each gene in each intermediate vector by formula (22) and compare the fitness value with the target vector. If the fitness is greater than the target vector, the gene will replace the gene corresponding to the target individual. Repeat Steps (2)–(4) until all individuals in the population have completed the operation of mutation, crossover, and selection. Then the new population goes into the next generation.

Step (5) repeats Steps (2)–(4) until the maximum number of iterations is reached. The optimal sensing time and the corresponding maximum effective throughput are obtained.

5.4. Parameters Setting. In order to prove the effectiveness of the SFDE algorithm in the process of system effective throughput optimization, the simulation carries out and compares with the Monte Carlo method. In order to fully reflect the superiority and contrast integrity of the improved SFDE, the experiment will contrast those brand-new variants, such as JADE, MDE, EPSDE, and CODE. In addition, this experiment will also join the standard particle swarm algorithm to illustrate the Differential Evolution more effectively. The simulation in the hardware configuration for the INTER CORE i7, CPU 3.19GHz, memory 1.98GB running on the computer, the program written to run the software for the MATLAB 7.0. The parameters used in the system simulation experiment are shown in Table 5.

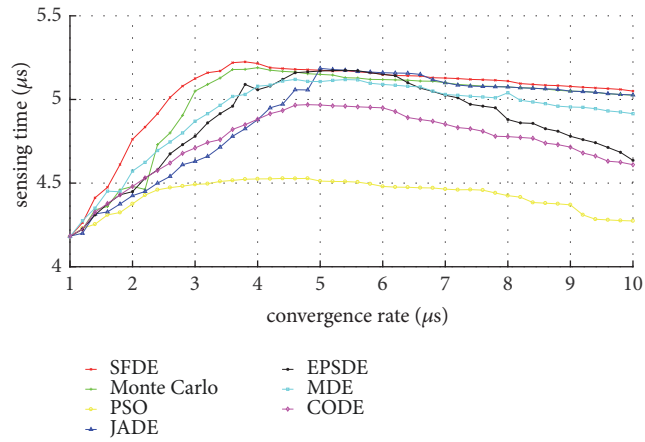


Figure 7: Comparison of SFDE and Monte Carlo for system effective throughput.

The convergence curve obtained by the simulation is shown in Figure 7. From the effective throughput convergence charts obtained in Figure 5, it can be seen that the optimal sensing time of the SFDE is 3.67ms and 3.7ms for Monte Carlo. The maximum effective throughput of SFDE is 5.2231bps/Hz, while Monte Carlo's throughput is 5.1947bps/Hz. The SFDE achieves better goat than the Monte Carlo, which is close to the theoretical value of 5.3bps/Hz. For PSO, it is the maximum throughput with 4.5126bps/Hz, much worse than that of SFDE. JADE is second only to Monte Carlo, but its corresponding sensing time is 5.67ms, which is significantly worse. In summary, SFDE in the throughput performance is very good.

It can be seen from the experimental results in Figure 8 that the convergence rate of SFDE in the optimization process is much faster than that of others, especially Monte Carlo. The SFDE spends only 2.1987s on finding the best sensing time, while Monte Carlo spends 52.1423s. So the sensing time of SFDE is 23.7 times faster than Monte Carlo. It is even 5.69 times faster than the EPSDE. Monte Carlo is not dominant in

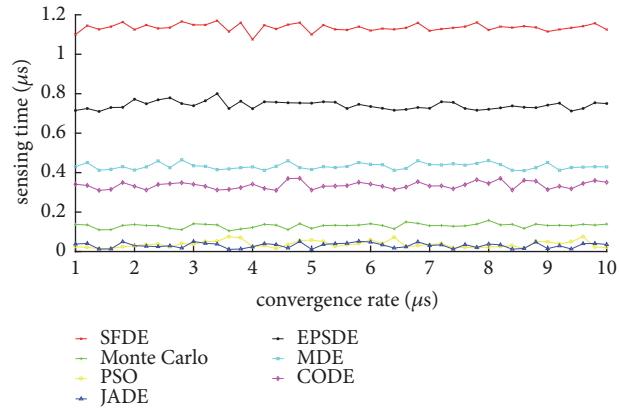


Figure 8: Comparison of SFDE and Monte Carlo convergence rate.

the selection of sensing time; that is, it is not suited to real-time.

From the above experimental results, SFDE increases the system throughput with ensuring the convergence of the premise. Eventually SFDE greatly improve the convergence rate. In other words, SFDE overcomes the shortcomings of Monte Carlo and others that are not suitable for real-time, which makes it more practical in solving the effective throughput optimization of cognitive users.

6. Conclusion

In this paper, in order to optimize the radio time perception, so that the system throughput can be maximized, a Shuffled Frog-leaping Differential Evolution (SFDE) algorithm is put forward, which combines shuffled frog leaping with differential evolution. The experimental studies show that in the same situation the proposed SFDE algorithm improves the existing performance of others. This article simulates the effective throughput of cognitive users in radio and uses the improved SFDE to optimize. Experimental results show that, under the same situation, the SFDE can reach the theoretical value of throughput optimization accuracy, and the convergence speed is also faster.

Data Availability

The experimental Excel data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

Financial supports from the National Natural Science Foundation of China (no. 61572074), the China Scholarship Council for visiting to UK (Grant no. 201706465028), and the 2012 Ladder Plan Project of Beijing Key Lab of Knowledge Engineering for Materials Science (no. Z12110 1002812005) are highly appreciated.

References

[1] Q. Fan and X. Yan, "Self-adaptive differential evolution algorithm with discrete mutation control parameters," *Expert Systems with Applications*, vol. 42, no. 3, pp. 1551–1572, 2015.

[2] P. Ochoa, O. Castillo, and J. Soria, "A fuzzy differential evolution method with dynamic adaptation of parameters for the optimization of fuzzy controllers," in *Proceedings of the 2014 IEEE Conference on Norbert Wiener in the 21st Century (21CW)*, pp. 1–6, Boston, Mass, USA, June 2014.

[3] W.-J. Yu, M. Shen, W.-N. Chen et al., "Differential evolution with two-level parameter adaptation," *IEEE Transactions on Cybernetics*, vol. 44, no. 7, pp. 1080–1099, 2014.

[4] J. Liang, B. Qu, X. Mao, and T. Chen, "Differential evolution based on fitness euclidean-distance ratio for multimodal optimization," in *Emerging Intelligent Computing Technology and Applications*, pp. 252–260, Springer, Berlin, Germany, 2012.

[5] S. Wang, Y. Duan, W. Shu, D. Xie, Y. Hu, and Z. Guo, "Differential evolution with elite mutation strategy," *Journal of Computational Information Systems*, vol. 9, no. 3, pp. 855–862, 2013.

[6] M. Ali, M. Pant, and A. Abraham, "Improving differential evolution algorithm by synergizing different improvement mechanisms," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 7, no. 2, pp. 1–32, 2012.

[7] S.-W. Wang, W.-S. Zhang, L.-X. Ding et al., "Research on parameter self-selection strategy of differential evolution," *Computer Science*, vol. 42, no. 1, pp. 256–259, 2015.

[8] Z.-W. Li, X.-G. Zhou, G.-J. Zhang et al., "Dynamic adaptive differential evolution algorithm," *Computer Science*, vol. 42, no. z1, pp. 52–56, 2015.

[9] M. M. Eusuff and K. E. Lansey, "Optimization of water distribution network design using the shuffled frog leaping algorithm," *Journal of Water Resources Planning and Management*, vol. 129, no. 3, pp. 210–225, 2003.

[10] J. Luo and M.-R. Chen, "Improved shuffled Frog Leaping algorithm and its multi-phase model for multi-depot vehicle routing problem," *Expert Systems with Applications*, vol. 41, no. 5, pp. 2535–2545, 2014.

[11] Q. Zhang, L. Liu, and H. Guo, "Improved shuffled flog leaping algorithm based on keeping the diversity of population," *Journal of Thermal Analysis and Calorimetry*, vol. 105, no. 1, pp. 53–59, 2009.

[12] H. Li, "Crossing and variation frog leaping algorithm," *Journal of Ludong University (Natural Science Edition)*, no. 1, pp. 16–20, 2015.

[13] S. Wang, Y. Li, and H. Yang, "Self-adaptive differential evolution algorithm with improved mutation mode," *Applied Intelligence*, vol. 47, no. 3, pp. 644–658, 2017.

[14] Z. Quan, S. Cui, A. H. Sayed, and H. V. Poor, "Wideband spectrum sensing in cognitive radio networks," in *Proceedings of the IEEE International Conference on Communications (ICC '08)*, pp. 901–906, Beijing, China, May 2008.

[15] E. C. Y. Peh, Y. Liang, Y. L. Guan, and Y. Zeng, "Optimization of cooperative sensing in cognitive radio networks: a sensing-throughput tradeoff view," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 9, pp. 5294–5299, 2009.

[16] X. Zhang, H. Bie, Q. Ye, C. Lei, and X. Tang, "dual-mode index modulation aided OFDM with constellation power allocation and low-complexity detector design," *IEEE Access*, vol. 5, pp. 23871–23880, 2017.

[17] J. Svenson and T. Santner, "Multiobjective optimization of expensive-to-evaluate deterministic computer simulator models," *Computational Statistics & Data Analysis*, vol. 94, pp. 250–264, 2016.

[18] J. Zhou, E. Dutkiewicz, R. P. Liu, X. Huang, G. Fang, and Y. Liu, "A modified shuffled frog leaping algorithm for PAPR reduction in OFDM systems," *IEEE Transactions on Broadcasting*, vol. 61, no. 4, pp. 698–709, 2015.

[19] Y. Yu, P. Zhang, Z. Song, and F. Chai, "Composite differential evolution algorithm for SHM with low carrier ratio," *IET Power Electronics*, vol. 11, no. 6, pp. 1101–1109, 2018.