The Journal of Supercomputing (2021) 77:5726–5756
https://doi.org/10.1007/s11227-020-03482-w

A novel mutation strategy selection mechanism for differential evolution based on local fitness landscape

Zhiping Tan[1] · Kangshun Li[1] · Yuan Tian[2] · Najla Al-Nabhan[3]

Accepted: 20 October 2020 / Published online: 10 November 2020

Abstract
The performance of differential evolution (DE) algorithm highly depends on the selection of mutation strategy. However, there are six commonly used mutation strategies in DE. Therefore, it is a challenging task to choose an appropriate mutation strategy for a specific optimization problem. For a better tackle this problem, in this paper, a novel DE algorithm based on local fitness landscape called LFLDE is proposed, in which the local fitness landscape information of the problem is investigated to guide the selection of the mutation strategy for each given problem at each generation. In addition, a novel control parameter adaptive mechanism is used to improve the proposed algorithm. In the experiments, a total of 29 test functions originated from CEC2017 single-objective test function suite which are utilized to evaluate the performance of the proposed algorithm. The Wilcoxon rank-sum test and Friedman rank test results reveal that the performance of the proposed algorithm is better than the other five representative DE algorithms.

Keywords Adaptive mutation strategy · Local fitness landscape · Differential evolution · Parameter adaptation

1 Introduction

Optimization is an important technique in real-world life, which is widely used in engineering domain, agriculture and industry [1]. In general, an unconstrained optimization problem is identified as a single-objective global optimization which

✉ Kangshun Li
likangshun@sina.com

Zhiping Tan
tzp2008ok@163.com

[1] College of Mathematics and Informatics, South China Agricultural University, Guangzhou 510642, China

[2] School of Computer Engineering, Nanjing Institute of Technology, Nanjing 210024, China

[3] Computer Science, King Saud University, Riyadh, Kingdom of Saudi Arabia

can be described as follows [2]. We need to search for a decision variables of vector $\vec{x} = \{x_1, x_2, ..., x_D\}$, which satisfies the variable bound, $x_{j,low} \leq x \leq x_{j,upp}$ and minimizes or maximizes a fitness function $f(\vec{x})$, where D denotes the dimension of the solution space, $x_{j,low}$ and $x_{j,upp}$ is the lower and upper bounds, respectively, $j = 1,2,\ldots,D$. In practical problems, the decision variable can be discrete, continuous or hybrid and the objective function can be uni-modal or multi-modal, hybrid and composition. Therefore, the single-objective global optimization problems have various different characteristics and complicated mathematical proprieties, which make the algorithms difficult to get the desired results [3].

As exhaustive search-based methods are time expensive when solving complex optimization problems. Evolutionary algorithms (EAs) have been widely used in recently years [4]. EAs are population-based search methods that have been proved to be a promising approach to find the best solution in effective time. They are some reasons for explanation: (1) they are convenient for computers to carry out parallel computation; (2) the framework of algorithms is simple; and (3) the parameters of the algorithms are few. Nevertheless, on account of EAs are stochastic algorithms, it cannot be ensured that they will obtain the desired results. In addition, the performance of EAs is sensitive to parameter settings and the selection of mutation strategy.

DE as a classic evolutionary algorithm has been applied to engineering design, neural network and artificial intelligence [5]. However, when facing complex optimization problems, it is difficult for the DE to get satisfactory results. One reason is that the performance of DE algorithm depends on the mathematical properties of optimization problems.

As a consequence, till now many researches utilized the standard DE algorithm or some DE variants to solve optimization problems [6]. The improvements of the DE algorithm mainly focus on the aspects including self-adaptive control parameters, multi-mutation strategies (use more than a single mutation strategy) and ensemble based (adaptive control parameters and mutation strategy), whereas how to combine these mutation strategies and control parameters in a more appropriate way is still a difficult task. The mutation strategy of DE greatly affects the performance of the algorithm. In most DE algorithms, the selection of the mutation strategy relies on the experience in most cases. Sometimes, some criteria are used to choose the suitable mutation strategy, such as the best performing one in the previous generation, roulette selection method and machine learning. However, all the methods are concerned about the optimization algorithms, the characteristics of the optimization problem are not considered in the design of algorithm. Fitness landscape can be used to analyze an optimization problem and judge the complexity of the optimization problem, because the calculation of fitness landscape is complex. Therefore, the selection of the mutation strategy in the DE according to the fitness landscape is rare, even though it may be possible to improve the performance of an algorithm. However, for these algorithms that do exist, they have some limitations: (1) the fitness landscape was used to analyze the static optimization problems. However, the optimization problems are dynamic problems during the evolution process; (2) the time consume is very expensive when calculating the fitness landscape metrics; and (3) a pre-training and testing mechanism are used, which indicates the fitness landscape only utilized to judge the considered test problems, and the improved algorithm may not be suitable for another optimization problems. It may be a meaningful study to use fitness landscape information to guide the selection of mutation strategy for every problem at each generation.

In this paper, an improved DE algorithm is proposed, in which a problem's local fitness landscape is considered, so that more guidance is given to select the favored mutation strategy during the evolutionary process. A novel adaptive control parameter method is adopted in the algorithm. Furthermore, a linear population size reduction mechanism is considered, in which population size is decreased along with the number of generation. To evaluate the performance of the proposed algorithm, a total of 29 benchmark test functions derived from CEC2017 competition are used in the experiments. These benchmark test functions have three different kinds of mathematical properties. The experimental results indicate that the proposed algorithm is superior to other five classic DE algorithms.

The rest of the paper is organized as follows. In Sect. 2, we review recent advances in DE algorithms. Then, the LFLDE algorithm is described in Sect. 3. In Sect. 4, the experimental results are analyzed and discussed. Some conclusions and future work are given in Sect. 5.

2 Related work

In this section, the standard DE algorithm and some improved DE variants will be discussed.

2.1 DE algorithm

The structure of standard DE consists of mutation, crossover and selection operators, which was conceived by Storn and Price in 1997. DE has fewer control parameters and desired performance which becomes a popular EA. Assume that a population is made up of NP individuals. Each individual is a *D*-dimensional vector. The initial population can be expressed as [7]:

$$x_{i,G} = \left\{ x_{i,G}^1, x_{i,G}^2, \ldots, x_{i,G}^D \right\}, \quad i = 1, \ldots, NP \tag{1}$$

where *G* represents the generation. Each dimension of the individual is constrained by:

$$x_{\min} = \left\{ x_{\min}^1, \ldots, x_{\min}^D \right\} \; and \; x_{\max} = \left\{ x_{\max}^1, \ldots, x_{\max}^D \right\} \tag{2}$$

Generally, the initial value of the *i*th individual of the *j*th component is created by:

$$x_{i,0}^j = x_{\min}^j + \text{rand}(0,\ 1) * \left(x_{\max}^j - x_{\min}^j \right) \tag{3}$$

where rand(0, 1) denotes a uniformly distributed random variable within the range [0,1].

2.1.1 Mutation operation

After initialization, the mutation operator is used to generate the mutant vectors $v_{i,G}$ with respect to each individual $x_{i,G}$. There are six most frequently used mutation strategies [8]:

DE/rand/1

$$v_{i,G} = x_{r1,G} + F * \left(x_{r2,G} - x_{r3,G}\right) \tag{4}$$

DE/rand/2

$$v_{i,G} = x_{r1,G} + F * \left(x_{r2,G} - x_{r3,G}\right) + F * \left(x_{r4,G} - x_{r5,G}\right) \tag{5}$$

DE/best/1

$$v_{i,G} = x_{\mathrm{best},G} + F * \left(x_{r1,G} - x_{r2,G}\right) \tag{6}$$

DE/best/2

$$v_{i,G} = x_{\mathrm{best},G} + F * \left(x_{r1,G} - x_{r2,G}\right) + F * \left(x_{r3,G} - x_{r4,G}\right) \tag{7}$$

DE/current-to-rand/1

$$v_{i,G} = x_{i,G} + F * \left(x_{r1,G} - x_{i,G}\right) + F * \left(x_{r2,G} - x_{r3,G}\right) \tag{8}$$

DE/current-to-best/1

$$v_{i,G} = x_{i,G} + F * \left(x_{\mathrm{best},G} - x_{i,G}\right) + F * \left(x_{r1,G} - x_{r2,G}\right) \tag{9}$$

The indices $x_{r1,G}$, $x_{r2,G}$, $x_{r3,G}$, $x_{r4,G}$ and $x_{r5,G}$ are mutually exclusive integers randomly chosen from the current population, which are different from the index i. The scale factor F is a positive control parameter. $x_{\mathrm{best},G}$ is the global optimal individual at generation G.

2.1.2 Crossover operation

After the step of the mutation, the crossover operator is adopted to chose the trial vector $u_{i,G}$ between $v_{i,G}$ and $x_{i,G}$, which can be expressed by the formula below:

$$u^j_{i,G} = \begin{cases} v^j_{i,G}, & \text{if } \left(\mathrm{rand}_j \le \mathrm{CR} \text{ or } j = n_j\right) \\ x^j_{i,G}, & \text{otherwise} \end{cases} \quad j = 1, 2, \ldots, D \tag{10}$$

where $\mathrm{rand_j}$ is a random number, which generated within the range [0,1]. The control parameter of crossover rate CR is a preset constant within the range from 0 to 1, and n_j is a random integer produced within the range [0,1].

2.1.3 Selection operation

After crossover operation, the values of the trial vectors should be checked whether they exceed the constrained bounds. We should reinitialize them within the predefined scope, after obtaining the final trial vectors. In the next, a selection operation is executed. If the trial vector has less or equal objective function value (for minimization problem) compared to the corresponding target vector, the trial vector will be chosen into the population in the next generation. Otherwise, the target vector will still retain in the population for the next generation. The greedy selection operation is performed as following:

$$x_{i,G+1} = \begin{cases} u_{i,G}, & \text{if } f(u_{i,G}) \leq f(x_{i,G}) \\ x_{i,G}, & \text{otherwise} \end{cases} \tag{11}$$

2.2 DE variant algorithms

Recently, a plenty of DE algorithms for dealing with single-objective global optimization problems have been proposed, and some recently proposed variant DE algorithms are discussed in this section. Differential evolution is a population-based algorithm, which is suitable for numerical optimization, the excellent performance had been confirmed, and it has been applied to various fields [9]. The classic DE has three vital control parameters which are scaling factor *F*, crossover rate CR and population size NP. The performance of DE largely depends on the three control parameters; moreover, the three control parameters are fixed and it is very sensitive to different optimization problems; it requires to be adjusted constantly by a user when solving different problems, and it could be a difficult task to find appropriate values for them [10]. To address this problem, a large amount of adaptive and self-adaptive DE variants had been reported. Islam et al. presented an adaptive DE (ADE), in which the parameters *F* and CR are updated according to the successful values [11]. A self-adaptive (jDE) method to renewal the *F* and CR values of the DE algorithm was presented by Breast et al. In jDE, each individual will be assigned a parameter value *F* and CR. Each parameter is randomly generated with some probability, and the new generated parameters are reversed for the next generation only when the corresponding trial individual is selected as the new individual of the population [12]. Zhang et al. proposed an adaptive DE algorithm with an external archives, and "DE/current-to-*p*best" mutation strategy. In addition, the control parameters CR and *F* of each individual are automatically updated according to the individuals survive to the next generation [13]. An improved version of JADE called SHADE was proposed by Tanabe and Fukunaga which uses a historical record of successful parameter setting [14]. Before long, Tanabe and Fukunaga proposed a L-SHADE, where a linear population size reduction (LPSR) strategy is applied to SHADE. The population linearly decreases along with the number of iterations increases. L-SHADE exhibited good performance, in comparison with the state-of-the-art DE algorithm on CEC2014 benchmark test function set [15]. Subsequently, a plenty of improved version of L-SHADE have been placed on top ranks at CEC competitions, such as JSO [16], LSHADE44 [17], LSHADE EpSin [18], LSHADE-RSP [19] and so on.

To better tackle various different optimization problems, the improvement of DE is not only focused on improving the three main control parameters, but also proposing different kinds of mutation selection strategies. Here, we will review some of them.

A self-adaptive DE (SaDE) was proposed by Qin et al. In SaDE, at first, four mutation strategies are stored in a strategy pool; each strategy is randomly selected for every individual at each generation. Before long, some promising strategies are picked up into strategy pool according to the success rate of the trial vector during the evolutionary [20]. Wang et al. proposed a composite DE (CoDE), in which three mutation operators are adopted. In each generation, the new trial vector derives from the best one which generates by three mutation strategies at the same time. The obtained results indicated that CoDE is an effective optimization algorithm for solving the CEC2005 test functions [21]. A self-adaptive multi-operator-based differential evolution (SAMO-DE) was conceived by Elsayed et al. In SAMO-DE, each search operators has its own sub-population. In each generation, the new selected operator relies on the reproductive success of the search operators. The experimental results showed that SAMO-DE was superior to other DE algorithms [22]. Mallipeddi et al. presented an ensemble differential evolution algorithm (EPSDE). In EPSDE, some strategy pools are designed to store the mutation operators and the control parameters. Besides, it establishes the relationship between mutation strategies and control parameters during the evolutionary process [23]. All of the above-mentioned algorithms did not try to use the fitness landscape metrics at the mutation strategy selection phase, so fitness landscape analysis is necessary to be taken into account for guidance of algorithm design.

3 Local fitness landscape-based adaptive mutation strategy differential evolution

This section proposes the framework of the LFLDE algorithm, in which the mutation strategy selection method is presented, and a novel control parameters adaptive strategy is adopted. Besides, population linear reduction is also used.

3.1 Local fitness landscape

The fitness landscape has been proven effective for analyzing evolutionary algorithms [24]. A fitness landscape is composed of fitness function values of all the individuals in the search space. Usually, fitness landscape is a static description of a problem. The difficulty of a problem is often measured according to the characteristics of the landscape [25]. Several landscape metrics have been developed for analyzing and evaluating the different characteristics of problems, such as the fitness distance correlation [26], information entropy measure [27], fitness cloud and length scale [24]. In this paper, a local fitness landscape method is utilized. Firstly, for a population (x_1, x_2,…, x_μ), find out the best individual among the whole population and tag it with x_{best}. Then calculate the Euclidean distance between each individual x_i ($i=1,\ldots,\mu$) and x_{best}, which can be expressed as:

$$d_i = \sum_{j=1}^{n} \left(x_{i_j} - x_{best_j} \right)^{\frac{1}{2}} \tag{12}$$

Secondly, sort the individuals according to the distance value, resulting in the following in ascending order: k_1, k_2,…, $k_{\mu.}$ After that, calculate the local fitness landscape evaluation metric of the individual. The measure metric value is denoted by χ. Suppose that the initial value is 0. In the end, the value will be increased to 1, if $f_{k_{i+1}} \leq f_{k_i}$. Normalize the value as:

$$\varphi = \frac{\chi}{\mu} \tag{13}$$

On account of the local fitness landscape is actual an ambiguous concept, the landscape feature φ can be deemed as the roughness of observed fitness landscape, and it can be used to judge the complexity of the optimization problem. If the value of $\varphi=0$, then local fitness landscape is more like a uni-modal landscape; it means that the optimization problem is easy to solve. On the contrary, if the value of $\varphi=1$, it suggests that the local fitness landscape is very rough, the optimal solution of the problem is hard to solve [28]. Thus, it is beneficial to choose different mutation strategies according to the landscape feature, because optimization problem is a dynamic process. Therefore, the landscape feature φ will fluctuate in each generation. To be consistent with φ, we use the dynamic mean value act as the landscape feature at each generation, which can be calculated as follows:

$$DM_{\varphi}(G) = \frac{\sum_{G=1}^{G_{\max}} \varphi(G)}{G} \tag{14}$$

where DM_{φ} is the mean feature value, and φ_1, φ_2,…, φ_G are every single value obtained from every generation.

3.2 Parameter adaptation

Numerous experiments have shown that the performance of the DE algorithm depends on the values of the control parameters (F and CR) during the evolutionary process. In generally, we use a different adaptation mechanism to generate their values. In the beginning, for each individual in the population, the control parameters CR_i and F_i are generated using a Gaussian distribution and Cauchy distribution, respectively. CR_i and F_i are calculated, respectively [13]:

$$\mathrm{CR}_{i,G} = \mathrm{rand}n_i\left(\mu_{CR},\ 0.1\right) \tag{15}$$

$$F_{i,G} = \mathrm{rand}c_i\left(\mu_F,\ 0.1\right) \tag{16}$$

where μ_F and μ_{CR} represent the mean values. At first, the standard deviations preset to a fixed constant 0.1; the mean μ_F and μ_{CR} are set to 0.5, then updated in each generation by the following formulas:

$$\mu_{\mathrm{CR}} = 0.95 - 0.05 * \mathrm{rand} \tag{17}$$

$$\mu_F = 0.65 - 0.15 * \mathrm{rand} \tag{18}$$

Besides, an adaptive p value method is adopted. A linear reduction p value for current-to-pBest/1 mutation is updated by the following formula:

$$p = \left(\frac{p_{\max} - p_{\min}}{\mathrm{FES}_{\max}}\right) \cdot \mathrm{FES} + p_{\min} \tag{19}$$

where FES is the current number of objective function evaluations, $\mathrm{FES}_{\max}$ is the maximum number of objective function evaluations, and $p_{\max}$ and $p_{\min}$ are the values that were initially defined.

3.3 Population updating method

In the meantime, to increase the diversity of the population at the early phase of the evolutionary process, while to reduce computational cost at later stages, the population size NP decreases linearly along with the evolution process is utilized. It has a large value at the beginning of the iterative process, until the population size reaches to $\mathrm{NP}_{\min}$; the corresponding formula can be defined as following [15]:

$$\mathrm{NP}_{G+1} = \mathrm{round}\left(\mathrm{NP}_{\min} - \mathrm{NP}_{\max}\right)/\mathrm{FES}_{\max} * \mathrm{FES} + \mathrm{NP}_{\max} \tag{20}$$

where $\mathrm{NP}_{\max}$ is the biggest population size at generation $G=0$.

3.4 Mutation strategy selection based on local landscape feature

According to the local fitness landscape feature, if dynamic mean value DM_φ greater than ε which indicates the local fitness landscape is very rough, the current-to-$p$best mutation strategy is used. Otherwise, the DE/best/2 mutation strategy is adopted. So, for an optimization problem, a local fitness landscape is used to judge the difficulty of the problem, then guiding the selection of the suitable mutation strategy.

$$\begin{cases} v_{i,G} = x_{i,G} + F * \left(x_{best}^{p} - x_{i,G}\right) + F * \left(x_{r1,G} - x_{r2,G}\right), & if\ \mathrm{DM}_\varphi > \varepsilon \\ v_{i,G} = x_{best} + F * \left(x_{r1,G} - x_{r2,G}\right) + F * \left(x_{r3,G} - x_{r4,G}\right) & \text{otherwise} \end{cases} \tag{21}$$

where is a predefined constant, in this paper $\varepsilon=0.5$. The pseudocode of the proposed LFLDE algorithm is presented in Table 1.

Table 1 Pseudocode of the LFLDE algorithm

```
Procedure of LFLDE
1. Set $NP_{min}$=4*$NP$; $NP_{max}$=18*$NP$; $G$=1;
2. Initialize $D$, $FES$;
3. Generate a random initial population $\{x_{i,0}^{D} \mid i = 1,2,..., NP\}$;
4. While $FES \le FES_{\max}$ do
5.         $\mu_{CR}$=0.95-0.05*$rand$;
6.         $\mu_{F}$=0.65-0.15*$rand$;
7.         $p$=0.1+0.1*$FES/FES_{\max}$;
8.         Calculate the landscape feature $\varphi$;
9.         Calculate the mean landscape feature $DM_{\varphi}$;
10.        for $i$=1 to $NP$
11.            $CR_{i,G} = randn_i(\mu_{CR}, 0.1)$; $F_{i,G} = randc_i(\mu_F, 0.1)$;
12.            Randomly choose $x_{best,G}^{p}$ as the best vectors;
13.            if $DM_{\varphi}(G) \ge \varepsilon$
14.                Generate mutant vector by current-to-$p$best;
15.            else
16.                Generate mutant vector by DE/best/2;
17.            end
18.           Generate $j_{rand} = rand\ \mathrm{int}(1, D)$;
19.           for $j$=1 to $D$
20.                if ($j = j_{rand}$ or $rand(0,1) < CR_i$)
21.                    $u_{j,i,G}=v_{j,i,G}$;
22.                else
23.                    $u_{j,i,G}=x_{j,i,G}$;
24.                end
25.            end
26.            if ($f(x_{i,G})<f(u_{i,G})$)
27.                $x_{i,G+1}=x_{i,G}$;
28.            else
29.                $x_{i,G+1}=u_{i,G}$;
30.            end
31.        end
32.      $FES \leftarrow FES+NP$; $G=G+1$;
33.      $NP \leftarrow round(NP_{min}-NP_{max})/FES_{\max}*FES+NP_{max}$;
34. end
```

4 Experimental results

4.1 Benchmark test functions and parameters setting

To evaluate the performance of the proposed LFLDE algorithm, a series of single-objective test functions originated from the CEC2017 competition are used in the experiments [29]. This test function set contains 30 a bounded single-objective functions with a diverse set of characteristics. For all test functions, the domain of definition is $[-100{,}100]^D$. All the test functions in the test suite can be divided into four types: uni-modal functions f_1–f_3; basic multimodal functions f_4–f_{10}; hybrid functions f_{11}–f_{20}; composition functions f_{21}–f_{30}. See [30] for details..

The performance of proposed algorithm is evaluated and compared with five recently proposed DE algorithms such as CoDE [21], SaDE [20], EPSDE [23], MPEDE [31] and LSHADE [15]. For the above algorithms, the parameter setting values were recommended in the cited original papers. The parameters of LFLDE are listed in pseudocode. All the population sizes are ten times the dimensions of the problem.

4.2 Algorithm complexity

All experiments were performed using MATLAB R2019a running on a laptop with Intel Core i7-7700 (3.60 GHz) CPU and 16 GB RAM running on Windows 10 system. First, the Big O notation is used to measure the computation complexity [32]. To find the optimal solutions, the proposed algorithms have an initialization stage and a subsequent stage of iterations. The computation complexity depends on *n*, *Popsize* and *Maxiter*:

$$\text{Computation complexity} = \text{O}(n \times \text{Popsize} \times \text{Maxiter}) \tag{22}$$

For LFLDE algorithm, according to Table 1, the maximum computational complexity in terms of Big O notation is:

$$\text{Computation complexity} = \text{O}(10{,}000 \times n \times n) \tag{23}$$

where Popsize is equal to 10, and the maximum number of iterations is equal to 1000. Thus, the computational complexity for all the proposed algorithms is directly proportional to the square of the input mark/channel value.

Table 2 Algorithm complexity

	T_0	T_1	T_3	$(T_3-T_1)/T_0$
$D=10$	0.07692	0.42722	0.65357	2.94426
$D=30$		0.64457	0.98876	4.47464
$D=50$		0.95681	1.82235	11.2524
$D=100$		2.56133	5.84857	42.7358

In the following, an intuitive computational complexity method is presented. Table 2 shows the time complexity of the LFLDE when solving 10, 30, 50 and 100 dimensions problem in detail. As defined in [33], T_0 is the time calculated by executing the following loop function:

$$\begin{gathered} \mathit{for}\ i = 1:1000000 \\ x = 0.55 + (\text{double})i;\ x = x + x;\ x = x/2;\ x = x + x; \\ x = \text{sqrt}(x);\ x = \log(x);\ x = \exp(x);\ x = x/(x+2); \\ \text{end} \end{gathered}$$

T_1 is the time to run 200,000 evaluations of test function f_{18} by itself with D dimensions, and T_2 is the time to execute LFLDE with 200,000 evaluations of f_{18} in D dimensions. T_3 is the mean T_2 values of 5 runs.

4.3 Algorithm performance

To evaluate the performance of algorithms, a total of 29 test functions are performed in the experiments. We utilized the solution error measure $f(x)-f(x^*)$ as the final result, and error values and standard deviations less than 10^{-8} are regarded as zero [34], where x is the best solution searched by each algorithm in each generation and x^* is the well-known global optimal solution of each test function. Every test for each function and each algorithm run 51 times independently, the condition for loop termination is set to $10{,}000 \times D$, which is the maximum number of function evaluations (FEs). The experimental results of the LFLDE carried out on the test functions with 10, 30, 50 and 100 dimensions are listed in Tables 3 and 4. It includes the gained best, worst, median, mean values and the standard deviations of error from the optimum solution of the proposed LFLDE over 51 runs for 29 benchmark functions. Note that the function f_2 was excluded from the comparison by its design which is not perfect.

The proposed algorithm LFLDE was equipped to search the optimal solutions for uni-modal functions f_1 and f_3 for all dimensions. Among multi-modal functions f_4–f_{10} only for the function f_6 and f_9 the global optimum has been achieved on 10D, 30D and 50D. For hybrid functions f_{11}–f_{20}, the 12th function f_{12} appears to be relatively difficult: this function has high error values. For composition functions f_{21}–f_{30}, the proposed algorithm appears to fall into local optima quite often; the algorithm is difficult to obtain the global optimal solution.

4.4 Statistical test and comparison to other algorithms

In this section, we will compare the performance of the LFLDE with CoDE, SaDE, EPSDE, MPEDE and LSHADE on the CEC2017 benchmark functions. The results are given in Tables 5, 6, 7 and 8 for each dimension. In these Tables, the mean and standard deviation values are shown for the six algorithms. The best results for each problem are shown in boldface, and the statistical testing is also shown in a separate column. The $+$, $-$, $\approx$ indicate whether a given algorithm performed significantly better ($+$), significantly ($-$) or not significantly different

Table 3 The results of the LFLDE algorithm for 10D and 30D

	10D					30D				
	Best	Median	Worst	Mean	Std	Best	Median	Worst	Mean	Std
f_1	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
f_3	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	5.5780E−02	5.4314E−01	1.2141E−01	1.3027E−01
f_4	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	6.3968E−03	9.731E+01	1.7008E+02	8.0970E+01	5.1078E+01
f_5	0.0000E+00	1.9452E+00	4.5213E+00	2.8614E+00	1.0261E+00	1.4937E+01	2.9728E+01	7.7738E+01	3.2148E+01	1.2130E+01
f_6	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
f_7	4.2428E+00	1.4407E+01	2.0609E+01	1.4478E+01	2.5689E+00	6.8898E+01	8.5824E+01	1.0786E+02	8.7441E+01	8.3983E+00
f_8	2.5859E−01	2.9861E+00	8.9668E+00	3.4420E+00	1.8203E+00	1.2942E+01	3.3851E+01	6.2508E+01	3.1864E+01	1.1025E+01
f_9	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
f_{10}	3.1127E−01	1.2901E+02	5.1056E+02	1.7165E+02	1.6562E+02	3.2773E+03	5.0614E+03	6.7406E+03	4.9685E+03	7.1837E+02
f_{11}	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	2.3440E+01	3.2141E+01	4.3315E+01	3.2485E+01	4.1108E+00
f_{12}	0.0000E+00	2.0814E−01	6.2442E−01	2.3263E−01	1.8915E−01	6.4180E+02	1.4591E+03	2.4321E+03	1.5630E+03	4.3564E+02
f_{13}	0.0000E+00	3.1878E−02	6.1411E+00	2.2682E+00	2.5246E+00	2.5359E+01	6.8049E+01	1.3215E+02	6.4721E+01	2.7165E+01
f_{14}	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	2.3107E+01	3.0995E+01	4.0545E+01	3.0749E+01	3.5680E+00
f_{15}	3.5925E−05	2.5035E−02	4.9999E−01	1.8680E−01	2.2078E−01	2.1155E+01	2.6044E+01	4.0685E+01	2.7390E+01	4.3379E+00
f_{16}	1.0757E−03	5.1779E−01	1.0632E+00	5.5560E−01	2.9898E−01	1.3675E+02	5.7848E+02	1.0727E+02	5.9033E+02	2.3639E+02
f_{17}	1.9729E−02	2.4844E+00	2.3439E+01	8.1845E+00	9.1275E+00	2.3120E+02	5.2167E+02	9.8010E+02	5.0397E+02	1.7659E+02
f_{18}	2.1375E−07	5.4496E−02	1.3067E+00	1.8423E−01	2.5356E−01	2.3122E+01	2.8713E+01	3.6439E+01	2.8835E+01	3.2090E+00
f_{19}	0.0000E+00	3.5147E−06	1.4380E−01	1.2360E−02	2.1860E−02	1.0626E+01	1.6050E+01	2.2626E+01	1.6885E+01	2.7897E+00
f_{20}	0.0000E+00	3.1217E−01	2.3020E+00	4.5141E−01	3.7668E−01	4.2014E+01	2.5082E+02	6.1836E+02	2.3804E+02	1.3987E+02
f_{21}	1.0000E+02	1.0000E+02	2.1274E+02	1.4999E+02	5.3581E+01	2.1567E+02	2.3349E+02	2.8330E+02	2.3102E+02	1.2102E+01
f_{22}	1.1563E+01	1.0000E+02	1.0040E+02	9.8273E+01	1.2384E+01	1.0000E+02	5.1831E+03	6.7027E+03	4.3108E+03	2.2134E+03
f_{23}	3.0000E+02	3.0300E+02	3.0636E+02	3.0303E+02	1.9012E+00	4.3814E+02	4.4935E+02	4.9197E+02	4.4033E+02	1.2550E+01
f_{24}	1.0000E+02	3.3099E+02	3.3531E+02	2.8570E+02	9.2745E+01	5.1248E+02	5.2979E+02	5.5257E+02	5.2013E+02	9.4801E+00

Table 3 (continued)

	10*D*					30*D*				
	Best	Median	Worst	Mean	Std	Best	Median	Worst	Mean	Std
f_{25}	3.9774E+02	3.9800E+02	4.4338E+02	4.0955E+02	1.9959E+01	4.7735E+02	4.8024E+02	4.9185E+02	4.8104E+02	3.2316E+00
f_{26}	3.0000E+02	3.0000E+02	3.0000E+02	3.0000E+02	0.0000E+00	1.4301E+03	1.7532E+03	2.4870E+03	1.8128E+03	2.1430E+02
f_{27}	3.8900E+02	3.8951E+02	3.8951E+02	3.8938E+02	2.2255E−01	5.0103E+02	5.1880E+02	5.5747E+02	5.1968E+02	9.9615E+00
f_{28}	3.0000E+02	3.0000E+02	6.1182E+02	3.3063E+02	8.7721E+01	4.5884E+02	4.5884E+02	4.5884E+02	4.5884E+02	3.8612E−10
f_{29}	2.2957E+02	2.3740E+02	2.5441E+02	2.3820E+02	5.9123E+00	3.2811E+02	3.9240E+02	7.0214E+02	4.5945E+02	1.3556E+02
f_{30}	3.9450E+02	3.9450E+02	3.9451E+02	3.9450E+02	5.3126E−03	5.7941E+05	5.9048E+05	7.5927E+05	6.1010E+05	4.0805E+04

Table 4 The results of the LFLDE algorithm for 50D and 100D

	50D					100D				
	Best	Median	Worst	Mean	Std	Best	Median	Worst	Mean	Std
f_1	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
f_3	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	1.8254E−06	1.6499E+00	3.2354E−02	2.3104E−01
f_4	5.8561E+01	5.8561E+01	5.8561E+01	5.8561E+01	0.0000E+00	1.8958E+02	1.9135E+02	2.0873E+02	1.9254E+02	9.4159E+00
f_5	7.1464E+00	1.7909E+01	4.3285E+01	1.9767E+01	7.7533E+00	3.9798E+01	7.6611E+01	1.4725E+02	8.0511E+01	2.4351E+01
f_6	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	1.5135E−05	3.5221E−04	9.4093E−03	1.0155E−03	1.5923E−03
f_7	3.6614E+01	4.0321E+01	8.2643E+01	4.1324E+01	8.2314E+00	1.3303E+02	1.4645E+02	1.6632E+02	1.4829E+02	1.0235E+01
f_8	9.9495E+00	1.8904E+01	6.2113E+01	2.1041E+01	1.0459E+01	4.0071E+01	5.1881E+01	1.0124E+02	5.6021E+01	1.8056E+01
f_9	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	8.1243E−01	7.8494E+00	1.6516E+00	2.0279E+00
f_{10}	2.7535E+03	4.1912E+03	6.3697E+03	4.1913E+03	7.7976E+02	9.5805E+04	1.1531E+04	1.3977E+04	1.0849E+04	1.1547E+03
f_{11}	8.1152E−01	5.9848E+00	7.5898E+01	1.0890E+01	1.6868E+01	7.2423E+01	1.6696E+02	2.7977E+02	1.6995E+02	4.4312E+01
f_{12}	8.1011E+00	2.3782E+02	8.6102E+02	2.4686E+02	1.7289E+02	1.8566E+04	5.6031E+04	2.1995E+05	6.6212E+04	3.9706E+04
f_{13}	3.0628E+00	2.2062E+01	3.6262E+01	1.9434E+01	8.5852E+00	7.8560E+01	1.5102E+02	2.9632E+02	1.6013E+02	4.5034E+01
f_{14}	1.9899E+00	2.7016E+01	3.6026E+01	2.2711E+01	9.9530E+00	4.8197E+01	7.2356E+01	1.0792E+02	7.2992E+01	1.2371E+01
f_{15}	1.7258E+00	6.5693E+00	1.4618E+01	6.8513E+00	2.9272E+00	9.7607E+01	1.7035E+02	2.8578E+02	1.7567E+02	3.9889E+01
f_{16}	1.0789E+01	2.4447E+02	5.8250E+02	2.4533E+02	1.6740E+02	9.8332E+02	2.0061E+03	3.0904E+03	2.0040E+03	4.9377E+02
f_{17}	1.1501E+01	3.5815E+01	7.1489E+01	3.7311E+01	1.0824E+01	6.9879E+02	1.3238E+03	2.0424E+03	1.3868E+03	3.4306E+02
f_{18}	2.1905E+00	2.3436E+01	2.8055E+01	2.3255E+01	3.4439E+00	1.0754E+02	1.7916E+02	2.7938E+02	1.8379E+02	3.9782E+01
f_{19}	3.0705E+00	5.9021E+00	9.6018E+00	6.1571E+00	1.5173E+00	6.6195E+01	1.3026E+02	1.7382E+02	1.2775E+02	2.5024E+02
f_{20}	1.4113E+00	2.5795E+01	1.6185E+02	3.1541E+01	3.1232E+01	7.7599E+01	1.6665E+03	2.5685E+03	1.6255E+03	3.8296E+02
f_{21}	2.1215E+02	2.3519E+02	2.7259E+02	2.3900E+02	1.2779E+01	2.5978E+02	2.8229E+02	3.2918E+02	2.8752E+02	1.6213E+01
f_{22}	1.0000E+02	1.0000E+02	1.0000E+02	1.0000E+02	0.0000E+00	8.9138E+03	1.2616E+04	1.6298E+04	1.2594E+04	1.5110E+03
f_{23}	3.5108E+02	3.7273E+02	3.9804E+02	3.7999E+02	1.0358E+01	6.0129E+02	6.6565E+02	7.8985E+02	6.7355E+02	3.6241E+01
f_{24}	4.2790E+02	4.4215E+02	4.6451E+02	4.4336E+02	9.1250E+00	9.7265E+02	1.0156E+03	1.1129E+03	1.0199E+03	3.1152E+01

Table 4 (continued)

	50*D*					100*D*				
	Best	Median	Worst	Mean	Std	Best	Median	Worst	Mean	Std
f_{25}	3.8668E+02	3.8669E+02	3.8673E+02	3.8669E+02	9.2772E−03	6.3726E+02	7.7011E+02	7.9991E+02	7.1579E+02	3.8199E+01
f_{26}	8.2923E+02	1.0683E+03	1.2796E+03	1.0594E+03	9.2606E+01	3.4738E+03	4.0978E+03	5.1229E+03	4.1104E+03	3.5744E+02
f_{27}	4.8446E+02	5.0342E+02	5.1572E+02	5.0099E+02	6.0748E+00	5.4919E+02	5.9370E+02	6.4786E+02	5.9681E+02	1.8761E+01
f_{28}	3.0000E+02	3.0000E+02	4.1397E+02	3.0670E+02	2.7084E+01	4.7818E+02	5.2366E+02	5.7613E+02	5.2469E+02	2.1554E+01
f_{29}	4.3893E+02	5.0212E+02	5.8991E+02	5.0234E+02	3.1482E+01	9.3582E+02	1.6404E+03	2.3848E+03	1.6563E+03	3.4037E+02
f_{30}	1.9438E+03	1.9715E+03	2.0871E+03	1.9766E+03	2.7815E+01	2.1055E+03	2.3014E+03	2.7466E+03	2.3341E+03	1.5594E+02

Table 5 Comparison of results for 10D

	CoDE Mean (Std Dev)	SaDE Mean (Std Dev)	EPSDE Mean (Std Dev)	MPEDE Mean (Std Dev)	LSHADE Mean (Std Dev)	LFLDE Mean (Std Dev)
f_1	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)**
f_3	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	0.0000E+00 (0.00E+00) ≈	4.6246E+01 (3.30E+02) −	**0.0000E+00 (0.00E+00)**
f_4	**0.0000E+00 (0.00E+00)** ≈	4.3325E−01 (7.74E−01) −	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)**
f_5	5.4356E+00 (1.33E+00) −	7.0021E+00 (1.68E+00) −	4.9700E+00 (1.25E+00) −	**2.7241E+00 (1.13E+00)** ≈	3.1046E+00 (7.61E−01) −	2.8614E+00 (1.02E+00)
f_6	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)**
f_7	1.8333E+01 (1.08E+00) −	1.9509E+01 (1.68E+00) −	1.5913E+01 (1.21E+00) −	**1.2801E+01 (1.44E+00)** ≈	1.5252E+01 (8.81E−01) ≈	1.4478E+01 (2.56E+00)
f_8	6.8704E+00 (1.27E+00) −	6.6485E+00 (1.43E+00) −	5.3575E+00 (1.44E+00) −	**3.0217E+00 (1.29E+00)** +	3.4938E+00 (9.82E−01) ≈	3.4420E+00 (1.82E+00)
f_9	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)**
f_{10}	2.7004E+02 (7.39E+01) −	4.0248E+02 (1.20E+02) −	4.1825E+02 (1.06E+02) −	7.1294E+01 (6.50E+01) +	**6.3443E+01 (6.46E+01)** +	1.7165E+02 (1.65E+02)
f_{11}	1.4532E−03 (4.41E−03) −	1.3799E−01 (3.07E−01) −	2.3126E+00 (9.41E−01) −	7.8036E−02 (2.70E−01) −	1.5711E+00 (5.64E−01) −	**0.0000E+00 (0.00E+00)**
f_{12}	1.5579E+01 (1.41E+01) −	7.4991E+01 (6.26E+01) −	1.2966E+02 (1.37E+02) ≈	2.6497E+01 (5.47E+00) −	4.4523E+01 (6.40E+01) −	**2.3263E−01 (1.89E−01)**
f_{13}	2.6990E+00 (1.27E+00) −	2.3765E+00 (2.75E+00) ≈	6.4252E+00 (1.86E+00) −	**2.1407E+00 (2.39E+00)** ≈	4.0194E+00 (2.08E+00) −	2.2682E+00 (2.52E+00)
f_{14}	**0.0000E+00 (0.00E+00)** ≈	3.8687E−01 (3.62E−01) −	2.0041E+01 (1.87E+00) −	6.0052E−02 (3.11E−01) −	2.9698E−01 (3.41E−01) −	**0.0000E+00 (0.00E+00)**

Table 5 (continued)

	CoDE Mean (Std Dev)	SaDE Mean (Std Dev)	EPSDE Mean (Std Dev)	MPEDE Mean (Std Dev)	LSHADE Mean (Std Dev)	LFLDE Mean (Std Dev)
f_{15}	**1.4644E−02 (6.88E−03)** +	1.0417E−01 (2.34E−01) +	9.6345E−01 (6.80E−01) −	9.8795E−01 (1.63E−01) −	2.2072E−01 (2.13E−01) −	1.8680E−01 (2.20E−01)
f_{16}	6.8449E−01 (1.96E−01) −	1.4533E+00 (6.84E−01) −	1.3212E+01 (6.70E+00) −	**5.1926E−01 (1.99E−01)** +	6.4848E−01 (1.79E−01) −	5.5560E−01 (2.98E−01)
f_{17}	**3.6579E−01 (1.83E−01)** +	2.7054E+00 (7.21E−01) +	2.1008E+01 (4.04E+00) −	2.1442E−01 (2.27E−01) +	**2.0571E−01 (2.33E−01)** +	8.1845E+00 (9.12E+00)
f_{18}	**7.0896E−03 (3.09E−03)** +	3.5798E−01 (2.12E−01) −	1.8854E+01 (4.58E+00) −	1.1677E−01 (2.11E−01) +	2.1619E−01 (1.97E−01) −	1.8423E−01 (2.53E−01)
f_{19}	3.1002E−02 (8.82E−02) −	2.1058E−02 (2.31E−02) −	7.6153E−01 (3.33E−01) −	**9.3385E−03 (1.07E−02)** +	2.0137E−02 (1.17E−02) −	1.2360E−02 (2.18E−02)
f_{20}	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	1.6370E+01 (5.26E+00) −	3.0602E−02 (9.37E−02) +	**0.0000E+00 (0.00E+00)** +	4.5141E−01 (3.76E−01)
f_{21}	**1.1001E+02 (3.32E+01)** +	1.2175E+02 (4.44E+01) +	1.5486E+02 (5.02E+01) ≈	1.5151E+02 (5.30E+01) ≈	1.5058E+02 (4.84E+01) ≈	1.4999E+02 (5.35E+01)
f_{22}	**3.6558E+01 (5.07E+01)** +	1.0000E+02 (0.00E+00) −	9.2972E+01 (2.50E+01) −	9.2338E+01 (2.64+01) +	1.0000E+02 (3.37E−09) −	9.8273E+01 (1.23E+01)
f_{23}	3.0488E+02 (1.19E+00) −	3.0503E+02 (3.30E+00) ≈	3.1299E+02 (2.19E+00) −	3.0453E+02 (2.20E+00) ≈	3.0444E+02 (1.35E+00) ≈	**3.0303E+02 (1.90E+00)**
f_{24}	3.6329E+02 (1.08E+02) −	**1.8174E+02 (1.11E+02)** +	3.3563E+02 (2.68E+00) −	2.9391E+02 (8.65E+01) −	3.0784E+02 (6.90E+01) −	2.8570E+02 (9.27E+01)
f_{25}	4.9790E+02 (1.52E−01) −	4.1097E+02 (2.06E+01) ≈	4.1899E+02 (2.33E+01) ≈	4.1335E+02 (2.20E+01) ≈	4.1048E+02 (2.05E+01) ≈	**4.0955E+02 (1.99E+01)**
f_{26}	**3.0000E+02 (0.00E+00)** ≈	**3.0000E+02 (0.00E+00)** ≈	2.9549E+02 (8.45E+01) ≈	**3.0000E+00 (0.00E+00)** ≈	**3.0000E+00 (0.00E+00)** ≈	**3.0000E+02 (0.00E+00)**
f_{27}	3.8765E+02 (1.06E+00) +	3.9382E+02 (1.35E−01) ≈	**3.8498E+02 (2.02E+01)** ≈	3.8938E+02 (3.62E−01) ≈	3.8926E+02 (2.59E−01) ≈	3.8938E+02 (2.22E−01)

Table 5 (continued)

	CoDE Mean (Std Dev)	SaDE Mean (Std Dev)	EPSDE Mean (Std Dev)	MPEDE Mean (Std Dev)	LSHADE Mean (Std Dev)	LFLDE Mean (Std Dev)
f_{28}	**3.0000E+02 (0.00E+00)** +	**3.0000E+02 (0.00E+00)** +	4.7352E+02 (2.30E+00) −	3.9898E+02 (1.41E+01) −	3.3503E+02 (9.70E+01) ≈	3.3063E+02 (8.77E+01)
f_{29}	2.4487E+02 (3.83E+00) −	2.5589E+02 (4.37E+00) ≈	2.5380E+02 (9.45E+00) −	2.3828E+02 (4.38E+00) ≈	2.3824E+02 (3.81E+00) ≈	**2.3820E+02 (5.91E+00)**
f_{30}	3.9526E+02 (1.37E−01) −	3.9852E+02 (6.08E+00) ≈	**2.1360E+02 (2.26E+01)** +	3.2443E+04 (1.60E+05) −	1.6613E+04 (1.14E+05) −	3.9450E+02 (5.31E−03)

Table 6 Comparison of results for 30*D*

	CoDE Mean (Std Dev)	SaDE Mean (Std Dev)	EPSDE Mean (Std Dev)	MPEDE Mean (Std Dev)	LSHADE Mean (Std Dev)	LFLDE Mean (Std Dev)
f_1	2.1041E+04 (5.66E+03) –	2.9988E−02 (8.93E−02) –	4.6790E−05 (2.29E−04) –	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)**
f_3	5.2341E+02 (1.68E+02) –	1.3625E−03 (1.86E−03) –	1.9560E+04 (3.78E+04) –	4.3549E−10 (3.11E−09) ≈	4.2675E+04 (3.82E+04) –	**0.0000E+00 (0.00E+00)**
f_4	8.7517E+01 (3.62E−01) –	**2.5536E+01 (3.26E+01)** +	5.9395E+01 (2.31E+00) –	5.2784E+01 (1.88E+01) +	5.8561E+01 (3.21E−14) ≈	5.8561E+01 (0.00E+00)
f_5	1.4167E+02 (4.23E+00) –	8.6026E+01 (7.27E+00) –	8.7224E+01 (7.11E+00) –	2.7722E+01 (7.28E+00) –	2.1306E+01 (3.12E+00) –	**1.9767E+01 (7.75E+00)**
f_6	1.8313E−01 (1.85E−02) –	7.2931E−09 (1.21E−083) ≈	6.0837E−04 (3.45E−04) –	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)**
f_7	1.8622E+02 (1.22E+01) –	1.2845E+02 (7.90E+00) –	1.2041E+02 (8.01E+00) –	5.8982E+01 (8.01E+00) –	5.1533E+01 (3.34E+00) –	**4.1324E+01 (8.23E+00)**
f_8	1.3935E+02 (1.16E+01) –	8.4635E+01 (5.37E+00) –	8.8707E+01 (7.64E+00) –	2.8883E+01 (6.31E+00) –	2.1359E+01 (3.08E+00) ≈	**2.1041E+01 (1.04E+01)**
f_9	9.6800E+01 (1.74E+01) –	**0.0000E+00 (0.00E+00)** ≈	7.3851E−01 (2.11E+00) –	8.9083E−03 (6.36E−02) –	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)**
f_{10}	4.8903E+03 (1.84E+02) –	4.2447E+03 (2.94E+02) ≈	4.8211E+03 (2.91E+02) –	2.8928E+03 (3.51E+02) +	**1.4504E+03 (2.03E+02)** +	4.1913E+03 (7.79E+02)
f_{11}	9.9206E+01 (9.89E+00) –	3.8269E+01 (1.58E+01) –	2.6192E+01 (5.65E+00) –	1.8074E+01 (1.01E+01) –	2.2623E+01 (2.66E+01) –	**1.0890E+01 (1.68E+01)**
f_{12}	5.9411E+04 (1.85E+04) –	1.4639E+03 (8.91E+02) –	2.8537E+04 (2.70E+04) –	9.1169E+02 (3.97E+02) –	1.0172E+03 (3.56E+02) –	**2.4686E+02 (1.72E+02)**
f_{13}	1.9281E+02 (1.28E+01) ≈	6.8857E+01 (3.40E+01) –	1.9276E+02 (3.02E+02) ≈	2.1326E+01 (1.2E+01) –	**1.7206E+01 (5.85E+00)** +	1.9434E+01 (8.58E+00)
f_{14}	6.0374E+01 (5.71E+00) –	5.6122E+01 (6.34E+00) –	4.3817E+01 (4.41E+00) –	**1.7286E+01 (1.00E+01)** +	2.1645E+01 (3.18E+00) ≈	2.2711E+01 (9.95E+00)

Table 6 (continued)

	CoDE Mean (Std Dev)	SaDE Mean (Std Dev)	EPSDE Mean (Std Dev)	MPEDE Mean (Std Dev)	LSHADE Mean (Std Dev)	LFLDE Mean (Std Dev)
f_{15}	5.6877E+01 (6.14E+00) −	2.3443E+01 (1.47E+01) −	7.3511E+01 (9.96E+01) −	**3.3889E+00 (2.37E+00)** +	3.4842E+00 (1.86E+00) +	6.8513E+00 (2.92E+00)
f_{16}	6.2845E+02 (7.47E+01) −	6.4276E+02 (1.63E+02) −	9.1398E+02 (1.56E+02) −	2.9864E+02 (1.59E+02) −	**1.6889E+02 (9.24E+00)** +	2.4533E+02 (1.67E+02)
f_{17}	1.5343E+02 (2.33E+01) −	1.1930E+02 (1.83E+01) −	3.4543E+02 (7.87E+01) −	6.5258E+01 (2.13E+01) −	3.9369E+01 (7.13E+00) −	**3.7311E+01(1.08E+01)**
f_{18}	6.7011E+01 (3.33E+00) −	3.4692E+01(5.75E+00) −	6.1509E+01 (4.74E+01) −	**2.1854E+01 (7.09E+00)** +	1.0526E+03 (7.35E+03) −	2.3255E+01 (3.44E+00)
f_{19}	2.9864E+01 (3.21E+00) −	2.6167E+01 (6.62E+00) −	2.1495E+01 (2.26E+00) −	8.0043E+00 (2.25E+00) −	**6.1323E+00 (1.91E+00)** ≈	6.1571E+00 (1.51E+00)
f_{20}	1.3816E+02 (2.83E+01) −	1.6802E+02 (4.77E+01) −	2.6369E+02 (7.48E+01) −	9.2037E+01 (5.71E+01) −	4.1352E+01 (1.46E+01) −	**3.1541E+01 (3.12E+01)**
f_{21}	3.3836E+02 (6.88E+00) −	2.8183E+02 (6.87E+00) −	2.9674E+02 (6.28E+00) −	2.2826E+02 (7.70E+00) ≈	**2.2238E+02 (4.25E+00)** ≈	2.3900E+02 (1.27E+01)
f_{22}	1.0013E+02 (2.08E−02) −	1.0000E+02 (2.26E−12) ≈	2.0717E+03 (1.40E+03) −	1.0000E+02 (3.43E−01) ≈	1.0000E+02 (1.00E−13) ≈	**1.0000E+02 (0.00E+00)**
f_{23}	4.8093E+02 (4.74E+00) −	4.3199E+02 (9.56E+00) −	4.4519E+02 (9.21E+00) −	3.7864E+02 (9.91E+00) ≈	**3.6948E+02 (3.89E+00)** ≈	3.7999E+02 (1.03E+01)
f_{24}	5.7300E+02 (7.31E+00) −	4.8873E+02 (9.31E+00) −	5.2948E+02 (9.82E+00) −	4.4231E+02 (8.99E+00) ≈	**4.3795E+02 (5.30E+00)** ≈	4.4336E+02 (9.12E+00)
f_{25}	3.8722E+02 (3.36E−02) −	3.8706E+02 (2.19E−01) ≈	3.8685E+02 (2.27E−02) ≈	3.8680E+02 (6.55E−02) ≈	3.8672E+02 (2.39E−02) ≈	**3.8669E+02 (9.27E−03)**
f_{26}	2.3402E+03 (1.10E+02) −	1.4502E+03 (.52E+02) −	1.6110E+03 (7.28E+01) −	1.2170E+03 (1.04E+02) −	1.1012E+03 (7.86E+01) ≈	**1.0594E+03 (9.26E+01)**
f_{27}	5.1490E+02 (3.46E+00) −	5.1148E+02 (1.16E+01) ≈	**5.0000E+02 (8.72E−05)** ≈	5.0078E+02 (7.30E+00) ≈	5.0437E+02 (5.22E+00) ≈	5.0099E+02 (6.07E+00)

Table 6 (continued)

	CoDE Mean (Std Dev)	SaDE Mean (Std Dev)	EPSDE Mean (Std Dev)	MPEDE Mean (Std Dev)	LSHADE Mean (Std Dev)	LFLDE Mean (Std Dev)
f_{28}	4.1788E+02 (8.70E+00) −	3.3111E+02 (4.51E+01) ≈	4.9958E+02 (1.40E+00) −	3.4843E+02 (5.65E+01) −	3.2166E+02 (4.48E+01) −	**3.0670E+02 (2.70E+01)**
f_{29}	6.9319E+02 (6.05E+01) −	5.7928E+02 (2.19E+01) −	6.3442E+02 (1.07E+02) −	4.7267E+02 (2.61E+01) +	**4.3810E+02 (1.51E+01)** +	5.0234E+02 (3.14E+01)
f_{30}	3.8980E+03 (2.08E+02) −	2.1614E+03 (6.44E+01) −	2.2302E+02 (1.17E+02) −	2.0319E+03 (1.05E+02) ≈	2.0037E+03 (6.09E+01) ≈	**1.9766E+03 (2.78 (E+01)**

Table 7 Comparison of results for 50D

	CoDE Mean (Std Dev)	SaDE Mean (Std Dev)	EPSDE Mean (Std Dev)	MPEDE Mean(Std Dev)	LSHADE Mean (Std Dev)	LFLDE Mean (Std Dev)
f_1	8.1693E+07 (8.68E+06) –	1.3324E+03 (1.06E+03) –	9.9122E+02 (2.24E+02) –	**0.0000E+00 (0.00E+00)** ≈	**0.00000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)**
f_3	3.7648E+04 (5.79E+03) –	1.4681E+02 (1.12E+02) –	4.2046E+05 (9.51E+04) –	**6.4641E−03 (3.70E−02)** +	1.27605E+05 (8.51E+04) –	1.2141E−01 (1.30E−01)
f_4	2.9041E+02 (9.42E+00) –	1.2515E+02 (3.98E+01) –	**3.5106E+01 (6.31E−01)** +	6.9128E+01 (4.43E+01) +	7.93326E+01 (2.15E+02) ≈	8.0970E+01 (5.10E+01)
f_5	3.5054E+02 (8.84E+00) –	2.1492E+02 (6.67E+00) –	2.4857E+02 (1.16E+01) –	5.0795E+01 (9.37E+00) –	3.34954E+01 (6.98E+00) ≈	**3.2148E+01 (1.21E+01)**
f_6	4.9539E+00 (2.48E−01) –	1.2322E−05 (1.71E−06) –	3.9610E−02 (7.96E−03) –	4.4793E−04 (1.65E−03) –	5.03372E−08 (1.26E−07) ≈	**0.0000E+00 (0.00E+00)**
f_7	4.4770E+02 (2.15E+01) –	2.7579E+02 (1.03E+01) –	2.9510E+02 (1.17E+01) –	1.1480E+04 (2.11E+01) –	**7.75634E+01 (4.53E+00)** +	8.7441E+01(8.39E+00)
f_8	3.6006E+02 (6.05E+00) –	2.3196E+02 (1.12E+01) –	2.4365E+02 (1.36E+01) –	5.5139E+01 (1.27E+01) –	3.24236E+01 (5.79E+00) ≈	**3.1864E+01 (1.10E+01)**
f_9	1.7660E+03 (3.00E+02) –	5.7260E−01 (6.51E−01) –	**0.0000E+00 (0.00E+00)** ≈	1.9713E−01 (4.53E−01) –	1.75545E−03 (1.25E−02) –	**0.0000E+00 (0.00E+00)**
f_{10}	1.0508E+04 (2.55E+02) –	8.8243E+03 (3.37E+02) –	1.0616E+04 (3.58E+02) –	6.4534E+03 (6.12E+02) –	**3.23224E+03 (2.15E+02)** +	4.9685E+03 (7.18E+02)
f_{11}	2.4145E+02 (1.60E+01) –	1.0104E+02 (2.04E+01) –	1.0257E+02 (3.62E+01) –	7.2995E+01 (1.42E+01) –	4.73087E+01 (9.57E+00) –	**3.2485E+01 (4.11E+00)**
f_{12}	2.2886E+07 (3.08E+06) –	2.2879E+04 (1.27E+04) –	2.5588E+05 (1.89E+05) –	2.5440E+03 (1.50E+03) –	2.21259E+03 (4.39E+02) –	**1.5630E+03 (4.35E+02)**
f_{13}	2.5537E+04 (4.19E+03) –	2.8745E+02 (4.86E+01) –	1.5402E+03 (1.91E+03) –	9.2009E+03 (3.64E+01) –	**5.50384E+01 (2.33E+01)** +	6.4721E+01 (2.71E+01)
f_{14}	1.7477E+02 (6.42E+00) –	1.3855E+02 (1.54E+01) –	1.1591E+02 (1.68E+01) –	4.6757E+01 (9.68E+00) –	3.09760E+01 (2.83E+00) ≈	**3.0749E+01 (3.56E+00)**

Table 7 (continued)

	CoDE Mean (Std Dev)	SaDE Mean (Std Dev)	EPSDE Mean (Std Dev)	MPEDE Mean(Std Dev)	LSHADE Mean (Std Dev)	LFLDE Mean (Std Dev)
f_{15}	4.8563E+02 (4.64E+01) –	1.3310E+02 (2.91E+01) –	1.8790E+02 (1.93E+02) –	4.5500E+01 (1.39E+01) –	4.34640E+01 (1.16E+01) –	**2.7390E+01 (4.33E+00)**
f_{16}	1.9867E+03 (2.11E+02 –	1.3074E+03 (1.44E+02) –	2.1429E+03 (1.68E+02) –	8.0602E+02 (2.80E+02) –	5.92701E+02 (1.34E+02) ≈	**5.9033E+02 (2.36E+02)**
f_{17}	1.0815E+03 (1.25E+02) –	9.4305E+02 (1.20E+02) –	1.3425E+03 (1.16E+02) –	7.4257E+02 (2.04E+02) –	**3.60627E+02 (8.61E+01)** +	5.0397E+02 (1.76E+02)
f_{18}	2.3954E+04 (3.49E+03) –	5.9105E+01 (1.66E+01) –	1.3100E+03 (9.68E+02) –	4.2184E+01 (1.99E+01) –	5.08801E+01 (2.29E+01) –	**2.8835E+01 (3.20E+00)**
f_{19}	2.1192E+02 (3.26E+01) –	5.1914E+01 (7.21E+00) –	5.8178E+01 (8.37E+00) –	2.6767E+01 (4.65E+00) –	2.93105E+01 (8.54E+00) –	**1.6885E+01 (2.78E+00)**
f_{20}	7.7424E+02 (9.77E+01) –	7.8669E+02 (7.79E+01) –	1.0573E+03 (1.37E+02) –	6.1257E+02 (1.87E+02) –	2.60250E+02 (8.38E+01) –	**2.3804E+02 (1.39E+02)**
f_{21}	5.4122E+02 (1.94E+01) –	4.0389E+02 (1.24E+01) –	4.5125E+02 (1.25E+01) –	2.5063E+02 (1.28E+01) ≈	2.34270E+02 (7.45E+00) ≈	**2.3102E+02 (1.21E+01)**
f_{22}	8.1411E+03 (4.18E+03) –	**1.0220E+02 (5.87E+00)** +	1.1105E+04 (3.54E+02) –	2.5657E+03 (3.36E+03) +	9.72543E+03 (1.52E+03) –	4.3108E+03 (2.21E+03)
f_{23}	7.7569E+02 (1.26E+01) –	6.3981E+02 (1.43E+01) –	6.4263E+02 (1.76E+01) –	4.7599E+02 (1.69E+01) –	4.50672E+02 (6.89E+00) ≈	**4.4033E+02 (1.25E+01)**
f_{24}	8.6006E+02 (1.56E+01) –	6.7093E+02 (1.31E+01) –	7.9954E+02 (1.59E+01) –	5.3872E+02 (1.21E+01) ≈	**5.19163E+02 (5.56E+00)** ≈	5.2013E+02 (9.48E+01)
f_{25}	6.1547E+02 (6.44E+00) –	5.3393E+02 (3.99E+01) –	**4.3126E+02 (3.33E−02)** +	5.1562E+02 (3.20E+01) –	4.82119E+02 (4.24E+00) ≈	4.8104E+02 (3.23E+00)
f_{26}	4.5797E+03 (1.77E+02) –	3.1241E+03 (1.38E+02) –	3.3568E+03 (4.56E+02) –	1.5393E+03 (1.37E+02) ≈	**1.34866E+03 (6.52E+01)** +	1.8128E+03 (2.14E+02)
f_{27}	8.8379E+02 (1.96E+01) –	5.2453E+02 (1.63E+01) ≈	**5.0001E+02 (7.62E−05)** +	5.2847E+02 (1.03E+01) ≈	5.31692E+02 (1.22E+01) ≈	5.1968E+02 (9.96E+00)

Table 7 (continued)

	CoDE Mean (Std Dev)	SaDE Mean (Std Dev)	EPSDE Mean (Std Dev)	MPEDE Mean(Std Dev)	LSHADE Mean (Std Dev)	LFLDE Mean (Std Dev)
f_{28}	7.6839E+02 (2.53E+01) −	5.0497E+02 (2.38E+01) −	5.0001E+02 (5.82E−05) −	4.8161E+02 (2.44E+01) −	4.63475E+02 (1.42E+01) ≈	**4.5884E+02** (**3.86E−10)**
f_{29}	1.6921E+03 (1.91E+02) −	8.0812E+02 (7.26E+01) −	1.5760E+03 (2.14E+02) −	4.1676E+02 (6.56E+01) +	**3.57170E+02** **(1.06E+01)** +	4.5945E+02 (1.35E+02)
f_{30}	1.1889E+07 (7.24E+05) −	6.4256E+05 (1.65E+04) −	**4.3502E+02** **(2.13E+02)** +	6.5315E+05 (9.44E+04) −	6.64365E+05 (7.91E+04) −	6.1010E+05(4.08E+04)

Table 8 Comparison of results for 100D

	CoDE Mean (Std Dev)	SaDE Mean (Std Dev)	EPSDE Mean (Std Dev)	MPEDE Mean (Std Dev)	LSHADE Mean (Std Dev)	LFLDE Mean (Std Dev)
f_1	5.8117E+09 (8.79E+08) –	5.1894E+03 (2.28E+03) –	3.7470E+03 (3.55E+03) –	2.4671E−04 (1.70E−04) –	**0.0000E+00 (0.00E+00)** ≈	**0.0000E+00 (0.00E+00)**
f_3	3.1096E+05 (1.79E+04) –	1.6064E+04 (3.19E+03) –	9.6216E+05 (1.56E+05) –	4.2024E+02 (1.09E+03) –	3.6104E+05 (2.03E+05) –	**3.2354E−02 (2.31E−01)**
f_4	1.3090E+03(6.20E+01) –	2.9893E+02 (2.82E+01) –	9.3834E+01 (7.22E−01) –	**1.7280E+02 (4.70E+01)** +	1.9397E+02 (1.13E+01) ≈	1.9254E+02 (9.41E+00)
f_5	9.6652E+02 (2.19E+01) –	6.4840E+02 (2.82E+01) –	7.2475E+02 (2.11E+01) –	1.1248E+02 (1.83E+01) –	**5.9655E+01 (8.65E+00)** +	8.0511E+01 (2.43E+01)
f_6	2.6230E+01 (2.05E+00) –	3.1974E−03 (6.47E+00) –	1.3219E+00 (4.10E−01) –	1.2616E−02 (1.09E−02) –	1.3376E−03 (8.93E−04) –	**1.0155E−03 (1.59E−03)**
f_7	1.2787E+03 (4.49E+01) –	7.6561E+02 (2.07E+01) –	8.2121E+02 (1.92E+01) –	2.9432E+02 (7.40E+01) –	1.5694E+02 (7.56E+00) ≈	**1.4829E+02 (1.02E+01)**
f_8	9.6241E+02 (2.98E+01) –	6.4203E+02 (2.11E+01) –	7.2562E+02 (1.77E+01) –	1.1103E+02 (1.66E+01) –	5.9008E+01 (7.76E+00) –	**5.6021E+01 (1.80E+01)**
f_9	1.9771E+04 (2.23E+03 –	3.5725E+00 (3.09E+00) –	1.1011E−01 (3.06E−01) +	1.7588E+00 (1.22E+00) ≈	**7.4122E−02 (1.37E−01)** +	1.6516E+00 (2.02E+00)
f_{10}	2.6971E+04 (5.44E+02) –	2.3543E+04 (3.82E+02) –	2.7852E+04 (5.09E+02) –	1.8989E+04 (945E+02) –	**1.0159E+04 (5.59E+02)** ≈	1.0849E+04 (1.15E+03)
f_{11}	1.4804E+04 (1.81E+03) –	5.6846E+02 (9.36E+01) –	7.4214E+02 (1.75E+02) –	4.3884E+02 (1.71E+02) –	5.2715E+03 (9.21E+03) –	**1.6995E+02 (4.43E+01)**
f_{12}	1.5035E+09 (1.92E+08 –	4.3446E+05 (1.16E+05) –	1.9335E+07 (3.17E+07) –	5.3390E+04 (2.59E+04) +	**2.1636E+03 (9.49E+02)** +	6.6212E+04 (3.97E+04)
f_{13}	9.9492E+06 (1.41E+06) –	2.5723E+03 (1.23E+03) –	1.0486E+05 (1.60E+05) –	4.0118E+02 (3.76E+02) –	1.2433E+03 (9.49E+02) –	**1.6013E+02 (4.50E+01)**
f_{14}	4.1490E+05 (6.60E+04) –	2.7115E+02 (5.25E+01) –	2.5214E+03 (2.37E+03) –	2.5996E+02 (3.88E+01) –	2.5451E+02 (3.42E+01) –	**7.2992E+01 (1.23E+01)**

Table 8 (continued)

	CoDE Mean (Std Dev)	SaDE Mean (Std Dev)	EPSDE Mean (Std Dev)	MPEDE Mean (Std Dev)	LSHADE Mean (Std Dev)	LFLDE Mean (Std Dev)
f_{15}	4.2959E+05 (6.38E+04) –	4.1379E+02 (1.17E+02) –	2.9128E+03 (2.05E+03) –	2.6141E+02 (6.02E+01) –	2.3894E+02 (4.02E+01) –	**1.7567E+02 (3.97E+01)**
f_{16}	7.7557E+03 (2.23E+02) –	4.9753E+03 (2.67E+02) –	7.0483E+03 (3.30E+02) –	2.5635E+03 (5.25E+02) –	2.0383E+03 (2.75E+02) ≈	**2.0040E+03 (4.93E+02)**
f_{17}	4.5997E+03 (1.63E+02) –	3.2334E+03 (1.82E+02) –	4.4448E+03 (2.53E+02 –	2.5158E+03 (3.74E+02) –	1.4423E+03 (1.91E+02) ≈	**1.3868E+03 (3.43E+02)**
f_{18}	3.0817E+06 (4.33E+05) –	3.8764E+02 (1.78E+02) –	1.5662E+05 (7.03E+04) –	2.8629E+02 (8.32E+01) –	2.2428E+02 (4.90E+01) –	**1.8379E+02 (3.97E+01)**
f_{19}	1.6026E+06 (3.51E+05) –	1.7821E+02 (3.12E+01) –	4.2246E+02 (2.81E+02) –	1.8374E+02 (3.52E+01) –	1.6882E+02 (2.17E+01) –	**1.2775E+02 (2.50E+02)**
f_{20}	3.7108E+03 (2.14E+02) –	3.3196E+03 (1.45E+02) –	3.9315E+03 (1.84E+02) –	2.8364E+03 (4.08E+02) –	1.7027E+03 (2.08E+02) ≈	**1.6255E+03 (3.82E+02)**
f_{21}	1.2053E+03 (3.03E+01) –	8.4399E+02 (1.63E+01) –	9.5014E+02 (1.87E+01) –	3.3296E+02 (2.18E+02) –	**2.8448E+02 (8.71E+00)** ≈	2.8752E+02 (1.62E+01)
f_{22}	2.8566E+04 (2.63E+02) –	2.3607E+04 (3.29E+E+03) –	2.8631E+04 (5.54E+02) –	1.9586E+04 (1.16E+03) –	**1.1477E+04 (5.23E+02)** ≈	1.2594E+04 (1.51E+03)
f_{23}	1.3951E+03 (2.25E+01) –	1.1265E+03 (2.50E+01) –	1.2222E+03 (1.59E+01) –	6.5971E+02 (2.49E+01) ≈	**5.9996E+02 (9.05E+00)** +	6.7355E+02 (3.62E+01)
f_{24}	1.8385E+03 (4.32E+01) –	1.1182E+03(2.04E+02) –	1.5984E+03 (1.95E+01) +	9.8099E+02 (2.14E+01) ≈	**9.3433E+02 (7.88E+00)** +	1.0199E+03 (3.15E+01)
f_{25}	2.3951E+03 (1.19E+02) –	8.5463E+02 (3.88E+02) –	7.5104E+02 (1.46E+01) –	7.3757E+02 (4.33E+01) ≈	7.5089E+02 (2.55E+01) ≈	**7.1579E+02 (3.81E+01)**
f_{26}	1.2980E+04 (2.27E+02) –	7.1750E+03 (1.84E+03) –	1.0878E+04 (1.76E+02) –	3.8297E+03 (2.30E+02) –	**3.5349E+03 (9.91E+01)** +	4.1104E+03 (3.57+02)
f_{27}	1.4900E+03 (3.53E+01) –	6.6926E+02 (2.82E+01) –	**5.0002E+02 (7.10E−05)** +	6.3689E+02 (2.40E+01) –	6.3901E+02 (1.93E+01) –	5.9681E+02 (1.87E+01)

Table 8 (continued)

	CoDE Mean (Std Dev)	SaDE Mean (Std Dev)	EPSDE Mean (Std Dev)	MPEDE Mean (Std Dev)	LSHADE Mean (Std Dev)	LFLDE Mean (Std Dev)
f_{28}	2.1929E+03 (1.45E+02) –	7.1848E+02 (2.90E+01) –	**5.0004E+02 (7.65E−05)** +	5.5075E+02 (3.21E+01) –	5.1446E+02 (1.76E+01) ≈	5.2469E+02 (2.15E+01)
f_{29}	5.9439E+03 (3.28E+02) –	3.6728E+03 (2.09E+02) –	4.7916E+03 (2.48E+02) –	2.1590E+03 (4.70E+02) –	**1.3623E+03 (1.68E+02)** +	1.6563E+03 (3.40E+02)
f_{30}	3.5230E+06 (5.23E+05) –	5.4847E+03 (8.32E+02) –	**4.9755E+02 (1.13E+02)** +	2.4891E+03 (1.57E+02) ≈	2.4273E+03 (1.51E+02) ≈	2.3341E+03 (1.55E+02)

Table 9 The summarized results of the Wilcoxon rank-sum test ($p = 0.05$)

Vs. LFLDE	$D = 10$	$D = 30$	$D = 50$	$D = 100$
CoDE				
+ (Win)	7	0	0	0
− (Lose)	14	28	29	29
≈ (Equal)	8	1	0	0
SaDE				
+ (Win)	5	1	1	0
− (Lose)	12	21	27	29
≈ (Equal)	12	7	1	0
EPSDE				
+ (Win)	1	0	4	5
−(Lose)	18	26	24	24
≈ (Equal)	10	3	1	0
MPEDE				
+ (Win)	8	6	4	2
− (Lose)	7	13	20	22
≈ (Equal)	14	10	5	5
LSHADE				
+ (Win)	3	5	6	7
− (Lose)	13	9	11	10
≈ (Equal)	13	15	12	12

better or worse (≈) compared to LFLDE according to the Wilcoxon rank-sum test at the 0.05 significance level [35].

Summarizing the results of the statistical testing compared with the LFLDE algorithm is presented in Table 9, if we compare a number of wins (+) and loses (−). We can see that LFLDE indicates the better performance than the state-of-the-DE algorithms on all dimensions. In addition, based on the average results obtained, the average ranking of all algorithms, as produced by the Friedman rank test, is summarized in Table 10. The results in Table 10 are consistent with the results in Table 9, in which LFLDE has the best rank, and LSHADE gets the

Table 10 Friedman ranks for all algorithms

Algorithm	Rank	F-rank			
		$D = 10$	$D = 30$	$D = 50$	$D = 100$
CoDE	1	3.29	5.62	5.65	5.82
SaDE	2	4.12	4.01	3.93	4.10
EPSDE	3	4.63	4.68	4.36	4.37
MPEDE	4	2.91	2.46	3.03	2.86
LSHADE	5	3.36	2.24	2.34	2.08
LFLDE	6	2.67	1.96	1.67	1.74

second rank. All the experiments show that the proposed algorithm framework is effective, and the fitness landscape can boost the performance of the algorithm.

5 Conclusions and further work

In this paper, we used fitness landscape information to analyze the optimization problems; then guiding the selection of mutation strategy at each generation, a novel control parameters adaptive mechanism is utilized to enhance the proposed algorithm. Besides, we also adopted population linear reduction method. The performance of the algorithm was evaluated on the set of benchmark functions provided for CEC 2017 special session on single-objective real-parameter optimization. The experimental results give evidence that the LFLDE algorithm is highly competitive when comparing to the advanced variant DE algorithms such as CoDE, SaDE, EPSDE, MPEDE and LSHADE on 29 benchmark functions with 10, 30, 50 and 100 dimensions. All the results indicated that the fitness landscape information is benefit for improve the performance of DE. For future research, we plan to apply the fitness landscape to realize the composite mutation strategy and control parameter selection in DE algorithm.

Acknowledgements This work is supported by National Key R&D Program of China with the Grant No.2018YFC0831100, the National Natural Science Foundation of China with the Grant No.61773296, the National Natural Science Foundation Youth Fund Project of China under Grant No. 61703170, the Major Science and Technology Project in Dongguan with Grant No. 2018215121005 and Key R&D Program of Guangdong Province with No. 2019B020219003. Besides, the authors extend their appreciation to the Deanship of Scientific Research at King Saud University for funding this work through Research Group no. RG-144-331.

References

1. Bansal S (2019) A comparative study of nature-inspired metaheuristic algorithms in search of near-to-optimal Golomb rulers for the FWM crosstalk elimination in WDM systems. Appl Artif Intell 33(14):1199–1265
2. Di Serafino D, Ruggiero V, Toraldo G et al (2018) On the steplength selection in gradient methods for unconstrained optimization. Appl Math Comput 318:176–195
3. Jahani E, Chizari M (2018) Tackling global optimization problems with a novel algorithm–Mouth Brooding Fish algorithm. Appl Soft Comput 62:987–1002
4. Li K, Chen R, Fu G et al (2018) Two-archive evolutionary algorithm for constrained multiobjective optimization. IEEE Trans Evol Comput 23(2):303–315
5. Das S, Mullick SS, Suganthan PN et al (2016) Recent advances in differential evolution—an updated survey. Swarm Evol Comput 27:1–30
6. Das S, Mullick SS, Suganthan PN (2016) Recent advances in differential evolution–an updated survey. Swarm Evol Comput 27:1–30
7. Storn R, Price K (1997) Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–359
8. Mohamed AW, Hadi AA, Jambi KM (2019) Novel mutation strategy for enhancing SHADE and LSHADE algorithms for global numerical optimization. Swarm Evol Comput 50:100455
9. Das S, Suganthan PN (2011) Differential evolution: a survey of the state-of-the-art. IEEE Trans Evol Comput 15(1):4–31

10. Qin AK, Huang VL, Suganthan PN et al (2009) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evol Comput 13(2):398–417
11. Islam SM, Das S, Ghosh S et al (2012) An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization. IEEE Syst Man Cybern 42(2):482–500
12. Zhang J, Sanderson AC (2009) JADE: adaptive differential evolution with optional external archive. IEEE Trans Evol Comput 13(5):945–958
13. Brest J, Greiner S, Boskovic B et al (2006) Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans Evol Comput 10(6):646–657
14. Tanabe R, Fukunaga A (2013) Evaluating the performance of SHADE on CEC 2013 benchmark problems. In: 2013 IEEE Congress on Evolutionary Computation. IEEE, pp 1952–1959
15. Tanabe R, Fukunaga AS (2014) Improving the search performance of SHADE using linear population size reduction. In: 2014 IEEE Congress on Evolutionary Computation (CEC). IEEE, pp 1658–1665
16. Brest J, Maučec MS, Bošković B (2017) Single objective real-parameter optimization: algorithm jSO. In: 2017 IEEE Congress on Evolutionary Computation (CEC). IEEE, pp 1311–1318
17. Poláková R, Tvrdík J, Bujok P (2016) Evaluating the performance of L-SHADE with competing strategies on CEC2014 single parameter-operator test suite. In: 2016 IEEE Congress on Evolutionary Computation (CEC). IEEE, pp 1181–1187
18. Awad NH, Ali MZ, Suganthan PN et al (2016) An ensemble sinusoidal parameter adaptation incorporated with L-SHADE for solving CEC2014 benchmark problems. In: 2016 IEEE Congress on Evolutionary Computation (CEC). IEEE, pp 2958–2965
19. Stanovov V, Akhmedova S, Semenkin E (2018) LSHADE algorithm with rank-based selective pressure strategy for solving CEC 2017 benchmark problems. In: 2018 IEEE Congress on Evolutionary Computation (CEC). IEEE, pp 1–8
20. Qin AK, Huang VL, Suganthan PN (2008) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evol Comput 13(2):398–417
21. Wang Y, Cai Z, Zhang Q (2011) Differential evolution with composite trial vector generation strategies and control parameters. IEEE Trans Evol Comput 15(1):55–66
22. Elsayed SM, Sarker RA, Essam DL (2011) Multi-operator based evolutionary algorithms for solving constrained optimization problem. Comput Oper Res 38(12):1877–1896
23. Mallipeddi R, Suganthan PN, Pan QK et al (2011) Differential evolution algorithm with ensemble of parameters and mutation strategies. Appl Soft Comput 11(2):1679–1696
24. Malan KM, Engelbrecht AP (2013) A survey of techniques for characterising fitness landscapes and some possible ways forward. Inf Sci 241:148–163
25. Wang M, Li B, Zhang G et al (2018) Population evolvability: dynamic fitness landscape analysis for population-based metaheuristic algorithms. IEEE Trans Evol Comput 22(4):550–563
26. Li K, Liang Z, Yang S et al (2019) Performance analyses of differential evolution algorithm based on dynamic fitness landscape. Int J Cogn Inf Nat Intell (IJCINI) 13(1):36–61
27. Malan KM, Engelbrecht AP (2009) Quantifying ruggedness of continuous landscapes using entropy. In: Congress on Evolutionary Computation, pp 1440–1447
28. Sallam KM, Elsayed SM, Sarker RA et al (2017) Landscape-based adaptive operator selection mechanism for differential evolution. Inf Sci 418–419:383–404
29. Viktorin A, Senkerik R, Pluhacek M et al (2019) Distance based parameter adaptation for success-history based differential evolution. Swarm Evol Comput 50:100462
30. Tong L, Dong M, Jing C et al (2018) An improved multi-population ensemble differential evolution. Neurocomputing 290:130–147
31. Wu G, Mallipeddi R, Suganthan PN et al (2016) Differential evolution with multi-population based ensemble of mutation strategies. Inf Sci 329:329–345
32. Bansal S, Gupta N, Singh AK Application of bat-inspired computing algorithm and its variants in search of near-optimal golomb rulers for WDM systems: a comparative study. In: Applications of Bat Algorithm and its Variants. Springer, Singapore, pp 79–101
33. Awad NH, Ali MZ, Mallipeddi R et al (2018) An improved differential evolution algorithm using efficient adapted surrogate model for numerical optimization. Inf Sci 451–452:326–347
34. Biedrzycki R, Arabas J, Jagodzinski D et al (2019) Bound constraints handling in differential evolution: an experimental study. Swarm Evol Comput 50:100453

35. Derrac J, García S, Molina D et al (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol Comput 1(1):3–18