The Journal of Supercomputing
https://doi.org/10.1007/s11227-019-03044-9

# Self-adaptive collective intelligence-based mutation operator for differential evolution algorithms

Jinhong Feng[1] · Jundong Zhang[1] · Chuan Wang[1] · Minyi Xu[1]

**Abstract**
In conventional differential evolutionary (DE) algorithm, mutation operator has significant influence on generating new vectors by mixing existing target vectors randomly selected from the current population. Recently, many mutation operators, which usually employ the best individual or some high-quality individuals randomly chosen, have been proposed to improve searching capability. However, such designs may easily suffer from premature convergence trapped by local optima. To make a trade-off between exploration and exploitation capability, this paper proposes a novel collective intelligence (CI)-based mutation operator, which is named as "current-to-sa-ci-best." In the presented mutation operator, the evolutionary information of $m$ best target vectors is linearly combined to generate new mutant vectors. Besides, $m$ is designed as an exponential-distributed random number which could be self-adapted based on successful records of $m$ values alongside evolution. Moreover, this mutation operator could be applied to any DE algorithm without destroying existing search capability by adding a greedy selection operator. To verify its effectiveness, the proposed CI-based mutation strategy, which is named as SaCI, was embedded into some state-of-the-art DE variants on 28 CEC2013 benchmark functions. Numerical results have confirmed that the SaCI operator may be beneficial to DEs to some extent.

**Keywords** Differential evolution · Mutation · Self-adaption · Collective intelligence

## 1 Introduction

Differential evolution (DE) algorithm has become very popular due to its simplicity and effectiveness since 1990s [19]. DE and its high-cited variants have been widely used to solve many real-world engineering problems on various domains such as

✉ Jinhong Feng
fjhdlmu@dlmu.edu.cn

[1] Marine Engineering College, Dalian Maritime University, Dalian 116026, Liaoning, People's Republic of China

Published online: 24 October 2019

PID tuning [21], power system [4, 26], parameter identification [22], time series prediction [8], feature selection [1] and clustering [6]. The performance of DE relies on the evolutionary operators (mutation, crossover and selection) and the control parameters [5]. Generally speaking, the mutation process is the key operator to generate new promising vectors. Well, the most appropriate mutation and corresponding parameter settings required by DE should be different and adapted to solve different optimization problems and evolving stages [23]. Besides, it is inefficient and time-consuming to use trial-and-error approach for determining the best mutation strategy [24]. Thus, it is very important and challenging to develop a self-adaptive mutation operator to fit various optimization problems during the whole evolution process.

Since the last decade, papers on designing efficient mutation strategies naturally have attracted increasing attention. In MPEDE [24], the authors proposed a multi-population-based approach to realize an ensemble of three mutation strategies, i.e., “current-to-pbest/1,” “current-to-rand/1” and “rand/1.” CoDE [23] just simply used three trial vector generation strategies and three control parameter settings. It randomly combined them to generate trial vectors. Similarly, EPSDE [24] proposed an ensemble of up to 8 mutation strategies and a pool of control parameters settings. In SaDE [17], the authors employed a pool of mutation strategy and its self-adapted selection mechanism by learning from their previous experiences. Some other DE algorithms such as JADE [27] and MDE_*p*BX [14] employed mutation strategies which selected one from a group of top ranked vectors or the best group from a randomly selected subset of the population. Overall, for mutation operator, there are a few shortcomings for the above methods: (1) It is difficult to determine the size of the mutation-strategies pool, as well as each mutation strategy. (2) New control parameters are usually added into the mutation-operator-selected process from the ensemble of mutation strategies. Based on these observations, we will develop a novel self-adaptive mutation strategy by using collective intelligence (CI) for DE algorithms.

CI [18] is group intelligence generated from the collaboration, collective efforts, and competition of many individuals and appears in consensus decision-making. This new intelligence phenomenon also could be defined as “the capacity of human communities to evolve toward higher-order complexity and harmony, through such innovation mechanisms as differentiation and integration, competition and collaboration” [28]. CI may achieve a collective wisdom in some cases and outperform the performance of the smartest individual within the group, although the majority of the group members are not experts. It has been successfully used in many applications such as interactive debugging [16], maximum power point tracking [25], nonlinear constrained optimization [9], business [20], sensor network [15] and prediction [2]. In this paper, we would try to embed CI into DE algorithm to generate a mutant vector mixed with some top ranked individuals in a collaborative manner without introducing any control parameter. More details could be seen in Sect. 3.

The main contribution of this paper is that we propose a novel mutation operator by using CI without designing a complex strategy selection mechanism, resulting in a simple but effective mutation vector. Besides, the present mutation operator, which is named as “current-to-sa-ci-best,” could be self-adapted based on records of the previous successful vectors by using an exponential distribution model. Many recent DE variants had introduced new control parameters to peruse better performances, giving recommended values for the control parameters. Different from the other new published DE algorithms, the presented mutation operator by using CI does not introduce any new control parameters. Further, the self-adaption mechanism of SaCI operator would be easily applied to almost all the existing DE algorithms, without destroying the searching ability of the original algorithm. It may be summarized as the main advantage of the proposed SaCI mutation operator.

The remainder of this paper is structured as follows: In the next section, related works on structure of conventional DE algorithm will be introduced. In Sect. 3, self-adaptive CI-based mutation operator will be proposed. In Sect. 4, comprehensive experimental results on CEC2013 benchmark functions will be shown and discussed. Finally, the contributions of this paper would be summarized in Sect. 5.

## 2 Conventional DE algorithm

DE algorithm is trying to evolve a population at each generation $G$, which could be denoted as $\mathbf{X}_{i,\,G} = (x_{i,G}{}^{1}, x^{2}{}_{i,\,G}, \ldots, x^{D}{}_{i,\,G} | i = 1, 2, \ldots, NP)$, where $NP$ is the population size and $D$ is the dimension of the problem. After initialization, DE enters a loop of evolutionary operations: mutation, crossover and selection.

### 2.1 Mutation

DE algorithm employs the mutation operator to generate a mutant vector $\mathbf{V}_{i,G}$ with respect to each individual $\mathbf{X}_{i,G}$, which is named as target vector. At the generation $G$, its corresponding mutant vector $\mathbf{V}_{i,G} = (v^{1}_{i,\,G}, v^{2}_{i,\,G}, \ldots, v^{D}_{i,\,G})$ would be generated via various mutation strategies. In this section, we would give a review on the five most frequently used mutation strategies as follows:

DE/rand/1 [7]

$$\mathbf{V}_{i,G} = \mathbf{X}_{r_1^i,G} + F \cdot \left( \mathbf{X}_{r_2^i,G} - \mathbf{X}_{r_3^i,G} \right) \tag{1}$$

DE/best/1 [7]

$$\mathbf{V}_{i,G} = \mathbf{X}_{\text{best},G} + F \cdot \left( \mathbf{X}_{r_1^i,G} - \mathbf{X}_{r_2^i,G} \right) \tag{2}$$

DE/rand-to-best/1 [17]

$$\mathbf{V}_{i,G} = \mathbf{X}_{i,G} + F \cdot \left( \mathbf{X}_{\text{best},G} - \mathbf{X}_{i,G} \right) + F \cdot \left( \mathbf{X}_{r_1^i,G} - \mathbf{X}_{r_2^i,G} \right) \tag{3}$$

DE/best/2 [7]

$$\mathbf{V}_{i,G} = \mathbf{X}_{\text{best},G} + F \cdot \left( \mathbf{X}_{r_1^i,G} - \mathbf{X}_{r_2^i,G} \right) + F \cdot \left( \mathbf{X}_{r_3^i,G} - \mathbf{X}_{r_4^i,G} \right) \tag{4}$$

DE/rand/2 [17]

$$\mathbf{V}_{i,G} = \mathbf{X}_{r_1^i,G} + F \cdot \left(\mathbf{X}_{r_2^i,G} - \mathbf{X}_{r_3^i,G}\right) + F \cdot \left(\mathbf{X}_{r_4^i,G} - \mathbf{X}_{r_5^i,G}\right) \tag{5}$$

where the indices $r_1^i$, $r_2^i$, $r_3^i$, $r_4^i$, $r_5^i$ and $i$ are exclusive integers randomly generated within the range from 1 to $NP$. The scale factor $F$ is a positive control parameter for scaling the difference vector. $\mathbf{X}_{\mathrm{best},G}$ is the best individual with the best fitness value in the current population at generation $G$.

## 2.2 Crossover

After the mutation operator, crossover operation is used to generate a trial vector $\mathbf{U}_{i,G}$ for each pair of $\mathbf{X}_{i,G}$ and $\mathbf{V}_{i,G}$. In the basic version, the binomial crossover is usually adopted as follows (Wang et al. 2016):

$$u_{i,G}^j = \begin{cases} v_{i,G}^j, & \text{if} \quad rand_j \le Cr \text{ or } j = j_{rand} \\ x_{i,G}^j, & \text{otherwise} \end{cases} \qquad j = 1, 2, \ldots, D \tag{6}$$

In Eq. (6), the crossover rate $Cr$ is a pre-defined constant within the range [0, 1]. $j_{rand}$ is a randomly integer within the range [1, $D$]. $rand_j$ is a uniformly distributed random number generated within [0, 1].

## 2.3 Selection

Before selection operation, the objective function values of all trial vectors are evaluated. After that, a selection operation is performed. $f(\mathbf{U}_{i,G})$ and $f(\mathbf{X}_{i,G})$ are objective function values of trial vector and its corresponding target vector, respectively. The selection operator could be defined as follows:

$$\mathbf{X}_{i,G+1} = \begin{cases} \mathbf{U}_{i,G}, & \text{if} \quad f(\mathbf{U}_{i,G}) < f(\mathbf{X}_{i,G}) \\ \mathbf{X}_{i,G}, & \text{otherwise} \end{cases} \tag{7}$$

The three basic operators above are looped till the specified termination criteria are met. The algorithmic description of the conventional DE algorithm is displayed as follows:

**Algorithm 1.** The pseudo code of DE with "DE/rand/1" mutation operator.

```
1   Randomly initialize each individual within the searching ranges; Calculate each fitness value
    of every individual; Set generation number $G = 1$; set function evaluations as $FEs = NP$; Set $F$ and
    $Cr$ values;
2   while $FEs \le$ Max_FEs
3       for $i = 1 : NP$
4           Generate mutant vector $\mathbf{V}_{i,G}$ according to Eq. (1);
5           Randomly select index of individual from $NP$ as $j_{rand}$;
6           Generate trial vector $\mathbf{U}_{i,G}$ according to Eq. (6);
7           Evaluate the fitness value of $\mathbf{U}_{i,G}$;
8           $FEs = FEs + 1$;
9           if $f(\mathbf{U}_{i,G}) \le f(\mathbf{X}_{i,G})$
10              $\mathbf{X}_{i,G+1} = \mathbf{U}_{i,G}$;
11          end
12      end
13      $G = G + 1$;
14  end
```

## 3 Self-adaptive CI-based mutation operator

In this part, a novel self-adaptive CI-based mutation strategy, which is denoted as "current-to-sa-ci-best," is presented. This mutation operator uses collective information of some top ranked and randomly selection vectors to generate mutant vector, which is described as follows:

$$\mathbf{V}_{sa\_ci\_i,G} = \mathbf{X}_{i,G} + F \cdot \left(\mathbf{X}_{sa\_ci\_best,G} - \mathbf{X}_{i,G}\right) + F \cdot \left(\mathbf{X}_{r_1^i,G} - \mathbf{X}_{r_2^i,G}\right) \tag{8}$$

where $\mathbf{X}_{i,G}$ is the $i$th target vector, $\mathbf{X}_{r_1^i,G}$ and $\mathbf{X}_{r_2^i,G}$ are two randomly selected distinctive vectors from the current population, and $\mathbf{X}_{sa\_ci\_best,G}$ is a collective vector which could be defined as follows:

$$\mathbf{X}_{sa\_ci\_best,G} = \sum_{k=1}^{m} w_k \cdot \mathbf{X}_{k,G} \tag{9}$$

In Eq. (9), the proposed collective vector is a linear combination of $m$ top ranked vectors. The weighting $w_k$ determines the contribution of each chosen vector within $m$ top ranked vectors. Here, we provide an expression of $w_k$ as follows:

$$w_k = \frac{m - k + 1}{1 + 2 + \ldots + m} \quad \text{for} \quad k = 1, 2, \ldots m. \tag{10}$$

In Eq. (9–10), $m$ has great influence on the quality of collective vector. If $m$ is set to 1, the proposed mutation strategy "current-to-sa-ci-best" would be simplified as "current-to-best/1." If $m$ is set to $NP$, the collective vector would become the average value of the whole population. Different from CIMDE [28], which just defined $m$ as a random integer within $[1, NP]$, $m$ is designed as a randomly generated integer which obeys exponential distribution contributed by its fitness value with respect to $\mathbf{X}_{i,G}$. In this paper, the whole self-adaption of $m$ could be described as follows: (1) generate the probability $p_i$ of exponentially distributed random number whose mean is $\mu$ for each target vector $i$; (2) generate $m_i$ for $i$th target vector according to $p_i$ by using roulette wheel selection; (3) if a better trial vector is obtained, the $m_i$ value would be added into the successful record which is named as $S_m$; (4) estimate the mean value $\mu$ of the successful record $S_m$; (5) go to step (1). In the above steps, $m$ is assumed as an exponential distribution random number with a mean value $\mu$. Successful $m$ values would be saved into $S_m$, and new $m$ values will be generated according to $S_m$. Since $m$ obeys exponential distribution, the value of $m$ will gradually be close to 1. Thus, a positive feedback process, which ultimately brings each target vector closer to the linear combination of several globally top ranked target vectors, is formed. In summary, such a design would make $m$ self-adapted according to the previous successful $m$ values without introducing any additional control parameter, resulting in a more powerful mutation strategy. To make the proposed SaCI mutation strategy clearer, the pseudocode of DE/current-to-sa-ci-best algorithm is given below.

**Algorithm 2.** The pseudo code of DE algorithm with SaCI mutation operator.

```
1   Randomly initialize each individual within the searching ranges; Calculate each fitness value
    of every individual; Set generation number $G = 1$; set function evaluations as $FEs = NP$; Set $F$ and
    $Cr$ values; set initial $\mu$ value to 0.001; set successful record of $m$ value $S_m = \emptyset$ ;
2   while $FEs \leq$ Max_FEs
3       Calculate exponential probability density function $p_i$ for each $\mathbf{X}_{i,G}$ with $\mu$; «
4       for $i = 1 : NP$
5           Generate mutant vector $\mathbf{V}_{i,G}$ according to Eq. (1);
6           Calculate $m_i$ value by using roulette wheel selection according to $p_i$; «
7           Generate mutant vector $\mathbf{V}_{sa\_ci\_i,G}$ according to Eq. (8-10); «
8           Randomly select index of individual from $NP$ as $j_{rand}$;
9           Generate trial vector $\mathbf{U}_{i,G}$ according to Eq. (6);
10          Generate trial vector $\mathbf{U}_{sa\_ci\_i,G}$ according to Eq. (6); «
11          Evaluate the fitness value of $\mathbf{U}_{i,G}$;
12          Evaluate the fitness value of $\mathbf{U}_{sa\_ci\_i,G}$; «
13          $FEs = FEs + 2$;
14          if $f(\mathbf{U}_{i,G}) \leq f(\mathbf{X}_{i,G})$ & $f(\mathbf{U}_{i,G}) \leq f(\mathbf{U}_{sa\_ci\_i,G})$
15              $\mathbf{X}_{i,G+1} = \mathbf{U}_{i,G}$;
16          else if $f(\mathbf{U}_{sa\_ci\_i,G}) \leq f(\mathbf{X}_{i,G})$ & $f(\mathbf{U}_{sa\_ci\_i,G}) \leq f(\mathbf{U}_{i,G})$ «
17              $\mathbf{X}_{i,G+1} = \mathbf{U}_{sa\_ci\_i,G}$; «
18              Add $m_i$ into $S_m$; «
19          end
20      end
21      Estimate the mean value $\mu$ of the pre-assumed exponentially distributed $S_m$; «
22      $G = G + 1$;
23  end
```

For ease of comparison, the differences between Algorithm 2 and Algorithm 1 are marked by using "«." From the above pseudocode, we can see in each iteration two different mutation vectors are generated for each target vector, finally resulting in two different trial vectors. In order to obtain the most outperforming vector, a greedy selection operator is developed to choose the most promising vector among the three vectors in line 14–19. Such a greedy operator could guarantee that the searching capability of previous mutation strategy would not deteriorate. Besides, such a design makes the proposed SaCI mutation strategy easily applicable. Next, we will conduct a comprehensive comparison to test the performance of the proposed SaCI mutation strategy.

## 4 Experimental results and analysis

In this part, the authors are trying to conduct comprehensive experiments to test the performance of the proposed SaCI mutation strategy. Benchmark functions presented in CEC2013 are chosen as the test suite. More details about the definition of benchmark problems are given in Table 1. The computational configurations are listed as follows:

- OS: Win 7.
- CPU: Intel i5 @2.60 GHz.
- RAM: 8G.
- Platform: Matlab 2012b.

**Table 1** The 28 CEC 2013 benchmark functions used in the experiments

| No. | Functions | $D$ | Range | $f(x^*)$ |
|---|---|---|---|---|
| 1 | Shifted Sphere | 10 | $[-100, 100]^D$ | −1400 |
| 2 | Shifted Rotated High Conditioned Elliptic | 10 | $[-100, 100]^D$ | −1300 |
| 3 | Shifted Rotated Bent Cigar | 10 | $[-100, 100]^D$ | −1200 |
| 4 | Shifted Rotated Discus | 10 | $[-100, 100]^D$ | −1100 |
| 5 | Shifted Different Powers | 10 | $[-100, 100]^D$ | −1000 |
| 6 | Shifted Rotated Rosenbrock | 10 | $[-100, 100]^D$ | −900 |
| 7 | Shifted Rotated Schaffers F7 | 10 | $[-100, 100]^D$ | −800 |
| 8 | Shifted Rotated Ackley | 10 | $[-100, 100]^D$ | −700 |
| 9 | Shifted Rotated Weierstrass | 10 | $[-100, 100]^D$ | −600 |
| 10 | Shifted Rotated Griewank | 10 | $[-100, 100]^D$ | −500 |
| 11 | Shifted Rastrigin | 10 | $[-100, 100]^D$ | −400 |
| 12 | Shifted Rotated Rastrigin | 10 | $[-100, 100]^D$ | −300 |
| 13 | Shifted Non-Continuous Rotated Rastrigin | 10 | $[-100, 100]^D$ | −200 |
| 14 | Shifted Schwefel | 10 | $[-100, 100]^D$ | −100 |
| 15 | Shifted Rotated Schwefel | 10 | $[-100, 100]^D$ | 100 |
| 16 | Shifted Rotated Katsuura | 10 | $[-100, 100]^D$ | 200 |
| 17 | Shifted Lunacek Bi_Rastrigin | 10 | $[-100, 100]^D$ | 300 |
| 18 | Shifted Rotated Lunacek Bi_Rastrigin | 10 | $[-100, 100]^D$ | 400 |
| 19 | Shifted Expanded Griewank plus Rosenbrock | 10 | $[-100, 100]^D$ | 500 |
| 20 | Shifted Expanded Schaffer F6 | 10 | $[-100, 100]^D$ | 600 |
| 21 | Shifted Composition Function 1 ($n=5$, Rotated) | 10 | $[-100, 100]^D$ | 700 |
| 22 | Shifted Composition Function 2 ($n=3$, Unrotated) | 10 | $[-100, 100]^D$ | 800 |
| 23 | Shifted Composition Function 3 ($n=3$, Rotated) | 10 | $[-100, 100]^D$ | 900 |
| 24 | Shifted Composition Function 4 ($n=3$, Rotated) | 10 | $[-100, 100]^D$ | 1000 |
| 25 | Shifted Composition Function 5 ($n=3$, Rotated) | 10 | $[-100, 100]^D$ | 1100 |
| 26 | Shifted Composition Function 6 ($n=5$, Rotated) | 10 | $[-100, 100]^D$ | 1200 |
| 27 | Shifted Composition Function 7 ($n=5$, Rotated) | 10 | $[-100, 100]^D$ | 1300 |
| 28 | Shifted Composition Function 8 ($n=5$, Rotated) | 10 | $[-100, 100]^D$ | 1400 |

There are 28 benchmark functions used in the following experiments. According to their properties, they are categorized into three types: uni-modal problems ($f_1-f_5$), basic multimodal problems ($f_6-f_{20}$) and composition problems ($f_{21}-f_{28}$). All the problems used in this paper are minimization problems. All test functions are shifted and scalable. Simulations terminate when reaching Max_FEs or the error value is smaller than $10^{-8}$.

## 4.1 Compared DE algorithms and simulation setup

In order to fairly test the performances of SaCI mutation strategy for DE algorithm, we have used the following eight state-of-the-art algorithms shown in Table 2 with their recommended setup in the original literature. All of the peer methods could be easily embedded with SaCI mutation operator.

All benchmark functions were tested in 10 dimensions. The population size $NP$ was set to 50 for $D=10$. The maximum of function evaluations (Max_FEs) was set to 3000$D$. Calculated results of different algorithms on each function were averaged over 30 independent runs and are reported in Table 3. In order to compare the significance between two algorithms, the Wilcoxon rank sum test at 0.05 level is used [10, 11]. The Wilcoxon rank sum test results regarding Algorithm 1 versus Algorithm 2 are shown as '+', '−', '≈', when Algorithm 1 is significantly better than, significantly worse than and significantly equal to Algorithm 2, respectively. The comparison results among DEs with SaCI operator and the original algorithms are summarized as ''*w*/*t*/*l*'' in the last row of the table. In detail, it means that the original algorithm wins in *w* functions, ties in *t* functions and loses in *l* functions, compared with its competitors with SaCI operator.

From Table 3, we can see for $D=10$ problems, DE algorithms with SaCI mutation operator would beat the corresponding original algorithms on 4 to 26 benchmark functions. Meanwhile, the methods with SaCI mutation operator deteriorated the performances on very few test functions. For example, jDE-SaCI had outperformed jDE on 12 test functions ($f_2$, $f_4$, $f_6$, $f_9$, $f_{10}$, $f_{12}$, $f_{13}$, $f_{20}$, $f_{24}$, $f_{25}$, $f_{26}$, $f_{27}$), and jDE-SaCI was defeated on 2 test functions ($f_3$, $f_{28}$). For the other benchmark functions (14 out of 28 cases), there were no significant differences between jDE and

**Table 2** Parameter settings of involved DE variants

| Algorithm | Setup |
|---|---|
| DE/rand/1 [19] | $F=0.5$, $Cr=0.9$ |
| DE/best/1 [19] | $F=0.5$, $Cr=0.9$ |
| SaDE [17] | $F=\text{Norm}(0.5, 0.3)$, $Cr=\text{Norm}(Cr_{mk}, 0.1)$, $LP=50$ |
| JADE [27] | $c=0.1$, $p=0.05$, $\mu_F=0.5$, $\mu_{CR}=0.5$ |
| jDE [3] | $F_l=0.1$, $F_u=0.9$, $\tau_1=\tau_2=0.1$ |
| AS-JADE [13] | $c=0.1$, $p=0.05$, $\mu_F=0.5$, $\mu_{CR}=0.5$, $\alpha=0.3$, $\beta=0.8$ |
| rank-jDE [12] | $F_l=0.1$, $F_u=0.9$, $\tau_1=\tau_2=0.1$ |
| CIMDE/rand/1 [28] | $c=0.1$, $\mu_F=0.7$, $\mu_{CR}=0.5$ |

**Table 3** Compared results of DE algorithms and the corresponding method with SaCI operator for $D = 10$

| | DE/rand/1 versus DE/rand/1-SaCI | | DE/best/1 versus DE/best/1-SaCI | | SaDE versus SaDE-SaCI | | JADE versus JADE-SaCI | |
|---|---|---|---|---|---|---|---|---|
| $f_1$ | 0.00E+00 ± 0.00E+00 | + | 7.93E+01 ± 1.00E+02 | – | 0.00E+00 ± 0.00E+00 | ≈ | 0.00E+00 ± 0.00E+00 | ≈ |
| | 6.18E−04 ± 2.05E−03 | | 1.88E+01 ± 2.13E+01 | | 0.00E+00 ± 0.00E+00 | | 0.00E+00 ± 0.00E+00 | |
| $f_2$ | 8.80E+03 ± 1.86E+04 | + | 5.73E+05 ± 1.36E+06 | ≈ | 5.37E+04 ± 1.45E+05 | ≈ | 1.84E+06 ± 1.02E+06 | – |
| | 1.33E+04 ± 1.82E+04 | | 1.65E+05 ± 2.74E+05 | | 1.65E+04 ± 1.68E+04 | | 2.95E+05 ± 1.86E+05 | |
| $f_3$ | 1.09E+00 ± 2.81E+00 | + | 4.34E+08 ± 7.93E+08 | ≈ | 1.29E+03 ± 3.47E+03 | + | 2.19E+06 ± 3.01E+06 | – |
| | 2.40E+06 ± 5.73E+06 | | 2.00E+08 ± 2.86E+08 | | 6.00E+03 ± 1.15E+04 | | 2.25E+03 ± 5.09E+03 | |
| $f_4$ | 3.95E+02 ± 8.00E+02 | ≈ | 9.54E+02 ± 1.08E+03 | ≈ | 1.44E+03 ± 1.41E+03 | ≈ | 1.78E+04 ± 9.66E+03 | – |
| | 1.51E+02 ± 2.19E+02 | | 6.84E+02 ± 3.95E+02 | | 8.55E+02 ± 6.49E+02 | | 8.92E+03 ± 3.88E+03 | |
| $f_5$ | 0.00E+00 ± 0.00E+00 | + | 7.70E+01 ± 7.92E+01 | ≈ | 0.00E+00 ± 3.60E−14 | ≈ | 0.00E+00 ± 8.81E−14 | – |
| | 2.16E−01 ± 5.09E−01 | | 4.42E+01 ± 4.02E+01 | | 0.00E+01 ± 0.00E+00 | | 0.00E+00 ± 0.00E+00 | |
| $f_6$ | 2.55E+00 ± 1.32E+00 | ≈ | 4.68E+01 ± 4.20E+01 | ≈ | 4.06E+00 ± 1.48E+00 | – | 4.17E+00 ± 2.99E+00 | ≈ |
| | 4.63E+00 ± 3.20E+00 | | 1.54E+01 ± 1.13E+01 | | 2.48E+00 ± 2.48E+00 | | 1.66E+00 ± 7.19E−01 | |
| $f_7$ | 1.42E−02 ± 1.33E−02 | + | 7.05E+01 ± 3.20E+01 | – | 3.44E−01 ± 3.37E−01 | + | 1.40E+01 ± 1.16E+01 | – |
| | 1.87E+00 ± 3.62E+00 | | 4.26E+01 ± 2.86E+01 | | 3.47E+00 ± 3.84E+00 | | 9.05E−01 ± 9.73E−01 | |
| $f_8$ | 2.04E+01 ± 8.54E−02 | ≈ | 2.05E+01 ± 1.05E−01 | ≈ | 2.04E+01 ± 7.48E−02 | ≈ | 2.04E+01 ± 7.60E−02 | ≈ |
| | 2.04E+01 ± 5.36E−02 | | 2.05E+01 ± 7.19E−02 | | 2.05E+01 ± 7.59E−02 | | 2.04E+01 ± 5.88E−02 | |
| $f_9$ | 9.56E+00 ± 9.94E−01 | – | 5.12E+00 ± 1.27E+00 | ≈ | 8.05E+00 ± 1.21E+00 | ≈ | 7.73E+00 ± 8.05E−01 | ≈ |
| | 2.10E+00 ± 2.32E+00 | | 5.06E+00 ± 1.04E+00 | | 6.89E+00 ± 8.89E−01 | | 7.89E+00 ± 1.17E+00 | |
| $f_{10}$ | 2.35E−01 ± 2.08E−01 | ≈ | 5.78E+01 ± 5.50E+01 | – | 1.20E−01 ± 7.43E−02 | ≈ | 6.10E−01 ± 2.48E−01 | ≈ |
| | 1.81E−01 ± 1.67E−01 | | 1.60E+01 ± 1.27E+01 | | 1.60E−01 ± 5.29E−02 | | 4.03E−01 ± 1.08E−01 | |
| $f_{11}$ | 1.96E+01 ± 6.63E+00 | – | 3.46E+01 ± 2.05E+01 | ≈ | 1.13E−11 ± 2.45E−11 | + | 3.19E+00 ± 1.26E+00 | – |
| | 9.66E+00 ± 3.60E+00 | | 2.04E+01 ± 5.52E+00 | | 8.61E−06 ± 1.11E−05 | | 1.77E+00 ± 1.01E+00 | |
| $f_{12}$ | 2.99E+01 ± 4.56E+00 | – | 4.38E+01 ± 2.05E+01 | – | 1.53E+01 ± 3.35E+00 | ≈ | 2.97E+01 ± 4.83E+00 | – |
| | 2.00E+01 ± 7.32E+00 | | 2.05E+01 ± 8.70E+00 | | 1.59E+01 ± 4.12E+00 | | 2.29E+01 ± 5.16E+00 | |
| $f_{13}$ | 2.83E+01 ± 7.01E+00 | – | 5.82E+01 ± 1.56E+01 | – | 1.85E+01 ± 2.81E+00 | ≈ | 2.72E+01 ± 6.35E+00 | ≈ |
| | 1.91E+01 ± 9.57E+00 | | 3.95E+01 ± 1.73E+01 | | 1.66E+01 ± 4.56E+00 | | 2.36E+01 ± 4.64E+00 | |

**Table 3** (continued)

| | DE/rand/1 versus DE/rand/1-SaCI | | DE/best/1 versus DE/best/1-SaCI | | SaDE versus SaDE-SaCI | | JADE versus JADE-SaCI | |
|---|---|---|---|---|---|---|---|---|
| $f_{14}$ | 1.36E+03 ± 2.24E+02 | ≈ | 5.67E+02 ± 2.17E+02 | ≈ | 3.22E+01 ± 5.23E+01 | + | 6.47E+02 ± 1.17E+02 | ≈ |
| | 1.37E+03 ± 1.79E+02 | | 5.24E+02 ± 1.43E+02 | | 7.17E+01 ± 6.44E+01 | | 5.82E+02 ± 1.47E+02 | |
| $f_{15}$ | 1.70E+03 ± 2.05E+02 | – | 8.68E+02 ± 3.78E+02 | ≈ | 1.40E+03 ± 1.93E+02 | ≈ | 1.62E+03 ± 1.52E+02 | ≈ |
| | 1.58E+03 ± 1.29E+02 | | 8.76E+02 ± 4.77E+02 | | 1.26E+03 ± 1.60E+02 | | 1.52E+03 ± 1.62E+02 | |
| $f_{16}$ | 1.05E+00 ± 2.17E−01 | ≈ | 1.41E+00 ± 4.59E−01 | – | 1.52E+00 ± 2.51E−01 | ≈ | 1.32E+00 ± 2.82E−01 | ≈ |
| | 1.11E+00 ± 1.92E−01 | | 1.02E+00 ± 2.12E−01 | | 1.43E+00 ± 2.10E−01 | | 1.41E+00 ± 3.90E−01 | |
| $f_{17}$ | 3.04E+01 ± 5.13E+00 | ≈ | 3.74E+01 ± 1.54E+01 | – | 1.11E+01 ± 7.63E−01 | + | 1.54E+01 ± 1.62E+00 | ≈ |
| | 2.84E+01 ± 3.54E+00 | | 2.22E+01 ± 1.03E+01 | | 1.24E+01 ± 9.05E−01 | | 1.44E+01 ± 1.04E+00 | |
| $f_{18}$ | 3.94E+01 ± 5.16E+00 | – | 4.85E+01 ± 1.70E+01 | – | 3.64E+01 ± 3.39E+00 | ≈ | 4.04E+01 ± 5.81E+00 | ≈ |
| | 3.37E+01 ± 4.44E+00 | | 3.47E+01 ± 1.21E+01 | | 3.62E+01 ± 3.79E+00 | | 3.70E+01 ± 3.56E+00 | |
| $f_{19}$ | 2.52E+00 ± 4.06E−01 | – | 8.04E+00 ± 1.06E+01 | – | 6.97E−01 ± 1.46E−01 | ≈ | 1.48E+00 ± 4.16E−01 | ≈ |
| | 2.09E+00 ± 2.33E−01 | | 2.26E+00 ± 1.44E+00 | | 7.71E−01 ± 1.40E−01 | | 1.44E+00 ± 2.78E−01 | |
| $f_{20}$ | 3.48E+00 ± 2.17E−01 | – | 3.55E+00 ± 5.33E−01 | – | 3.36E+00 ± 3.11E−01 | ≈ | 3.35E+00 ± 2.78E−01 | ≈ |
| | 2.94E+00 ± 3.79E−01 | | 3.06E+00 ± 5.02E−01 | | 3.33E+00 ± 3.12E−01 | | 3.23E+00 ± 2.78E−01 | |
| $f_{21}$ | 3.68E+02 ± 7.48E+01 | + | 3.83E+02 ± 5.28E+01 | ≈ | 4.00E+02 ± 2.38E−13 | ≈ | 3.82E+02 ± 6.04E+01 | ≈ |
| | 4.00E+02 ± 3.40E−04 | | 4.02E+02 ± 1.34E+00 | | 4.00E+02 ± 2.38E−13 | | 4.00E+02 ± 2.38E−13 | |
| $f_{22}$ | 1.58E+03 ± 1.93E+02 | ≈ | 7.18E+02 ± 2.70E+02 | – | 2.44E+02 ± 1.21E+02 | ≈ | 8.58E+02 ± 1.53E+02 | ≈ |
| | 1.45E+03 ± 1.90E+02 | | 4.94E+02 ± 2.13E+02 | | 2.25E+02 ± 9.51E+01 | | 7.84E+02 ± 2.05E+02 | |
| $f_{23}$ | 1.87E+03 ± 2.26E+02 | – | 1.14E+03 ± 3.96E+02 | – | 1.68E+03 ± 2.54E+02 | ≈ | 1.69E+03 ± 2.55E+02 | ≈ |
| | 1.56E+03 ± 2.40E+02 | | 8.38E+02 ± 2.85E+02 | | 1.43E+03 ± 2.06E+02 | | 1.65E+03 ± 1.86E+02 | |
| $f_{24}$ | 2.24E+02 ± 1.98E+00 | – | 2.16E+02 ± 4.56E+00 | ≈ | 2.21E+02 ± 1.73E+00 | – | 2.22E+02 ± 5.19E+00 | ≈ |
| | 2.09E+02 ± 9.03E+00 | | 2.14E+02 ± 3.49E+00 | | 2.17E+02 ± 1.92E+00 | | 2.22E+02 ± 2.34E+00 | |
| $f_{25}$ | 2.24E+02 ± 1.75E+00 | ≈ | 2.15E+02 ± 4.11E+00 | – | 2.23E+02 ± 1.63E+00 | – | 2.24E+02 ± 1.43E+00 | ≈ |
| | 2.23E+02 ± 7.03E+00 | | 2.12E+02 ± 3.82E+00 | | 2.20E+02 ± 1.48E+00 | | 2.23E+02 ± 2.33E+00 | |
| $f_{26}$ | 2.08E+02 ± 3.09E+01 | ≈ | 2.21E+02 ± 7.60E+01 | – | 2.00E+02 ± 1.21E−02 | – | 2.12E+02 ± 5.93E+01 | ≈ |
| | 2.04E+02 ± 4.61E+01 | | 1.78E+02 ± 5.86E+01 | | 1.85E+02 ± 2.85E+01 | | 2.01E+02 ± 3.39E+00 | |

Table 3 (continued)

| | DE/rand/1 versus DE/rand/1-SaCI | | DE/best/1 versus DE/best/1-SaCI | | SaDE versus SaDE-SaCI | | JADE versus JADE-SaCI | |
|---|---|---|---|---|---|---|---|---|
| $f_{27}$ | 5.46E+02 ± 1.36E+01 | – | 5.06E+02 ± 4.87E+01 | – | 5.05E+02 ± 2.73E+01 | ≈ | 5.38E+02 ± 1.36E+01 | ≈ |
| | 3.56E+02 ± 6.31E+01 | | 4.29E+02 ± 5.19E+01 | | 4.84E+02 ± 2.74E+01 | | 5.21E+02 ± 2.14E+01 | |
| $f_{28}$ | 3.00E+02 ± 1.01E−12 | + | 6.93E+02 ± 1.80E+02 | – | 3.00E+02 ± 3.67E−13 | + | 3.50E+02 ± 1.11E+02 | – |
| | 3.27E+02 ± 8.73E+01 | | 4.91E+02 ± 2.07E+02 | | 3.48E+02 ± 1.06E+02 | | 3.26E+02 ± 8.64E+01 | |
| *w/l/t* | 7/11/10 | | 0/17/11 | | 6/4/18 | | 0/8/20 | |
| | jDE versus jDE−SaCI | | AS-JADE versus AS-JADE−SaCI | | Rank-jDE versus rank-jDE−SaCI | | CIMDE/rand/1 versus CIMDE/rand/1-SaCI | |
| $f_1$ | 0.00E+00 ± 0.00E+00 | ≈ | 0.00E+00 ± 0.00E+00 | ≈ | 0.00E+00 ± 0.00E+00 | ≈ | 1.26E+04 ± 3.21E+03 | – |
| | 0.00E+00 ± 0.00E+00 | | 0.00E+00 ± 0.00E+00 | | 0.00E+00 ± 0.00E+00 | | 7.45E+00 ± 1.69E+01 | |
| $f_2$ | 1.10E+05 ± 7.50E+04 | – | 9.36E+05 ± 6.51E+05 | – | 7.83E+04 ± 1.05E+05 | – | 1.49E+08 ± 7.16E+07 | – |
| | 9.64E+03 ± 7.66E+03 | | 3.06E+05 ± 2.19E+05 | | 6.83E+03 ± 8.45E+03 | | 6.18E+04 ± 6.61E+04 | |
| $f_3$ | 3.18E+03 ± 4.45E+03 | + | 1.26E+06 ± 2.96E+06 | – | 3.61E+03 ± 1.12E+04 | ≈ | 6.01E+10 ± 2.45E+10 | – |
| | 3.56E+03 ± 7.97E+03 | | 1.38E+02 ± 1.74E+02 | | 2.99E+01 ± 6.83E+01 | | 1.07E+08 ± 1.17E+08 | |
| $f_4$ | 1.11E+03 ± 1.38E+03 | – | 1.75E+04 ± 4.91E+03 | – | 4.94E+02 ± 6.29E+02 | – | 1.69E+05 ± 1.91E+05 | – |
| | 5.39E+01 ± 6.16E+01 | | 7.23E+03 ± 2.46E+03 | | 2.46E+01 ± 2.33E+01 | | 5.30E+02 ± 3.57E+02 | |
| $f_5$ | 0.00E+00 ± 1.25E−13 | ≈ | 7.96E−13 ± 2.19E−12 | ≈ | 0.00E+00 ± 0.00E+00 | ≈ | 4.59E+03 ± 3.30E+03 | – |
| | 0.00E+00 ± 0.00E+00 | | 0.00E+00 ± 0.00E+00 | | 0.00E+00 ± 0.00E+00 | | 4.44E+01 ± 4.33E+01 | |
| $f_6$ | 2.64E+00 ± 6.02E−01 | – | 5.22E+00 ± 7.51E−01 | – | 2.89E+00 ± 2.37E+00 | – | 8.41E+02 ± 3.77E+02 | – |
| | 1.15E+00 ± 1.45E+00 | | 1.39E+00 ± 4.22E−01 | | 1.11E+00 ± 1.50E+00 | | 1.32E+01 ± 1.78E+01 | |
| $f_7$ | 1.20E+00 ± 8.07E−01 | ≈ | 7.23E+00 ± 1.71E+01 | ≈ | 3.92E−01 ± 4.61E−01 | ≈ | 7.63E+02 ± 1.41E+03 | – |
| | 1.93E+00 ± 4.65E+00 | | 5.39E−01 ± 5.47E−01 | | 1.26E+00 ± 3.65E+00 | | 1.89E+01 ± 1.84E+01 | |
| $f_8$ | 2.04E+01 ± 7.69E−02 | ≈ | 2.04E+01 ± 8.68E−02 | ≈ | 2.05E+01 ± 8.36E−02 | ≈ | 2.07E+01 ± 9.17E−02 | – |
| | 2.05E+01 ± 6.66E−02 | | 2.04E+01 ± 6.19E−02 | | 2.05E+01 ± 8.65E−02 | | 2.05E+01 ± 1.28E−01 | |
| $f_9$ | 8.47E+00 ± 6.79E−01 | – | 7.92E+00 ± 7.68E−01 | ≈ | 2.52E+00 ± 1.43E+00 | ≈ | 1.06E+01 ± 1.14E+00 | ≈ |
| | 7.48E+00 ± 8.48E−01 | | 7.98E+00 ± 1.25E+00 | | 1.66E+00 ± 1.46E+00 | | 9.73E+00 ± 5.61E−01 | |

**Table 3** (continued)

| | jDE versus jDE−SaCI | | AS-JADE versus AS-JADE−SaCI | | Rank-jDE versus rank-jDE−SaCI | | CIMDE/rand/1 versus CIMDE/rand/1-SaCI | |
|---|---|---|---|---|---|---|---|---|
| $f_{10}$ | 4.62E−01 ± 8.94E−02 | – | 4.76E−01 ± 1.53E−01 | ≈ | 9.07E−02 ± 7.57E−02 | ≈ | 1.17E+03 ± 3.82E+02 | – |
| | 3.65E−01 ± 8.80E−02 | | 4.04E−01 ± 9.56E−02 | | 8.57E−02 ± 4.42E−02 | | 8.48E+00 ± 1.02E+01 | |
| $f_{11}$ | 3.14E−07 ± 3.98E−07 | ≈ | 5.57E−01 ± 8.52E−01 | + | 9.05E−02 ± 3.00E−01 | ≈ | 2.40E+02 ± 3.54E+01 | – |
| | 2.67E−07 ± 7.17E−07 | | 1.87E+00 ± 1.15E+00 | | 0.00E+00 ± 0.00E+00 | | 1.01E+01 ± 5.32E+00 | |
| $f_{12}$ | 2.40E+01 ± 6.03E+00 | – | 2.48E+01 ± 6.09E+00 | ≈ | 1.07E+01 ± 4.96E+00 | ≈ | 2.06E+02 ± 2.93E+01 | – |
| | 1.83E+01 ± 5.19E+00 | | 2.31E+01 ± 4.46E+00 | | 7.42E+00 ± 2.32E+00 | | 2.23E+01 ± 1.21E+01 | |
| $f_{13}$ | 2.74E+01 ± 5.63E+00 | – | 2.72E+01 ± 4.22E+00 | – | 1.51E+01 ± 5.57E+00 | ≈ | 2.50E+02 ± 4.79E+01 | – |
| | 1.94E+01 ± 3.99E+00 | | 2.09E+01 ± 3.93E+00 | | 1.16E+01 ± 4.45E+00 | | 2.23E+01 ± 1.02E+01 | |
| $f_{14}$ | 1.85E+02 ± 6.80E+01 | ≈ | 5.29E+02 ± 8.34E+01 | ≈ | 7.54E+00 ± 7.77E+00 | ≈ | 2.55E+03 ± 2.00E+02 | – |
| | 1.83E+02 ± 8.43E+01 | | 5.82E+02 ± 9.05E+01 | | 6.91E+00 ± 6.89E+00 | | 1.53E+03 ± 2.21E+02 | |
| $f_{15}$ | 1.57E+03 ± 1.49E+02 | ≈ | 1.61E+03 ± 1.40E+02 | – | 1.06E+03 ± 3.40E+02 | ≈ | 2.50E+03 ± 1.86E+02 | – |
| | 1.57E+03 ± 1.67E+02 | | 1.47E+03 ± 1.45E+02 | | 1.06E+03 ± 3.46E+02 | | 1.61E+03 ± 2.79E+02 | |
| $f_{16}$ | 1.36E+00 ± 4.01E−01 | ≈ | 1.36E+00 ± 4.24E−01 | ≈ | 1.30E+00 ± 3.85E−01 | ≈ | 6.46E−01 ± 6.82E−01 | ≈ |
| | 1.45E+00 ± 3.92E−01 | | 1.27E+00 ± 2.73E−01 | | 1.28E+00 ± 1.45E−01 | | 1.18E+00 ± 2.20E−01 | |
| $f_{17}$ | 1.39E+01 ± 1.08E+00 | ≈ | 1.31E+01 ± 4.78E−01 | + | 1.04E+01 ± 2.14E−01 | ≈ | 3.99E+02 ± 8.12E+01 | – |
| | 1.42E+01 ± 1.44E+00 | | 1.45E+01 ± 1.28E+00 | | 1.04E+01 ± 1.85E−01 | | 3.25E+01 ± 6.48E+00 | |
| $f_{18}$ | 3.90E+01 ± 3.26E+00 | ≈ | 3.54E+01 ± 5.05E+00 | ≈ | 2.47E+01 ± 5.04E+00 | + | 3.86E+02 ± 8.54E+01 | – |
| | 3.94E+01 ± 4.48E+00 | | 3.83E+01 ± 4.16E+00 | | 3.23E+01 ± 9.21E+00 | | 4.05E+01 ± 8.21E+00 | |
| $f_{19}$ | 1.04E+00 ± 1.58E−01 | ≈ | 1.37E+00 ± 3.95E−01 | ≈ | 4.60E−01 ± 1.98E−01 | + | 4.94E+04 ± 5.18E+04 | – |
| | 9.66E−01 ± 1.65E−01 | | 1.43E+00 ± 2.27E−01 | | 6.83E−01 ± 2.07E−01 | | 2.02E+00 ± 7.92E−01 | |
| $f_{20}$ | 3.54E+00 ± 1.82E−01 | – | 3.39E+00 ± 3.54E−01 | ≈ | 3.07E+00 ± 3.21E−01 | ≈ | 4.83E+00 ± 1.42E−01 | – |
| | 3.21E+00 ± 2.61E−01 | | 3.07E+00 ± 3.52E−01 | | 3.01E+00 ± 2.79E−01 | | 3.44E+00 ± 4.18E−01 | |
| $f_{21}$ | 3.64E+02 ± 8.10E+01 | ≈ | 4.00E+02 ± 2.38E−13 | ≈ | 3.82E+02 ± 6.04E+01 | ≈ | 1.15E+03 ± 1.19E+02 | – |
| | 4.00E+02 ± 2.38E−13 | | 4.00E+02 ± 2.38E−13 | | 4.00E+02 ± 2.38E−13 | | 3.76E+02 ± 6.11E+01 | |
| $f_{22}$ | 4.86E+02 ± 1.47E+02 | ≈ | 7.86E+02 ± 1.25E+02 | ≈ | 7.11E+01 ± 4.36E+01 | ≈ | 2.77E+03 ± 1.86E+02 | – |
| | 4.28E+02 ± 1.19E+02 | | 6.60E+02 ± 2.01E+02 | | 1.01E+02 ± 9.79E+01 | | 1.58E+03 ± 1.81E+02 | |

**Table 3** (continued)

| | jDE versus jDE−SaCI | | AS-JADE versus AS-JADE−SaCI | | Rank-jDE versus rank-jDE−SaCI | | CIMDE/rand/1 versus CIMDE/rand/1-SaCI | |
|---|---|---|---|---|---|---|---|---|
| $f_{23}$ | 1.75E+03 ± 2.52E+02 | ≈ | 1.80E+03 ± 1.54E+02 | ≈ | 1.08E+03 ± 2.70E+02 | ≈ | 2.75E+03 ± 1.73E+02 | – |
| | 1.61E+03 ± 2.53E+02 | | 1.73E+03 ± 1.39E+02 | | 1.06E+03 ± 4.90E+02 | | 1.69E+03 ± 1.04E+02 | |
| $f_{24}$ | 2.22E+02 ± 2.13E+00 | – | 2.23E+02 ± 1.61E+00 | ≈ | 2.10E+02 ± 5.56E+00 | – | 2.32E+02 ± 2.42E+00 | – |
| | 2.19E+02 ± 2.81E+00 | | 2.23E+02 ± 1.58E+00 | | 2.05E+02 ± 3.14E+00 | | 2.26E+02 ± 6.75E−01 | |
| $f_{25}$ | 2.23E+02 ± 1.20E+00 | – | 2.23E+02 ± 2.11E+00 | ≈ | 2.13E+02 ± 4.95E+00 | ≈ | 2.29E+02 ± 2.98E+00 | – |
| | 2.20E+02 ± 2.43E+00 | | 2.23E+02 ± 1.76E+00 | | 2.12E+02 ± 6.10E+00 | | 2.24E+02 ± 2.64E+00 | |
| $f_{26}$ | 2.00E+02 ± 3.20E−02 | – | 1.91E+02 ± 2.72E+01 | ≈ | 2.00E+02 ± 6.44E−03 | – | 3.01E+02 ± 4.07E+01 | – |
| | 1.86E+02 ± 3.11E+01 | | 2.00E+02 ± 4.57E−01 | | 1.76E+02 ± 4.19E+01 | | 2.16E+02 ± 5.94E+01 | |
| $f_{27}$ | 5.31E+02 ± 1.42E+01 | – | 5.25E+02 ± 2.13E+01 | ≈ | 4.13E+02 ± 6.41E+01 | ≈ | 6.21E+02 ± 2.15E+01 | – |
| | 4.93E+02 ± 1.91E+01 | | 5.29E+02 ± 1.65E+01 | | 4.04E+02 ± 5.31E+01 | | 5.45E+02 ± 2.20E+01 | |
| $f_{28}$ | 3.00E+02 ± 2.79E−09 | + | 2.84E+02 ± 6.18E+01 | ≈ | 2.82E+02 ± 6.03E+01 | ≈ | 1.67E+03 ± 1.40E+02 | – |
| | 3.23E+02 ± 7.53E+01 | | 3.00E+02 ± 4.13E−12 | | 2.82E+02 ± 6.03E+01 | | 4.81E+02 ± 1.50E+02 | |
| *w/l/t* | 2/12/14 | | 2/6/20 | | 2/5/21 | | 0/26/2 | |

jDE-SaCI. For another instance, DE/best/1-SaCI had outperformed DE/best/1 on 17 test functions ($f_1$,$f_7$,$f_{12}$,$f_{13}$,$f_{16}$,$f_{17}$,$f_{18}$,$f_{19}$,$f_{20}$,$f_{22}$,$f_{23}$,$f_{25}$,$f_{26}$,$f_{27}$,$f_{28}$), and DE/best/1-SaCI was defeated on no test functions. For the other benchmark functions (11 out of 28 cases), there were no significant differences between DE/best/1 and DE/best/1-SaCI. In general, the proposed SaCI-based mutation strategy may improve the performances of original algorithms effectively. Further, we would like to show some median convergence curves of all compared algorithms, shown in Fig. 1.

From the above figures, we could see: (1) in most cases, the algorithm with SaCI mutation operator did not deteriorate the searching capability. Usually, better convergence performance could be resulted on some test functions. (2) CIMDE/rand/1 did not perform well on most test functions, while DIMDE/rand/1-SaCI could result competitive convergence curves on most benchmark functions. That means that the proposed hybrid mutation operator and greedy selection operator are effective. All the above figures could demonstrate the comprehensive results in Table 2 vividly.

## 4.2 Influence of population size

In this section, in order to make the scope of this paper focused, we would like to try to research the influence of population size on the proposed SaCI mutation operator. Here, we choose jDE-SaCI as the paradigm to test its performances when $NP=25$, 50, 75, 100, respectively. Over 30 independent runs are conducted on CEC 2013 benchmark functions at $D=10$. Like the previous experiment, Max_FEs is set to $3000D$. The experimental results are shown in Table 4. Moreover, in order to show the overall rankings of jDE-SaCI with different population sizes, Friedman test [10, 11] is conducted at $\alpha=0.05$ in MATLAB environment. The $p$ values for all functions are shown in Table 4.

From $p$ values in Table 4, we can see $NP$ did not influence the performance of jDE-SaCI on 6 out of 28 test functions ($f_2$, $f_7$, $f_8$, $f_{16}$, $f_{18}$, $f_{26}$). For other test functions, $NP$ did influence the searching capability of jDE-SaCI. Besides, jDE-SaCI with $NP=25$ and $NP=50$ usually could obtain outstanding results compared with larger $NP$ settings. Therefore, in this paper, it may be acceptable that population size is set to 50.

## 4.3 Analysis of extra time cost

In order to compare the computational complexities of the involved algorithms fairly, we would conduct a test program to calculate time consumed on this running platform. The mentioned calculation method for algorithm complexity was suggested by technical report for the CEC 2013 Special Session on Real-Parameter Optimization, which could be download via http://www3.ntu.edu.sg/home/epnsugan/. In this section, $NP$ is set to 50. The test steps are listed as follows:

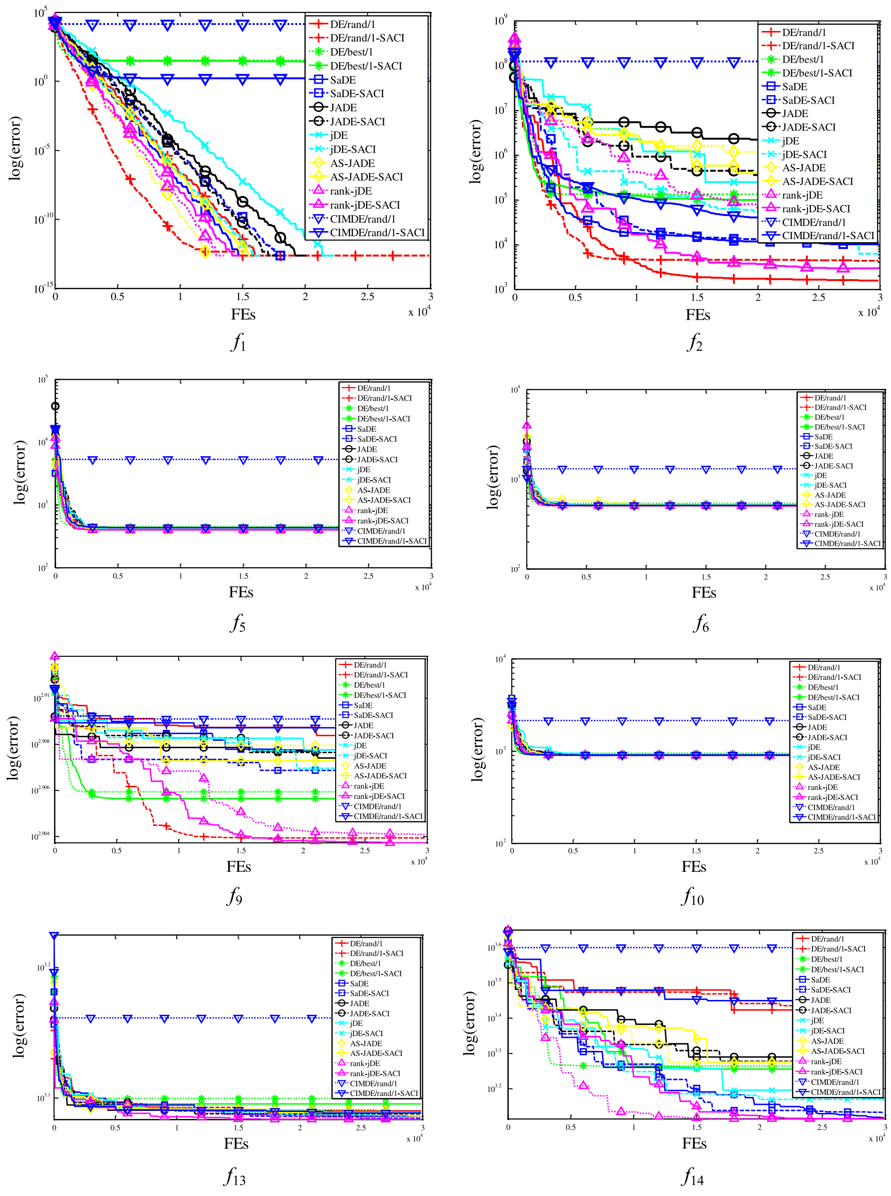

**Fig. 1** Half median convergence curves of all the compared algorithms

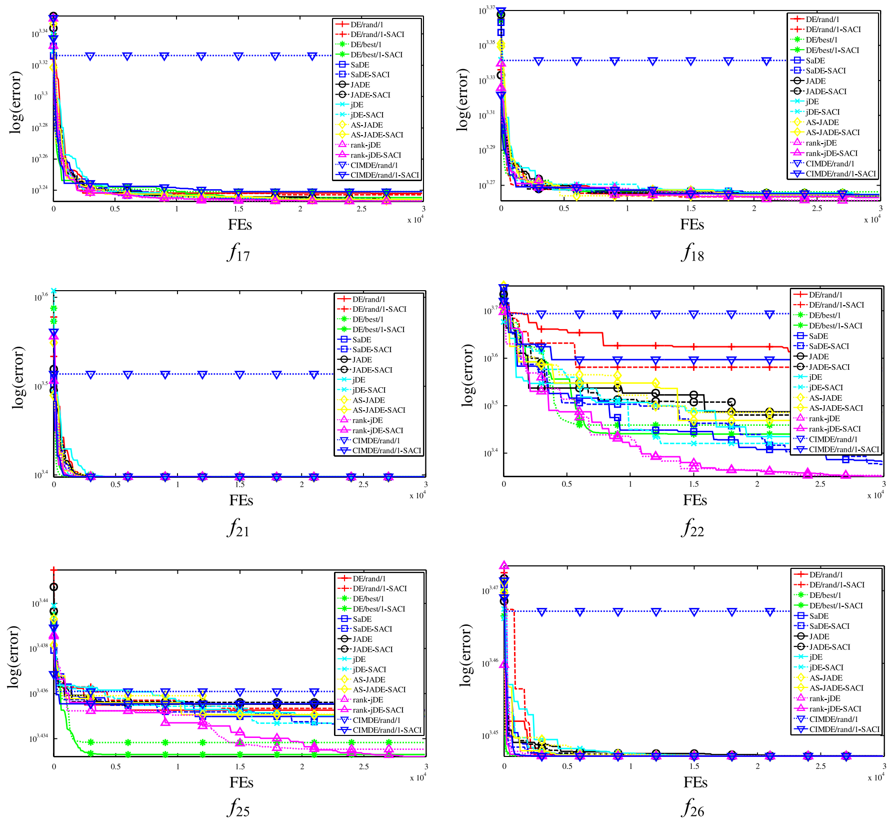

**Fig. 1** (continued)

Step 1: Run the MATLAB program below:

```
for i = 1 : 10,000,00
 x = 0.55 + i;
 x = x + x;
 x = x./2;
 x = x * x;
 x = sqrt (x);
 x = log (x);
 x = exp (x);
 y = x/x;
end
```

Computation time for the above $= T_0$;
Step 2: Evaluate the computation time just for $f_{14}$. For 200,000 evaluations of a certain dimension $D$, it gives $T_1$;

**Table 4** Compared results and Friedman test of jDE-SaCI with different population sizes for $D = 10$

| | $NP = 25$ | $NP = 50$ | $NP = 75$ | $NP = 100$ | $p$ value |
|---|---|---|---|---|---|
| $f_1$ | **0.00E+00 ± 0.00E+00** | **0.00E+00 ± 0.00E+00** | **0.00E+00 ± 0.00E+00** | 2.05E−12 ± 1.66E−12 | 3.22E−07 |
| $f_2$ | 2.60E+04 ± 4.38E+04 | **9.64E+03 ± 7.66E+03** | 1.91E+04 ± 1.44E+04 | 4.46E+04 ± 2.06E+04 | 1.32E−02 |
| $f_3$ | 3.46E+04 ± 1.12E+05 | 3.56E+03 ± 7.97E+03 | **1.11E+03 ± 1.25E+03** | 3.69E+04 ± 3.12E+04 | 1.97E−04 |
| $f_4$ | 9.57E+01 ± 1.31E+02 | **5.39E+01 ± 6.16E+01** | 1.88E+02 ± 1.58E+02 | 3.95E+02 ± 1.19E+02 | 6.83E−04 |
| $f_5$ | **0.00E+00 ± 0.00E+00** | **0.00E+00 ± 0.00E+00** | 8.64E−12 ± 9.02E−12 | 1.11E−08 ± 5.06E−09 | 3.22E−07 |
| $f_6$ | **3.75E−01 ± 1.20E+00** | 1.15E+00 ± 1.45E+00 | 1.96E+00 ± 6.02E−01 | 3.81E+00 ± 1.33E+00 | 4.33E−05 |
| $f_7$ | 7.01E+00 ± 9.82E+00 | 1.93E+00 ± 4.65E+00 | **1.72E+00 ± 1.46E+00** | 2.03E+00 ± 1.04E+00 | 6.42E−02 |
| $f_8$ | 2.05E+01 ± 8.35E−02 | 2.05E+01 ± 6.66E−02 | 2.05E+01 ± 1.13E−01 | **2.04E+01 ± 1.08E−01** | 8.19E−01 |
| $f_9$ | **3.72E+00 ± 2.50E+00** | 7.48E+00 ± 8.48E−01 | 7.60E+00 ± 8.66E−01 | 7.66E+00 ± 6.94E−01 | 5.31E−03 |
| $f_{10}$ | **8.18E−02 ± 4.51E−02** | 3.65E−01 ± 8.80E−02 | 4.03E−01 ± 1.07E−01 | 5.16E−01 ± 1.08E−01 | 2.43E−05 |
| $f_{11}$ | **0.00E+00 ± 0.00E+00** | 2.67E−07 ± 7.17E−07 | 1.74E+00 ± 9.87E−01 | 4.35E+00 ± 1.14E+00 | 3.22E−07 |
| $f_{12}$ | **1.08E+01 ± 6.56E+00** | 1.83E+01 ± 5.19E+00 | 2.38E+01 ± 5.45E+00 | 2.28E+01 ± 2.82E+00 | 2.88E−03 |
| $f_{13}$ | **1.41E+01 ± 7.74E+00** | 1.94E+01 ± 3.99E+00 | 2.52E+01 ± 4.65E+00 | 2.34E+01 ± 3.38E+00 | 1.97E−02 |
| $f_{14}$ | **2.50E+01 ± 1.42E+01** | 1.83E+02 ± 8.43E+01 | 2.94E+02 ± 8.43E+01 | 3.66E+02 ± 1.07E+02 | 2.43E−05 |
| $f_{15}$ | **1.28E+03 ± 1.98E+02** | 1.57E+03 ± 1.67E+02 | 1.62E+03 ± 2.42E+02 | 1.50E+03 ± 1.91E+02 | 6.18E−03 |
| $f_{16}$ | 1.56E+00 ± 2.50E−01 | **1.45E+00 ± 3.92E−01** | 1.50E+00 ± 2.50E−01 | 1.63E+00 ± 1.73E−01 | 6.64E−01 |
| $f_{17}$ | **1.12E+01 ± 2.60E−01** | 1.42E+01 ± 1.44E+00 | 1.63E+01 ± 8.73E−01 | 1.67E+01 ± 1.59E+00 | 7.66E−06 |
| $f_{18}$ | **3.77E+01 ± 4.26E+00** | 3.94E+01 ± 4.48E+00 | 4.21E+01 ± 2.79E+00 | 4.21E+01 ± 3.97E+00 | 9.44E−02 |
| $f_{19}$ | **8.40E−01 ± 1.66E−01** | 9.66E−01 ± 1.65E−01 | 1.06E+00 ± 7.24E−02 | 1.23E+00 ± 1.88E−01 | 2.18E−04 |
| $f_{20}$ | 3.29E+00 ± 5.02E−01 | **3.21E+00 ± 2.61E−01** | 3.50E+00 ± 2.66E−01 | 3.50E+00 ± 3.68E−01 | 2.52E−02 |
| $f_{21}$ | 3.91E+02 ± 3.02E+01 | 4.00E+02 ± 2.38E−13 | **3.82E+02 ± 6.04E+01** | 4.00E+02 ± 2.12E−12 | 1.15E−06 |
| $f_{22}$ | **2.00E+02 ± 1.13E+02** | 4.28E+02 ± 1.19E+02 | 5.66E+02 ± 1.23E+02 | 5.98E+02 ± 1.57E+02 | 2.98E−04 |
| $f_{23}$ | **1.48E+03 ± 2.24E+02** | 1.61E+03 ± 2.53E+02 | 1.75E+03 ± 1.37E+02 | 1.76E+03 ± 1.98E+02 | 7.20E−03 |
| $f_{24}$ | **2.11E+02 ± 6.19E+00** | 2.19E+02 ± 2.81E+00 | 2.21E+02 ± 1.77E+00 | 2.20E+02 ± 4.93E+00 | 1.14E−03 |
| $f_{25}$ | **2.14E+02 ± 6.53E+00** | 2.20E+02 ± 2.43E+00 | 2.22E+02 ± 2.27E+00 | 2.22E+02 ± 2.28E+00 | 2.74E−03 |
| $f_{26}$ | **1.66E+02 ± 4.68E+01** | 1.86E+02 ± 3.11E+01 | 1.88E+02 ± 2.64E+01 | 1.98E+02 ± 5.57E+00 | 1.83E−01 |
| $f_{27}$ | **3.50E+02 ± 5.45E+01** | 4.93E+02 ± 1.91E+01 | 4.95E+02 ± 2.68E+01 | 5.12E+02 ± 1.80E+01 | 6.58E−05 |
| $f_{28}$ | 3.11E+02 ± 1.18E+02 | 3.23E+02 ± 7.53E+01 | **2.82E+02 ± 6.03E+01** | **2.82E+02 ± 6.03E+01** | 1.73E−03 |

Best results are shown in boldface

Step 3: The complete computation time for the algorithm with 200,000 evaluations of the same $D$-dimensional benchmark function $f_{14}$ is $T_2$.
Step 4: Execute step 3 for five times and get 5 $T_2$ values. $T'_2 = \text{mean}\ (T_2)$;
Step 5: The complexity of the algorithm is reflected by: $T'_2$, $T_1$, $T_0$, and $(T'_2 - T_1)/T_0$.

The compared results are shown in Table 5. Five SaCI mutation operators-based DE algorithms, which are DE/best/1-SaCI, SaDE-SaCI, JADE-SaCI, AS-JADE-SaCI and CIMDE/rand/1-SaCI, spent less time cost than the corresponding algorithms. It may mean that the proposed mutation operator could speed up the searching process and avoid using the original complex genetic operations, while the other three SaCI-based DE algorithms spent more time than the original algorithms. In general, the presented SaCI mutation operator did not deteriorate the calculation efficiency of the selected DE algorithms.

**Table 5** Comparison results of time complexity calculation for all the algorithms (s)

| | DE/rand/1 | DE/rand/1-SaCI | DE/best/1 | DE/best/1-SaCI | SaDE | SaDE-SaCI | JADE | JADE-SaCI |
|---|---|---|---|---|---|---|---|---|
| $T_0$ | 0.1300 | | | | | | | |
| $T_1$ | 2.1000 | 1.9400 | 1.9400 | 1.9700 | 1.9470 | 1.9600 | 1.9200 | 1.9200 |
| $T'_2$ | 5.8580 | 6.9560 | 6.5076 | 6.9122 | 16.2710 | 13.4920 | 18.6740 | 13.7280 |
| $(T'_2-T_1)/T_0$ | 28.9077 | 38.5846 | 35.1354 | 33.3932 | 97.4422 | 82.3714 | 119.6714 | 78.7200 |
| | jDE | jDE-SaCI | AS-JADE | AS-JADE-SaCI | rank-jDE | rank-jDE-SaCI | CIMDE/rand/1 | CIMDE/rand/1-SaCI |
| $T_0$ | 0.1300 | | | | | | | |
| $T_1$ | 1.9200 | 1.9400 | 1.9100 | 1.8800 | 1.8900 | 1.9300 | 1.9000 | 1.9400 |
| $T'_2$ | 4.4240 | 6.7520 | 22.3780 | 15.4740 | 7.4960 | 9.3022 | 19.7700 | 13.9300 |
| $(T'_2-T_1)/T_0$ | 19.2615 | 37.0154 | 136.4533 | 90.6267 | 43.1231 | 56.7092 | 127.6429 | 74.9375 |

## 5 Conclusions and future work

In this paper, we have proposed a novel mutation strategy which utilizes CI theory for DE variants. Mainly, the evolutionary information of *m* best target vectors is linearly combined to generate new mutant vectors. Besides, *m* is designed as an exponential-distributed random number which could be self-adapted based on successful records of *m* values alongside evolution. Moreover, this mutation operator could be applied to any DE algorithm without destroying existing search capability by adding a greedy selection operator. Comprehensive experimental results demonstrate that the proposed mutation strategy could improve the overall performances of some state-of-the-art DE variants effectively.

In the future, we try to expand the application of this SaCI-based mutation strategy to more versatile DE methods. Meanwhile, it is attractive to apply such SaCI-based mutation strategy for various DE algorithms on real engineering problems such as parameters identification of remote operated vehicle.

**Acknowledgements** The authors would like to thank Natural Science Foundation of China under Contract No. 51709027, 51506019; Natural Science Foundation of Liaoning Province, China under Contract No. 2014025006; Education Department General Project of Liaoning Province, China under Contract No. L2014209; Doctoral Scientific Research Foundation Project of Liaoning Province, China under Contract No. 20170520265; Yong Elite Scientists Sponsorship Program By CAST under Contract No. 2016QNRC001 for financially supporting this research; Fundamental Research Funds for the Central Universities under Contract No. 3132019007.

### Compliance with ethical standards

**Conflict of interest** The authors declare that they have no conflict of interest.

## References

1. Al-Ani A, Alsukker A, Khushaba RN (2013) Feature subset selection using differential evolution and a wheel based search strategy. Swarm Evolut Comput 9:15–26
2. Avlonitis M, Karydis I, Sioutas S (2015) Early prediction in collective intelligence on video users' activity. Inf Sci 298:315–329
3. Brest J, Greiner S, Boskovic B, Mernik M, Zumer V (2006) Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans Evol Comput 10(6):646–657
4. Cai HR, Chung CY, Wong KP (2008) Application of differential evolution algorithm for transient stability constrained optimal power flow. IEEE Trans Power Syst 23(2):719–728
5. Cui L, Li G, Lin Q, Chen J, Lu N (2016) Adaptive differential evolution algorithm with novel mutation strategies in multiple sub-populations. Comput Oper Res 67:155–173
6. Das S, Abraham A, Konar A (2008) Automatic clustering using an improved differential evolution algorithm. IEEE Trans Syst Man Cybern Part A Syst Hum 38(1):218–237
7. Das S, Suganthan PN (2011) Differential evolution: a survey of the state-of-the-art. IEEE Trans Evol Comput 15(1):4–31
8. Dash R, Dash PK, Bisoi R (2014) A self adaptive differential harmony search based optimized extreme learning machine for financial time series prediction. Swarm Evolut Comput 19:25–42
9. de-los-Cobos-Silva S, Mora-Gutierrez RA, Gutierrez-Andrade MA, Rincon-Garcia EA, Ponsich A, Lara-Velazquez P (2018) Development of seven hybrid methods based on collective intelligence for solving nonlinear constrained optimization problems. Artif Intell Rev 49(2):245–279

10. Derrac J, Garcia S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evolut Comput 1(1):3–18
11. Garcia S, Fernandez A, Luengo J, Herrera F (2010) Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power. Inf Sci 180(10):2044–2064
12. Gong W, Cai Z (2013) Differential evolution with ranking-based mutation operators. IEEE Trans Cybern 43(6):2066–2081
13. Gong W, Fialho A, Cai Z, Li H (2011) Adaptive strategy selection in differential evolution for numerical optimization: an empirical study. Inf Sci 181(24):5364–5386
14. Islam SM, Das S, Ghosh S, Roy S, Suganthan PN (2012) An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization. IEEE Trans Syst Man Cybern Part B Cybern 42(2):482–500
15. Jha DK, Chattopadhyay P, Sarkar S, Ray A (2016) Path planning in GPS-denied environments via collective intelligence of distributed sensor networks. Int J Control 89(5):984–999
16. Petrillo F, Gueheneuc YG, Pimenta M, Freitas CD, Khomh F (2019) Swarm debugging: the collective intelligence on interactive debugging. J Syst Softw 153:152–174
17. Qin AK, Huang VL, Suganthan PN (2009) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evol Comput 13(2):398–417
18. Schut MC (2010) On model design for simulation of collective intelligence. Inf Sci 180(1):132–155
19. 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
20. Tauscher K (2017) Leveraging collective intelligence: how to design and manage crowd-based business models. Bus Horiz 60(2):237–245
21. Villarreal-Cervantes MG, Alvarez-Gallegos J (2016) Off-line PID control tuning for a planar parallel robot using DE variants. Expert Syst Appl 64:444–454
22. Wang C, Liu Y, Liang X, Guo H, Chen Y, Zhao Y (2018) Self-adaptive differential evolution algorithm with hybrid mutation operator for parameters identification of PMSM. Soft Comput 22(4):1263–1285
23. 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
24. Wu G, Mallipeddi R, Suganthan PN, Wang R, Chen H (2016) Differential evolution with multi-population based ensemble of mutation strategies. Inf Sci 329:329–345
25. Yang B, Yu T, Zhang XS, Li HF, Shu HC, Sang YY, Jiang L (2019) Dynamic leader based collective intelligence for maximum power point tracking of PV systems affected by partial shading condition. Energy Convers Manag 179:286–303
26. Yang GY, Dong ZY, Wong KP (2008) A modified differential evolution algorithm with fitness sharing for power system planning. IEEE Trans Power Syst 23(2):514–522
27. Zhang JQ, Sanderson AC (2009) JADE: adaptive differential evolution with optional external archive. IEEE Trans Evol Comput 13(5):945–958
28. Zheng LM, Zhang SX, Tang KS, Zheng SY (2017) Differential evolution powered by collective information. Inf Sci 399:13–29