



# Self-adaptive opposition-based differential evolution with subpopulation strategy for numerical and engineering optimization problems

Jiahang Li<sup>1,2</sup> · Yuelin Gao<sup>1,2</sup> · Hang Zhang<sup>2,3</sup> · Qinwen Yang<sup>2,3</sup>

Received: 12 April 2021 / Accepted: 18 March 2022 / Published online: 4 May 2022  
© The Author(s) 2022

## Abstract

Opposition-based differential evolution (ODE) is a well-known DE variant that employs opposition-based learning (OBL) to accelerate the convergence speed. However, the existing OBL variants are population-based, which causes many shortcomings. The value of the jumping rate is not self-adaptively adjusted, so the algorithm easily traps into local optima. The population-based OBL wastes fitness evaluations when the algorithm converges to sub-optimal. In this paper, we proposed a novel OBL called subpopulation-based OBL (SPOBL) with a self-adaptive parameter control strategy. In SPOBL, the jumping rate acts on the individual, and the subpopulation is selected according to the individual's jumping rate. In the self-adaptive parameter control strategy, the surviving individual's jumping rate in each iteration will participate in the self-adaptive process. A generalized Lehmer mean is introduced to achieve an equilibrium between exploration and exploitation. We used DE and advanced DE variants combined with SPOBL to verify performance. The results of performance are evaluated on the CEC 2017 and CEC 2020 test suites. The SPOBL shows better performance compared to other OBL variants in terms of benchmark functions as well as real-world constrained optimization problems.

**Keywords** Differential evolution · Opposition-based learning · Subpopulation-based · Evolutionary algorithms · Generalized Lehmer mean · Real-world optimization problems

**Arabic keywords** تيمزراوخ روطت · تيلضافتلا · تيجيتارتسا ملعتلا · يسكعلا ةعومجم تيعرف · تيمزراوخلا · تيروطتلا ممعلا رميل · ينعي ةلكشم · نيسحتلا · تيعفاو

## Introduction

Many challenging problems arise in real-world applications [1–5]. Single-objective optimization algorithms play an essential role in many fields. Although some real-world problems can be solved, they are limited by the lack of

computational power, and it is crucial to design an efficient algorithm. Bio-inspired computation can easily solve these problems. It is mainly divided into two categories: Evolutionary Computation (EC) and Swarm Intelligence (SI) [6]. Differential Evolution (DE) is the most outstanding evolutionary algorithm.

DE is a simple yet powerful population-based evolutionary algorithm for real-parameter optimization problems in the continuous domain, proposed by Storn and Price [7]. It has the advantages of simplicity, effectiveness, stability, etc. [8], and is widely used in real-world optimization problems [9], engineering applications [10], pattern recognition [11], feature selection [12], machine learning and other fields [13]. DE and its variants are used to solve complex problems. The common issues of DE are premature convergence and stagnation. These issues are closely related to population diversity. If the population diversity is very small, this means that the

✉ Yuelin Gao  
gaoyuelin@263.net

Jiahang Li  
13704767290@163.com

<sup>1</sup> School of Mathematics and Information Science, North Minzu University, Yinchuan 750021, China

<sup>2</sup> Ningxia Province Key Laboratory of Intelligent Information and Data Processing, Yinchuan 750021, China

<sup>3</sup> School of Computer Science and Engineering, North Minzu University, Yinchuan 750021, China

DE converges very fast. Otherwise, if the population diversity is consistently big, DE will not converge to the global optimal. Hence, it is essential to find a trade-off between exploration and exploitation [14].

Since DE was proposed, many techniques have been combined with DE to improve its performance [9,10]. In the past decade, among different kinds of DE variants, opposition-based DE (ODE) was one of the best [15]. ODE employed OBL on population initialization and generation jumping [16]. DE employed OBL to accelerate the convergence speed by comparing the fitness of a candidate to its opposite and choosing the fitter one. The mathematical proof of OBL was given in [17–19]. Different opposite solutions are always closer to the global optimal with higher probability than random solutions. The issue of OBL variants has received considerable critical attention. Since then, many researchers have designed OBL variants to enhance the performance of soft computing algorithms [20–22]. To save the number of fitness evaluations, Rahnamayan et al. proposed OBL with protective generation jumping (OBLPGJ) to stop the opposition if the opposite solution's success rate drops to a fixed threshold [23]. Rahnamayan et al. proposed quasi-opposition DE (QODE) [24]. In QODE, the opposite point was mapped to an interval. QODE reduced the expected distance between the optimum and its opposite. Ergezer et al. proposed quasi-reflection ODE (QRODE), which was the most cost effective opposition method with the highest probability of being closer to the optimal [18]. Wang et al. proposed generalized opposition-based DE (GODE) and successfully applied it to large-scale optimization problems [25]. Xu et al. proposed current-optimum-ODE (COODE) based on the optimal of the current population [26]. Seif et al. proposed the opposition-based metaheuristic optimization algorithm (OBA), which introduced extended opposition and reflected extended opposition [27]. Most studies in OBL have only focused on the upper and lower bounds of the current search space. In [28], centroid opposition-based DE (CODE) was proposed, which computed opposition by involving the entire population. In [29], Gaussian distribution-based opposition was presented to solve the inverse problem of structural damage assessment. This opposition aimed to enhance the exploration ability during both the initialization and iteration process. Xu et al. proposed opposition-mutual learning using mutual learning to guarantee the trade-off between exploration and exploitation [30].

In the opposition, some opposite solutions are randomly mapped to the current search space [18,24,27], some are calculated according to the upper and lower bounds of the current search space [15,25], some are calculated according to the optimal of the current population [26], and some are calculated based on entire population [28]. The above OBL variants are all population-based, which calculate opposite solutions by entire population information if the condition of

jumping rate is satisfied. No previous study has investigated subpopulation-based opposition. The following drawbacks affect OBL variants: (1) As the OBL variant follows a population-based strategy, fitness evaluations can be wasted if opposite individuals no longer enter the next generation during the optimization process [31]. (2) Up to now, far too little attention has been paid to parameter control strategy in OBL community.

This paper proposes a subpopulation-based OBL (SPOBL) with a self-adaptive parameter control strategy that mitigates all the limitations. Instead of using a population-based opposition, we employed a subpopulation-based opposition to reduce the waste of fitness evaluations. We found that the self-adaptive strategy is introduced by assigning jumping rate to each individual, it is possible to select some individuals as a subpopulation to perform the opposition. In the self-adaptive strategy, the subpopulation size is controlled to balance between exploration and exploitation during the evolutionary process. A generalized Lehmer mean is proposed, and the subpopulation size is dynamically controlled by a new parameter  $p$ . We carried out experiments on the CEC 2017 and CEC 2020 test suites [32,33]. The results of 29 benchmark functions and 3 real-world problems confirm the excellent performance of SPOBL compared to eight state-of-the-art OBL variants. The main contributions of this paper are as follows:

1. A new OBL variant called SPOBL is proposed, which is competitive with eight state-of-the-art OBL variants.
2. The novel SPOBL can be combined with any DE variant and shows superior performance.
3. SPOBL not only performs well on benchmark functions, but also outperforms other algorithms in dealing with real-world optimization problems.

The rest of this paper is organized as follows. The next section reviews the canonical DE algorithm and OBL variants. The subsequent section introduces SPOBL and integrates it with DE. The experimental studies are shown next. Finally, the conclusions are given.

## Preliminaries

### Canonical DE algorithm

#### Population initialization

The search space optimized by a real function is  $[a, b]^D$ , where  $D$  is the dimension of the optimized function.  $a$  is the lower bound of search space.  $b$  is the upper bound of the

search space. The population of DE is initialized as

$$X_{i,j}^0 = a + (b - a) \cdot \text{rand}(0, 1), \tag{1}$$

where  $\text{rand}(0, 1)$  is a random number whose value is uniformly distributed from  $(0,1)$ . The size of  $X^0$  is  $D \times NP$ .  $NP$  is the population size. The population is expressed as  $X^G = \{\mathbf{x}_i^G | i = 1, 2, \dots, NP\}$ ,  $\mathbf{x}_i^G = \{x_{i,j}^G | j = 1, 2, \dots, D, x_{i,j}^G \in [a, b]\}$ ,  $G$  represents the number of iterations.

### Mutation

After the population  $X^G = \{\mathbf{x}_i^G | i = 1, 2, \dots, NP\}$  is initialized. Mutation operator is performed on the target vector  $\mathbf{x}_i^G = \{x_{i,j}^G | j = 1, 2, \dots, D\}$ . DE/rand/1 mutation strategy is as follows:

DE/rand/1

$$\mathbf{v}_i = \mathbf{x}_{r_1} + F \cdot (\mathbf{x}_{r_2} - \mathbf{x}_{r_3}), \tag{2}$$

where the indexes  $r_1 \neq r_2 \neq r_3 \neq i$  are numbers randomly selected from  $[1, NP]$ .  $F$  is a real positive control parameter scaling difference vector. The value of  $F$  is bigger, the diversity of the population is higher. The convergence speed is faster due to the value of  $F$  being smaller.

### Crossover

To maintain the population diversity after mutation, the donor vector  $\mathbf{v}_i$  and the target vector  $\mathbf{x}_i$  are crossed under the probability of  $CR$ . The specific operations are as follows:

$$u_{i,j} = \begin{cases} v_{i,j} & \text{if } \text{rand}(0,1) \leq CR, \text{ or, } j = j_{\text{rand}} \\ x_{i,j} & \text{otherwise,} \end{cases} \tag{3}$$

where  $CR \in [0, 1]$  is the crossover rate,  $j_{\text{rand}}$  is a random integer from  $[1, D]$ . The donor vector  $\mathbf{v}_i$  differs from its target vector  $\mathbf{x}_i$  due to the use of  $j_{\text{rand}}$ .  $CR$  controls the number of variables from  $\mathbf{v}_i$  to  $\mathbf{u}_i$ , so the larger value of  $CR$  maintains the population diversity.

### Selection

Selection is an approach after mutation and crossover. The trail vector  $\mathbf{u}_i$  (offspring) should make a one-to-one selection with their corresponding target vector  $\mathbf{x}_i$  (parent). The specific selection formula is as follows:

$$\mathbf{x}_i^{G+1} = \begin{cases} \mathbf{u}_i^G & \text{if } \text{fitness}(\mathbf{u}_i^G) < \text{fitness}(\mathbf{x}_i^G) \\ \mathbf{x}_i^G & \text{otherwise,} \end{cases} \tag{4}$$

where  $\text{fitness}(\cdot)$  denotes the fitness function to be minimized. Therefore, if the new trial vector produces a value equal to

or less than the objective function, it will replace the current generation’s corresponding target vector.

### Opposition-based DE

OBL was presented as a machine learning scheme for reinforcement learning [16]. In the past 15 years, various metaheuristic algorithms have been enhanced by OBL and its variants. The primary purpose behind OBL is to consider the fitness values between candidate and opposite candidate simultaneously to obtain a better approximation of the current best candidate [15,19,20].

Let  $P = (x_1, x_2, \dots, x_D)$  be a point in  $D$ -dimensional space, where  $x_j \in [a_j, b_j]$ ,  $j = 1, 2, \dots, D$ . The opposite point  $\check{P} = (\check{x}_1, \check{x}_2, \dots, \check{x}_D)$  is defined by its segments.

$$\check{x}_j = a_j + b_j - x_j \tag{5}$$

### Opposition-based population initialization

Due to a lack of prior knowledge, we can only generate the opposite population based on the initial population. The specific steps are as follows:

- Use (1) to generate an initial population  $X$ .
- Use (5) to generate the opposite population  $O_{i,j} = a_j + b_j - X_{i,j}$ ,  $i = 1, 2, \dots, NP$ ,  $j = 1, 2, \dots, D$ , where  $O_{i,j}$  and  $X_{i,j}$  respectively represent the  $j$ th dimensional component of the  $i$ th vector of the opposite population and the original population.
- Select the first  $NP$  individuals with the best fitness value from the  $\{X \cup O\}$  to form the initial population.

### Opposition-based generation jumping

In the same way, OBL has been introduced into the evolutionary process. The population can quickly converge to the local optimum during the evolutionary process. The value of the jumping rate  $J$  is used to control the number of times the OBL is executed during the evolutionary process. After a population has completed mutation, crossover and selection, it is determined whether the population performs opposition based on the jumping rate. Unlike the population initialization, in generation jumping, the bound of the opposite individual changes with the iteration progress, and the opposite individual is calculated based on dynamic bound.

$$O_{i,j} = L_j + U_j - X_{i,j}, \tag{6}$$

where  $L_j$  and  $U_j$  are the lower bound and upper bound of  $j$ th variable in the current population.

### OBL variants

In [24], QOBL was proposed, quasi opposite point was introduced as a uniformly random point between the center point and the opposite point. The quasi-opposition  $\check{x}_j^{qo}$  is defined as follows:

$$\check{x}_j^{qo} = \begin{cases} \text{rand}(M_j, \check{x}_j), & \text{if } x_j < M_j \\ \text{rand}(\check{x}_j, M_j), & \text{otherwise} \end{cases}$$

$$M_j = \frac{a_j + b_j}{2}, \tag{7}$$

where  $\text{rand}(x, y)$  is a uniformly number between  $x$  and  $y$ .  $M_j$  is mid-point of  $j$ -dimensional components,  $\check{x}_j$  represents opposite point of  $j$ -dimensional components.

After quasi opposition proposed, Ergezer et al. [18] introduced QROBL. The quasi-reflection opposition  $\check{x}_j^{qr}$  is defined as follows:

$$\check{x}_j^{qr} = \begin{cases} \text{rand}(x_j, M_j), & \text{if } x_j < M_j \\ \text{rand}(M_j, x_j), & \text{otherwise.} \end{cases} \tag{8}$$

Similar to quasi-opposition and quasi-reflection opposition, Seif et al. designed two OBL variants [27]: extended opposition  $\check{x}_j^{eo}$  and reflected extended opposition  $\check{x}_j^{reo}$  are defined by

$$\check{x}_j^{eo} = \begin{cases} \text{rand}(\check{x}_j, b_j), & \text{if } x_j < M_j \\ \text{rand}(a_j, \check{x}_j), & \text{otherwise} \end{cases} \tag{9}$$

$$\check{x}_j^{reo} = \begin{cases} \text{rand}(x_j, b_j), & \text{if } x_j < M_j \\ \text{rand}(a_j, x_j), & \text{otherwise.} \end{cases} \tag{10}$$

In [26], Xu et al. proposed the current optimum opposition (COOBL), which used the information of the current best individual.

$$\check{x}_j^{coo} = 2x_{\text{best},j} - x_j, \tag{11}$$

where  $x_{\text{best},j}$  is the best solution of current population.

Generalized OBL was proposed by Wang et al. [25]. The GOBL is defined by:

$$\check{x}_j^{go} = k \cdot (a_j + b_j) - x_j, \tag{12}$$

where  $k$  is a random number from  $[0,1]$ .

In [28], Rahnamayan et al. proposed centroid opposition (COBL), which employed the information of the entire population (centroid point). The COBL is defined by:

$$\check{x}_j^{co} = 2C_j - x_j, \quad C_j = \frac{\sum_{i=1}^{NP} x_{i,j}}{NP}. \tag{13}$$

### Proposed algorithm

First, in the evolutionary process, the generation of the opposite population depends on the jumping rate. All individuals generate corresponding opposite individuals if the random number is smaller than the jumping rate, which is called the population-based opposite strategy. However, population-based opposition reduces the utilization of the opposite population when the global optimal is approached. Therefore, the subpopulation-based opposite strategy is proposed in this paper. Moreover, there is no fixed parameter setting suitable for all problems or at different stages of evolution of the same problem. The self-adaptive parameter control strategy can solve this problem. For these reasons, the subpopulation-based OBL (SPOBL) with a self-adaptive parameter strategy is proposed. SPODE is the proposed algorithm, which is the combination of DE and SPOBL.

### Motivations

There are two primary motivations in this paper: (1) previous OBL variants are population-based, and there is no research on subpopulation-based. (2) The research on parameter control of OBL is not deep enough, so parameter control of OBL is a crucial problem in the OBL community.

First, OBL and its variants aim to increase population diversity and thus accelerate the convergence of the algorithm. However, there has been a problem that the individual generated by the opposition cannot enter the next generation during the evolutionary process, so the opposition wastes fitness evaluations. Such individual is called insensitive individual, and this individual is commonly found in existing population-based OBL variants. To eliminate the influence of the insensitive individual on the algorithm, this paper proposes a subpopulation-based opposition framework from the individual’s perspective.

Figure 1 shows the definition of the insensitive individual, where  $x$  represents the original individual, and  $ox$  represents

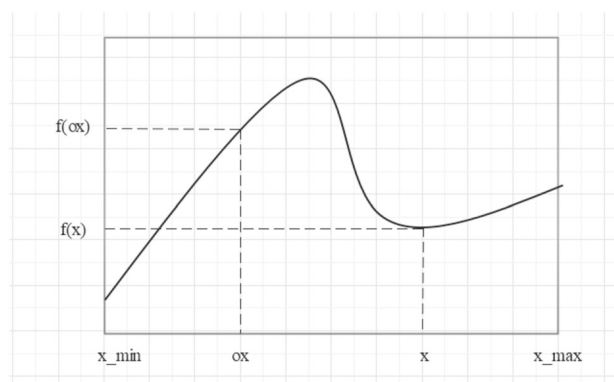


Fig. 1 Definition of the insensitive individual

the opposite individual of  $x$ .  $f(x)$  and  $f(ox)$  represent the corresponding fitness values, respectively. It can be seen that when  $f(x) < f(ox)$ , then  $x$  is the insensitive individual (only consider the problem of minimization). To eliminate the influence of insensitive individuals on the performance of the algorithm, a subpopulation-based opposition is proposed.

The subpopulation-based opposition selects the individuals to perform the opposition by their jumping rates. Hence, insensitive individuals still participate in opposition during the iteration, and the subpopulation-based opposition is also influenced by the individuals. Although population-based and subpopulation-based oppositions are persistently affected by insensitive individuals, the latter is less impacted than the former. It also inspired us to design another strategy to further eliminate this effect.

Besides, OBL is a mechanism that allows the algorithm to explore more search regions, which will increase the exploration capability of the algorithm. To balance between exploration and exploitation, we proposed a self-adaptive parameter control strategy. This strategy is used to select individuals suitable for the opposition and enhances the exploitation ability by controlling the subpopulation size. At the beginning stage of evolution, the subpopulation size is large because exploration is more important than exploitation. However, overemphasizing exploration and ignoring exploitation will make the population converge too slowly or even not converge. As the iteration progresses, the subpopulation size should become smaller to eliminate the influence of insensitive individuals and finally balance between exploration and exploitation.

## Subpopulation opposition-based learning

### Subpopulation strategy

In [34], the population was divided into multiple subpopulations according to fitness values. Then different mutation strategies were assigned to each subpopulation. The subpopulation in this paper is randomly selected according to probability. The opposite subpopulation and the original population are combined. Then select the first  $NP$  fittest individuals to enter the next generation. In the improved algorithm, the jumping rate  $\mathbf{J}$  is a vector and acts on individuals, while the jumping rate  $J$  of ODE is a number and acts on the population. The previous OBL variants were all population based, and the OBL variant in this paper is subpopulation based. The specific operation steps are as follows:

- After the population  $X = \{x_i | i = 1, 2, \dots, NP\}$  is initialized, the jumping rate vector  $\mathbf{J} = \{j_1, j_2, \dots, j_{NP}\}$  is generated using a Gaussian distribution with mean  $\mu_J$ , where  $\mu_J$  represents the mean value of the  $\mathbf{J}$ , and its initial value is set to 0.3 [15].

- After the population undergoes mutation, crossover and selection, the subpopulation  $SX = \{x_j | j = 1, 2, \dots, m < NP\}$  with size  $m$  is generated according to  $\mathbf{J} = \{j_i | i = 1, 2, \dots, NP\}$ . The corresponding individual becomes a member of the subpopulation, if a random number is less than  $j_i$ . The subpopulation size is related to  $\mu_J$ . The relationship between the subpopulation size and  $\mu_J$  is  $mean(m) \approx \mu_J \cdot NP$ ,  $mean(o)$  is the arithmetic mean of  $o$ .
- Equation (13) performs on  $SX$  to generate the opposite subpopulation  $OSX = \{x_j | j = 1, 2, \dots, m\}$ .
- Select the first  $NP$  fittest individuals from the set  $\{X \cup OSX\}$  to enter the next generation and record the surviving opposite individuals' jumping rates as  $S_J$ .

The individuals of the subpopulation are randomly selected according to their jumping rates. Random selection can increase the population diversity and maintain search efficiency. In each iteration, every individual is matched with a jumping rate. When the individual's jumping rate is higher than a random number, this individual as the subpopulation member and participates in the opposition.  $\mu_J$  is the mean value of the entire population's jumping rate. With the increase of  $\mu_J$ , the subpopulation size also increases. It can be seen from the above that  $\mu_J$  is an important parameter that affects the subpopulation size. Thus, the generalized Lehmer mean controls the value of  $\mu_J$  in the self-adaptive strategy. In summary, the generalized Lehmer mean can control the subpopulation size during the evolutionary process. More details about self-adaptive parameter control strategy and generalized Lehmer mean are observed in the next subsection.

### Self-adaptive parameter control strategy

As the iteration progresses, the individual jumping rate  $j_i$  is generated by Gaussian random number with the mean value of  $\mu_J$  and variance of 0.1 [35]:

$$j_i = randn_i(\mu_J, 0.1). \quad (14)$$

$S_J$  denotes the set of jumping rates that successfully enters the next generation of individuals from  $OSX$ . The initial value of  $\mu_J$  is 0.3 and is updated with the following formula as the iteration progresses [15,22].

$$\mu_J = (1 - c) \cdot \mu_J + c \cdot \text{Lehmer}(S_J, p), \quad (15)$$

where  $\text{Lehmer}(S_J, p)$  represents the generalized Lehmer mean of  $S_J$  with  $p$  as a parameter, the pseudo-code of SPOBL is provided in Algorithm 1. The corresponding explanation of the pseudo-code is shown in Fig. 2.

The core idea of Eq. (15) is: Individuals with good parameters are often easier to survive. It uses  $S_J$  to record the

**Algorithm 1** SPOBL

**Step 1** Select and calculate the opposition subpopulation

$$Jind = find(rand(size(J)) \leq J);$$

$$SP = P(Jind, :);$$

$$CSP = sum(SP)/length(SP);$$

$$OSP = 2CSP - SP;$$

**Step 2** Select first NP individuals

select first NP fittest individuals from  $\{P, OSP\}$  record survive individuals as  $P^{G+1}$ , record jumping rate  $J$  of surviving individuals from  $OSP$  as  $S_J$ .

**Step 3** Self-adaptive parameter control

$$\mu_J = (1 - c) \cdot \mu_J + c \cdot \text{Lehmer}(S_J, p);$$

jumping rates that successfully enter the next generation of individuals. Then,  $S_J$  is employed to generate parameters more suitable for the evolutionary process. In [35],  $c = 0.05$  usually performs best.

**Generalized Lehmer mean**

We can conclude from the previous analysis that the subpopulation size is related to  $\mu_J$ , which determines the subpopulation diversity. The diversity will decrease [36] if the convergence speed is accelerated. To better balance between population diversity and convergence speed, the arithmetic mean and Lehmer mean were used to generate the crossover rate and scaling factor [35]. Compared to arithmetic mean, Lehmer mean provides a larger mean from the same group of data. To further extend the range of values of  $\mu_J$ , we introduced the generalized Lehmer mean with

the parameter  $p$  for controlling the value of  $\mu_J$  during the evolutionary process. Thus, the subpopulation size is dynamically controlled according to parameter  $p$  of the generalized Lehmer mean during the evolutionary process. The calculation formula of the generalized Lehmer mean is shown below:

$$L(x_1, x_2, \dots, x_n, p) = \frac{x_1^p + x_2^p + \dots + x_n^p}{x_1^{p-1} + x_2^{p-1} + \dots + x_n^{p-1}}. \tag{16}$$

Let  $n = 2$ ,  $x_1$  and  $x_2$  are two positive numbers. In this case, partial derivative of generalized Lehmer mean of two positive numbers with respect to  $p$  is:

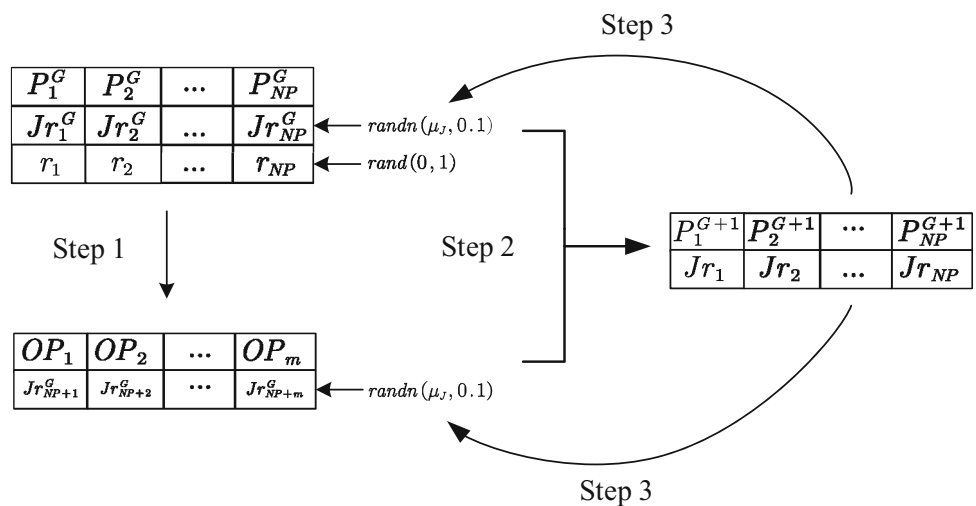
$$\frac{\partial L(x_1, x_2, p)}{\partial p} = \frac{x_1^{p-1} x_2^{p-1} (x_1 - x_2) (\ln x_1 - \ln x_2)}{(x_1^{p-1} + x_2^{p-1})^2} \geq 0, \tag{17}$$

where Eq. (16) is a monotonically increasing function of  $p$ . With the increase of  $p$ , the value of generalized Lehmer mean obtained from the same set of data gradually increases. According to the previous analysis, the subpopulation size is controlled by controlling the value of  $p$ .

Let  $p$  is a real number, the value of  $L$  is an interesting problem when  $p$  tends to infinity. So Eq. (16) can also be expressed in the following form:

$$L(x_1, \dots, x_n, p) = \frac{x_1^p \left[ 1 + \left(\frac{x_2}{x_1}\right)^p + \dots + \left(\frac{x_n}{x_1}\right)^p \right]}{x_1^{p-1} \left[ 1 + \left(\frac{x_2}{x_1}\right)^{p-1} + \dots + \left(\frac{x_n}{x_1}\right)^{p-1} \right]}. \tag{18}$$

**Fig. 2** Pseudo-code interpretation of SPOBL



According to Eq. (18), the following conclusions can be drawn:

$$\lim_{p \rightarrow -\infty} L_p(x_1, \dots, x_n) = \min\{x_1, \dots, x_n\} \tag{19}$$

$$\lim_{p \rightarrow +\infty} L_p(x_1, \dots, x_n) = \max\{x_1, \dots, x_n\}. \tag{20}$$

The value of  $\mu_J$  is a crucial problem in the evolutionary process, which affects the algorithm’s performance. In [15,22], the recommended value of jumping rate was [0.1,0.4]. On the one hand, if the value of  $p$  is too large, it will result in the individual jumping rate with a value of 0.6, thus the fitness evaluations will be wasted. On the other hand, if the value of  $p$  is too small, that will cause the value of the individual jumping rate to be 0.1, and the effect of opposition is not significant. Thus, we designed a numerical test and considered four schemes for adapting  $\mu_J$  to gain the best performance. These four schemes are shown below.

1. *Positive*: The value of  $p$  is 2, this scheme results in  $\mu_J$  rapidly grow.
2. *Neutral*: The value of  $p$  is 1, this scheme causes  $\mu_J$  to remain almost unchanged.
3. *Negative*: The value of  $p$  is 0.5, this scheme results in  $\mu_J$  rapidly decrease.
4. *Time varying*: The  $\mu_J$  is linear decreases from 0.6 to 0.1.

The specific results are shown in Fig. 3. The horizontal axis indicates the number of iterations and the vertical axis represents the value of  $\mu_J$ . We can see that when  $p$  equals 1, the value of  $\mu_J$  remains around 0.3 and the trend of  $\mu_J$  is parallel to the horizontal axis. However, when  $p$  is 0.5, the value of  $\mu_J$  rapidly decreases in the first 200 iterations, and after 200 iterations it also plateaus and the final value of  $\mu_J$  remains at 0.1. When the value of  $p$  is 1.5 and 2, the value of  $\mu_J$  also rapidly increases in the first 200 iterations. However, the speed of increment is different, and the larger the value of  $p$ , the faster  $\mu_J$  increases. When  $p$  equals 2, the final value of  $\mu_J$  falls around 0.6. Thus, the range of  $p$  is determined to be between 0.5 and 2 according to the results. The effect of the four schemes on the performance of SPODE has been further investigated in Sect. 4.2.4.

The generalized Lehmer mean can also be expressed as several common mean values. Without loss of generality, in this case, the generalized Lehmer mean is

$$L(x_1, x_2, p) = \begin{cases} \frac{2x_1x_2}{x_1 + x_2} & p = 0 \\ \sqrt{x_1x_2} & p = 0.5 \\ \frac{x_1 + x_2}{2} & p = 1 \\ \frac{x_1^2 + x_2^2}{x_1 + x_2} & p = 2 \end{cases} \tag{21}$$

from Eq. (21), it can be seen that generalized Lehmer mean is the harmonic mean, geometric mean, arithmetic mean and contraharmonic mean when  $p = 0$ ,  $p = 0.5$ ,  $p = 1$  and  $p = 2$ , respectively.

### Subpopulation opposition DE

The SPOBL is proposed by combining the subpopulation-based strategy and the self-adaptive parameter control mechanism with the generalized Lehmer mean. SPODE is a combination of DE and SPOBL. Unlike other ODE variants, SPODE applies the jumping rate to the individual. SPODE executes SPOBL in every iteration according to a predefined jumping rate. If a random number generated according to the uniform distribution is lower than or equal to the jumping rate, the individual corresponding to this jumping rate will be selected as a subpopulation and performs opposition. The entire pseudo-code of SPODE is presented in Algorithm 2.

### Computational complexity of SPODE

Compared with classical DE, ODE variants require additional computations on the opposition. In each generation, all individuals in the subpopulation take into account the opposition. The computational complexity of SPOBL is  $O(D \times NP)$ . The complexity of classical DE is  $O(G_{\max} \times D \times NP)$ , and the total computational complexity of DE with SPOBL is  $O(G_{\max} \times [D \times NP + D \times NP])$  which can be simplified to  $O(G_{\max} \times D \times NP)$ . However, the main computational time of evolutionary algorithms is used to evaluate the fitness function. Moreover, the additional calculation cost due to diversity and opposition is negligible compared with fitness evaluations.

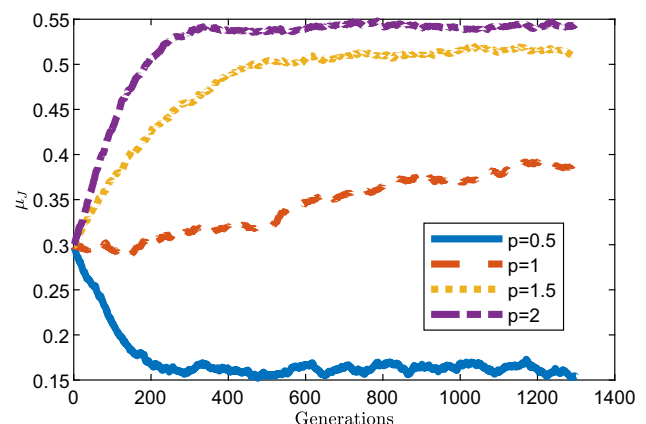


Fig. 3 Example of Lehmer mean of different p

**Algorithm 2** SPODE**Step 1 Initialization**

set up MaxFEs,  $G$ ,  $\mu_J$ . generation the population  $P0$  of NP individuals,  $P0 = \{\mathbf{x}_1^G, \mathbf{x}_2^G, \dots, \mathbf{x}_{NP}^G\}$ , with  $\mathbf{x}_i^G = \{x_{i,1}^G, x_{i,2}^G, \dots, x_{i,D}^G\}$ , each individual is in range  $[a, b]$ .

**for**  $i = 1 : NP$  **do**

$OP_{i,j} = a + b - P0_{i,j}$ ;

**end for**

select first NP fittest individuals from  $\{P0, OP\}$  as  $P$ .

**Step 2 Evolution Process**

**while** The termination criterion is satisfied **do**

**Step 2.1 Mutation**

**for**  $i = 1 : NP$  **do**

$J = \mathcal{N}(\mu_J, 0.1)$ , Randomly choose  $i \neq r_1 \neq r_2 \neq r_3$  from  $[1, NP]$ .

$\mathbf{V}_i^G = \mathbf{x}_{r_1} + F \cdot (\mathbf{x}_{r_2} - \mathbf{x}_{r_3})$ ;

**Step 2.2 Crossover**

Generate  $j_{\text{rand}} = \text{randint}(1, D)$ ;

**for**  $j = 1 : D$  **do**

**if**  $j = j_{\text{rand}} \vee \text{rand}(0, 1) < CR$  **then**

$U_{i,j}^G = V_{i,j}^G$ ;

**else**

$U_{i,j}^G = P_{i,j}^G$ ;

**end if**

**end for**

**Step 2.3 Selection**

**if**  $f(\mathbf{U}_i^G) < f(\mathbf{P}_i^G)$  **then**

$\mathbf{P}'_i = \mathbf{U}_i^G$ ;

**else**

$\mathbf{P}'_i = \mathbf{P}_i^G$ ;

**end if**

**end for**

$P = P'$ ;

**Step 3 SPOBL**

$J_{\text{ind}} = \text{find}(\text{rand}(\text{size}(J)) \leq J)$ ;

$SP = P(J_{\text{ind}}, :)$ ;

$CSP = \text{sum}(SP) / \text{length}(SP)$ ;

$OSP = 2CSP - SP$ ;

select first NP fittest individuals from  $\{P, OSP\}$  record survive individuals as  $P^{G+1}$ , record jumping rate  $J$  of surviving individuals from  $OSP$  as  $S_J$ .

$\mu_J = (1 - c) \cdot \mu_J + c \cdot \text{Lehmer}(S_J, p)$ ;

$G = G + 1$ ;

**end while**

**Experimental verification****Experimental setup**

All the experiments were conducted on Windows 10 Pro-64 bit of a PC with Inter(R) Core(TM) i7-9700 CPU @ 3.0GHz. All the algorithms were implemented in the MATLAB 9.7 (R2019b) programming language.

**Benchmark functions**

We employed a set of 29 benchmark functions for explaining the performance of the proposed algorithm. The current well-known single objective optimization test suites contain CEC 2005, 2013, 2014, and 2017. We chose the CEC 2017 as the test suite for this experiment. In the CEC 2017, there are two unimodal functions (F1, F3), seven simple multimodal functions (F4–F10), ten expanded hybrid functions (F11–20), ten composition functions (F21–F30). Please refer to [33] for more details about CEC 2017.

**Parameter setting**

We compared the performance of SPOBL with eight state-of-the-art OBL variants, namely: (1) OBL [15], (2) QOBL [24], (3) QROBL [18], (4) GOBL [25], (5) COOBL [26], (6) EOBL [27], (7) REOBL [27] and (8) COBL [28]. To have a fair comparison, all algorithms in this section use DE/rand/1/bin. The values of  $F$  and  $CR$  are 0.5 and 0.9 [9]. The jumping rate of ODE, GODE, COODE and CODE is 0.3 [22,37]. The jumping rate of QODE, QRODE, EODE and REODE is 0.05 [22,37]. The initial value of  $\mu_J$  is 0.3. The dimensionality of the problems  $D$  is 30 and 50, and the corresponding population size NP is 150 and 250 [9,10]. The max fitness evaluations is  $10^4 \times D$ . Every algorithm independently runs 30 times on each benchmark function [33].

**Comparison metrics**

The fitness error value (FEV) evaluates the performance of the test algorithm, which can be defined as follows.

$$\text{FEV} = F(\hat{x}) - F(x^*), \quad (22)$$

where  $F(x)$  represents fitness function,  $\hat{x}$  is the best solution of test algorithm,  $x^*$  is the global optimal of  $F(x)$ .

**Statistical test**

To compare whether there is a significant difference in performance between the two algorithms, we used the pairwise comparison, which is the Wilcoxon rank-sum test with



$\alpha = 0.05$  significance level. The null hypothesis is that the proposed algorithm and the corresponding algorithm are independent. When the null hypothesis is rejected, we used three symbols to denote the relationship between SPODE and the corresponding algorithm.

1. +: The performance of the proposed algorithm is significantly better than that of the corresponding algorithm.
2. =: The performance of the proposed algorithm is not statistically significant than that of the corresponding algorithm.
3. -: The performance of the proposed algorithm is significantly worse than that of the corresponding algorithm.

In pairwise comparison, the family-wise error rate (FWER) loses control, such as the Wilcoxon test, should not be used to conduct various comparisons involving a set of algorithms [38]. Therefore, to determine whether the difference between multiple algorithms. We employ the Friedman test with Tukey–Kramer post hoc [39].

## Experiment results

In this subsection, we designed six experiments to verify the performance of the proposed algorithm. In Sect. 4.2.1, we evaluated OBL variants and SPOBL with DE on the CEC 2017 at 30-D and 50-D. In Sect. 4.2.2, we analyzed the run-time complexity of OBL variants embedding on DE. In Sect. 4.2.3, we evaluated SPODE with different strategies to verify different algorithms' influence. In Sect. 4.2.4, the comparison is performed between different  $p$  and time-varying. Finally, to illustrate the effectiveness of SPOBL on advanced DE variants, we calculated the efficiency of SPOBL embedding on LSHADE and jSO in Sects. 4.2.5 and 4.2.6. The best results are highlighted in boldface.

### Comparison for DE with OBL variants

In this part, the performance results on CEC 2017 at 30-D and 50-D are presented. Table 1 provides the results of mean and standard deviation and the Wilcoxon test. From the table, it can be seen that SPODE performs better than most ODE variants. For the unimodal functions (F1, F3), COODE and SPODE find the optimal on F1 and F3, respectively. Both SPODE and CODE converge to the local optimum on F1. This reveals that the exploitation ability of SPODE should be further studied. The performance of COODE on F1 and F3 rank first and second, which indicates that COODE has a high exploitation ability on the unimodal functions. For the multimodal functions (F4–F10), CODE outperforms the other algorithms on F5, F7, F8 and F10. SPODE only achieves the best performance on F6, ODE on F9 and COODE on F4. However, SPODE ranks second on F5, F7 and F8. It

indicates that CODE has sufficient exploration ability on multimodal functions. SPODE is inspired by CODE, so SPODE is still very competitive in terms of multimodal functions. For hybrid functions (F11–F20), SPODE converges to global optimal on F11, F13, F14, F15 and F20. QRODE on F12 and F18. CODE on F16, F17 and F19. It demonstrates that SPODE has significant enhancement to performance due to the self-adaptive parameter control mechanism. For composition functions (F21–F30), SPODE has achieved superior results on F22, F23, F24 and F26. QODE has obtained the best performance on the F27, F28 and F30. QRODE on F25. CODE on F21 and F29. The composition functions have many local optima and many complex characteristics. Hence, the results denote that the proposed algorithm has a better balance between exploration and exploitation. The proposed mechanism facilitates better performance.

The Wilcoxon test results show that SPODE significantly finds more accurate solutions than ODE, QODE, QRODE, GODE, COODE, EODE and REODE on more than half of the functions. Compared with CODE, SPODE has significantly improved performance on 12 functions, but decreased performance on 4 functions. In other words, SPODE performs much better than other ODE variants. As shown in Table 2, the Friedman test results with Tukey–Kramer's post hoc are provided. It can be seen from Table 2 that SPODE ranks first among all algorithms, and the outperformance over GODE and COODE is statistically significant. The top three algorithms are SPODE, ODE, and CODE. All in all, SPODE is better than all comparison algorithms in performance on CEC 2017 at 30-D.

Moreover, we analyzed the results of Table 1 based on the function's properties. In unimodal functions (F1, F3), the performance gap of each algorithm is not significant. CODE dominates the best performance on hybrid functions. The performance of SPODE is better than other algorithms on hybrid functions (F11–F20) and composition functions (F21–F30). Table 1 shows that the proposed algorithm better balance between exploration and exploitation, and it can find better solutions than other algorithms when dealing with complex problems.

Furthermore, to clearly observe the convergence curves. We divided the comparison algorithms into two groups. The first group is based on the original OBL and contains QOBL, QROBL, EOBL and REOBL. The calculation of these OBL variants requires less information (only information about the current individual and the midpoint of the search space is needed). The second group includes OBL, COBL, COOBL and GOBL. In these OBL variants, more information is needed, such as the current best individual and centroid individual. Therefore, the second group is more complicated than that of the first group. The first and second groups are shown in Figs. 4 and 5, respectively.

Table 1 Results of ODE variants on CEC 2017 at 30-D

	SFODE	ODE	QODE	QRQDE	GOODE	COODE	EODE	REODE	CODE
F1	Mean	3.00E+03	3.32E-01	8.48E-04	2.34E-01	<b>1.64E-10</b>	7.94E-04	9.03E-04	2.38E+03
	Std	4.20E+03	1.81E+00 (-)	4.77E-04 (-)	1.40E-01 (-)	<b>4.38E-10</b> (-)	3.75E-04 (-)	4.95E-04 (-)	2.69E+03 (=)
F3	Mean	<b>1.34E+03</b>	6.02E+04	2.96E+03	1.07E+04	1.44E+04	4.67E+04	2.71E+04	3.12E+03
	Std	8.33E+02	1.93E+04 (+)	1.48E+03 (+)	5.37E+03 (+)	2.12E+04 (=)	1.74E+04 (+)	1.05E+04 (+)	1.26E+03 (+)
F4	Mean	8.74E+01	5.83E+01	5.86E+01	5.87E+01	<b>5.36E+01</b>	5.86E+01	5.87E+01	8.71E+01
	Std	2.12E+01	7.90E+00 (-)	<b>4.69E-04</b> (-)	1.03E-01 (-)	1.82E+01 (-)	9.71E-04 (-)	1.01E+00 (-)	1.15E+01 (=)
F5	Mean	2.74E+01	9.95E+01	1.83E+02	1.89E+02	1.16E+02	1.82E+02	1.85E+02	<b>2.55E+01</b>
	Std	2.69E+01	4.75E+01 (+)	1.27E+01 (+)	1.22E+01 (+)	5.24E+01 (+)	1.35E+01 (+)	<b>9.66E+00</b> (+)	1.55E+01 (=)
F6	Mean	<b>1.56E-07</b>	4.08E-04	5.31E-04	4.78E-04	8.23E-04	5.42E-04	5.53E-04	4.46E-03
	Std	<b>8.53E-07</b>	1.91E-04 (+)	1.41E-04 (+)	1.36E-04 (+)	2.37E-03 (+)	1.29E-04 (+)	1.95E-04 (+)	1.49E-02 (+)
F7	Mean	1.10E+02	1.43E+02	2.19E+02	2.23E+02	1.49E+02	2.19E+02	2.19E+02	<b>4.83E+01</b>
	Std	6.01E+01	3.73E+01 (+)	1.00E+01 (+)	8.91E+00 (+)	<b>7.52E+00</b> (+)	9.98E+00 (+)	1.16E+01 (+)	1.42E+01 (-)
F8	Mean	2.11E+01	8.48E+01	1.85E+02	1.83E+02	1.06E+02	1.82E+02	1.86E+02	<b>2.04E+01</b>
	Std	2.27E+01	4.89E+01 (+)	1.00E+01 (+)	1.08E+01 (+)	5.55E+01 (+)	1.27E+01 (+)	9.78E+00 (+)	<b>7.56E+00</b> (=)
F9	Mean	1.88E-01	<b>1.14E-14</b>	7.11E-11	3.86E-11	3.84E+00	9.27E-11	7.51E-11	6.97E-01
	Std	5.29E-01	<b>3.47E-14</b> (-)	9.77E-11 (-)	3.64E-11 (-)	9.61E+00 (+)	6.09E-11 (-)	6.50E-11 (-)	1.72E+00 (=)
F10	Mean	6.45E+03	4.28E+03	6.77E+03	6.76E+03	6.04E+03	6.76E+03	7.03E+03	<b>2.80E+03</b>
	Std	4.17E+02	1.08E+03 (-)	4.01E+02 (+)	3.23E+02 (+)	9.49E+02 (-)	5.12E+02 (+)	<b>2.32E+02</b> (+)	1.39E+03 (-)
F11	Mean	<b>2.59E+01</b>	2.95E+01	6.87E+01	6.61E+01	5.54E+01	6.76E+01	7.25E+01	3.32E+01
	Std	2.55E+01	2.71E+01 (=)	1.74E+01 (+)	<b>1.55E+01</b> (+)	4.89E+01 (+)	2.06E+01 (+)	2.19E+01 (+)	2.79E+01 (=)
F12	Mean	6.78E+04	1.76E+04	5.20E+03	<b>4.31E+03</b>	1.85E+04	4.42E+03	5.38E+03	1.16E+05
	Std	7.59E+04	7.23E+03 (-)	2.20E+03 (-)	<b>1.49E+03</b> (-)	1.65E+04 (-)	2.12E+03 (-)	3.13E+03 (-)	1.18E+05 (=)
F13	Mean	<b>4.47E+01</b>	8.68E+01	8.95E+01	8.79E+01	3.39E+03	8.88E+01	8.70E+01	3.30E+02
	Std	2.91E+01	1.40E+01 (+)	8.04E+00 (+)	7.25E+00 (+)	<b>6.10E+00</b> (+)	7.68E+00 (+)	6.77E+00 (+)	6.35E+02 (+)
F14	Mean	<b>2.01E+01</b>	6.90E+01	6.50E+01	6.69E+01	1.01E+02	6.74E+01	6.80E+01	2.64E+01
	Std	1.27E+01	6.20E+00 (+)	5.05E+00 (+)	<b>4.41E+00</b> (+)	4.70E+01 (+)	6.60E+00 (+)	4.69E+00 (+)	9.01E+00 (+)
F15	Mean	<b>1.14E+01</b>	1.97E+01	4.66E+01	4.59E+01	3.24E+02	4.72E+01	4.74E+01	2.23E+01
	Std	9.08E+00	1.49E+01 (+)	<b>3.98E+00</b> (+)	5.60E+00 (+)	8.68E+02 (+)	4.89E+00 (+)	4.13E+00 (+)	1.38E+01 (+)
F16	Mean	4.77E+02	6.39E+02	1.14E+03	1.18E+03	1.54E+03	1.13E+03	1.09E+03	<b>3.12E+02</b>
	Std	3.82E+02	4.20E+02 (=)	2.00E+02 (+)	1.28E+02 (+)	4.05E+02 (+)	1.28E+02 (+)	1.78E+02 (+)	1.73E+02 (=)
F17	Mean	1.12E+02	1.05E+02	2.79E+02	2.57E+02	5.73E+02	2.58E+02	2.71E+02	<b>5.69E+01</b>
	Std	7.61E+01	8.64E+01 (=)	1.17E+02 (+)	1.13E+02 (+)	2.79E+02 (+)	1.06E+02 (+)	1.11E+02 (+)	<b>4.75E+01</b> (-)
F18	Mean	3.41E+03	4.95E+01	4.55E+01	<b>4.35E+01</b>	1.43E+04	4.47E+01	4.39E+01	7.40E+03
	Std	4.94E+03	4.41E+00 (-)	3.25E+00 (-)	<b>2.70E+00</b> (-)	4.26E+04 (=)	3.59E+00 (-)	3.80E+00 (-)	6.26E+03 (+)
F19	Mean	1.01E+01	1.77E+01	2.95E+01	3.05E+01	3.59E+01	3.01E+01	2.99E+01	<b>8.95E+00</b>
	Std	6.31E+00	5.48E+00 (+)	2.55E+00 (+)	<b>2.43E+00</b> (+)	2.75E+01 (+)	2.77E+00 (+)	2.94E+00 (+)	4.01E+00 (=)

Table 1 continued

	SPODE	ODE	QODE	QRODE	GODE	COODE	EODE	REODE	CODE
F20	Mean	3.53E+01	1.34E+02	6.48E+01	2.29E+02	7.54E+02	1.18E+02	1.46E+02	3.90E+01
	Std	2.46E+01	1.09E+02 (=)	3.00E+01 (+)	1.94E+02 (+)	2.37E+02 (+)	1.07E+02 (+)	1.44E+02 (+)	4.86E+01 (=)
F21	Mean	2.25E+02	3.62E+02	3.76E+02	3.78E+02	3.09E+02	3.74E+02	3.73E+02	<b>2.22E+02</b>
	Std	2.81E+01	4.12E+01 (+)	1.14E+01 (+)	1.08E+01 (+)	5.24E+01 (+)	<b>9.91E+00</b> (+)	1.08E+01 (+)	1.28E+01 (=)
F22	Mean	<b>1.00E+02</b>	1.00E+02	1.00E+02	1.00E+02	3.15E+03	5.39E+02	1.00E+02	2.50E+02
	Std	<b>1.72E-13</b>	7.71E-13 (+)	1.99E-07 (+)	1.54E-05 (+)	3.14E+03 (+)	1.67E+03 (+)	9.54E-07 (+)	5.77E+02 (=)
F23	Mean	<b>3.62E+02</b>	4.67E+02	5.29E+02	5.30E+02	4.69E+02	5.28E+02	5.30E+02	3.75E+02
	Std	<b>9.12E+00</b>	6.13E+01 (+)	1.24E+01 (+)	1.32E+01 (+)	6.49E+01 (+)	9.57E+00 (+)	1.10E+01 (+)	2.05E+01 (+)
F24	Mean	<b>4.39E+02</b>	5.08E+02	5.94E+02	5.93E+02	5.47E+02	5.92E+02	5.91E+02	4.44E+02
	Std	<b>6.35E+00</b>	7.43E+01 (+)	1.01E+01 (+)	9.62E+00 (+)	6.18E+01 (+)	1.19E+01 (+)	1.16E+01 (+)	1.04E+01 (+)
F25	Mean	3.87E+02	3.87E+02	<b>3.87E+02</b>	3.87E+02	3.87E+02	3.87E+02	3.87E+02	3.87E+02
	Std	2.09E-01	2.19E-02 (-)	1.81E-02 (-)	3.16E-02 (-)	9.66E-02 (=)	<b>1.74E-02</b> (-)	2.06E-02 (-)	4.09E-01 (+)
F26	Mean	<b>1.07E+03</b>	1.51E+03	2.75E+03	2.78E+03	2.03E+03	2.70E+03	2.73E+03	1.23E+03
	Std	1.16E+02	5.03E+02 (+)	1.19E+02 (+)	1.20E+02 (+)	5.77E+02 (+)	1.54E+02 (+)	<b>9.72E+01</b> (+)	1.33E+02 (+)
F27	Mean	4.97E+02	4.94E+02	4.93E+02	4.90E+02	5.33E+02	4.95E+02	4.96E+02	5.02E+02
	Std	7.06E+00	7.02E+00 (=)	9.34E+00 (-)	1.12E+01 (-)	4.15E+01 (+)	7.89E+00 (=)	<b>6.97E+00</b> (=)	8.76E+00 (+)
F28	Mean	4.01E+02	3.22E+02	3.07E+02	3.14E+02	3.73E+02	3.04E+02	3.07E+02	4.14E+02
	Std	3.54E+01	4.06E+01 (-)	2.62E+01 (-)	3.24E+01 (-)	8.76E+01 (=)	1.91E+01 (-)	2.62E+01 (-)	2.86E+01 (+)
F29	Mean	5.27E+02	1.01E+03	9.05E+02	9.69E+02	1.02E+03	9.00E+02	8.74E+02	<b>4.53E+02</b>
	Std	1.03E+02	1.01E+02 (+)	8.61E+01 (+)	1.08E+02 (+)	3.68E+02 (+)	1.11E+02 (+)	9.15E+01 (+)	<b>4.32E+01</b> (-)
F30	Mean	2.68E+03	2.39E+03	2.09E+03	2.24E+03	3.14E+03	2.10E+03	2.10E+03	3.27E+03
	Std	5.22E+02	1.53E+02 (-)	7.98E+01 (-)	1.05E+02 (-)	3.94E+03 (+)	<b>4.59E+01</b> (-)	5.50E+01 (-)	1.05E+03 (+)
+/-/-		15/5/9	20/0/9	20/0/9	19/0/10	20/5/4	20/1/8	20/1/8	13/12/4

Bold values represent the best results in each row

**Table 2** Friedman test with Tukey–Kramer’s post hoc for ODE variants on CEC 2017 at 30-D

	Algorithm	Average ranking	$z$ value	$p$ value	Adj. $p$ value (Tukey–Kramer)	Sig.	Test statistics	
1	SPODE	<b>3.60</b>						
2	ODE	4.03	−8.67E−01	2.74E−01	9.99E−01	No	$N$	29
3	QODE	4.60	−2.00E+00	5.40E−02	8.81E−01	No	Chi-Square	36.88
4	QRODE	4.53	−1.87E+00	6.99E−02	9.17E−01	No	df	8
5	GODE	6.70	−6.20E+00	1.79E−09	2.66E−04	Yes	$p$ value	1.20E−05
6	COODE	6.43	−5.67E+00	4.25E−08	1.44E−03	Yes	Sig.	Yes
7	EODE	5.03	−2.87E+00	6.55E−03	4.96E−01	No		
8	REODE	5.60	−4.00E+00	1.34E−04	9.22E−02	No		
9	CODE	4.47	−1.73E+00	8.88E−02	9.45E−01	No		

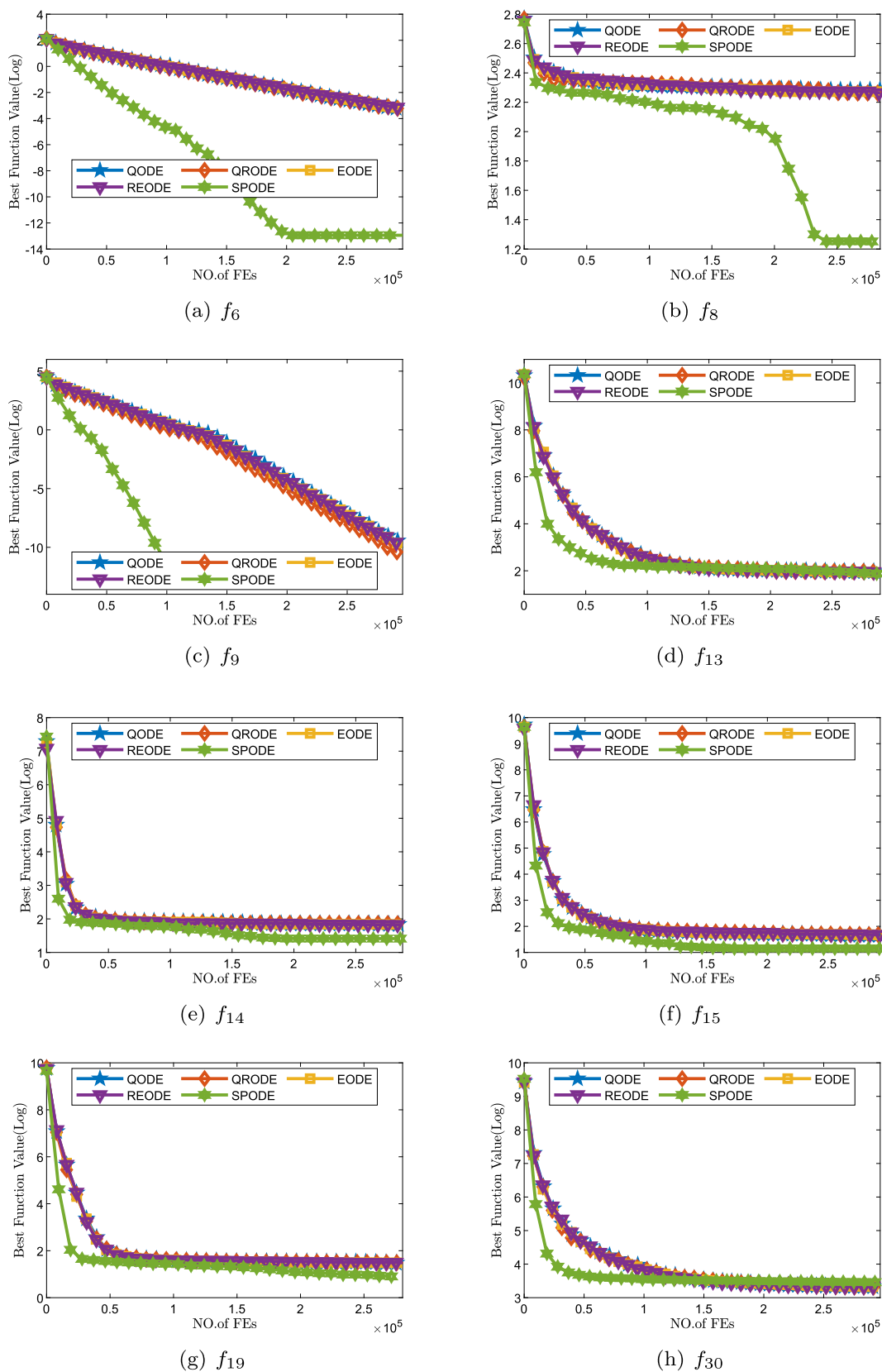
In the first group, we can observe that SPODE achieves better solutions on F6, F8, F9, F13, F14, F15 and F19. SPODE also has the fastest convergence speed compared to ODE variants in the first group. The main reason is that the subpopulation strategy can select the proper individuals to participate in the opposition, and the first group ODE variants have not fully utilized the information in the search space, so SPODE demonstrates superior performance when compared with them. In the second group, a different situation is shown in Fig. 5. SPODE has the best solution on F6, F13, F14 and F15, and the final converged optimal of SPODE is not very different from other algorithms on these functions. However, SPODE still provides the fastest convergence speed. It reveals that the performance of the second group of ODE variants is better than that of the first group. Thus, SPODE has offered superior results than that of the eight ODE variants on CEC 2017 at 30-D.

The performance results on CEC 2017 at 50-D are presented to compare the difference between 30-D and 50-D. The trends in Table 3 are similar to those in Table 1. For unimodal functions (F1, F3), SPODE ranks second and first on F1 and F3, respectively. COODE still ranks first on F1, but SPODE ranks second on F1. This means that SPODE remains a great performance with the dimensional increase. For multimodal functions (F4–F10), SPODE obtains the best results on F5 and F6. ODE on F9. COODE on F4. CODE on F7, F8 and F10. SPODE ranks second place on F7 and F8. There is only one function on which SPODE achieves the optimal on 30-D, but there are two functions on 50-D. It means that the exploration ability of SPODE is enhanced. For hybrid functions (F11–F20), it can be noted that SPODE has gotten the best performance on F11, F14, F16 and F19. ODE on F13 and F15. QRODE on F18. COODE on F12. CODE on F17 and F20. In these functions, since hybrid functions have characteristics of multimodal functions as well as unimodal functions, no single algorithm can have the best performance on all hybrid functions, but SPODE shows the performance on the functions compared to other algorithms. For composition functions (F21–F30), SPODE outperforms the other

algorithms on F21, F23, F24, F26 and F30. QRODE on F22. REODE on F25, F27 and F28. CODE on F29. The obtained results show that the proposed algorithm is competitive and stable due to the equilibrium state between exploration and exploitation.

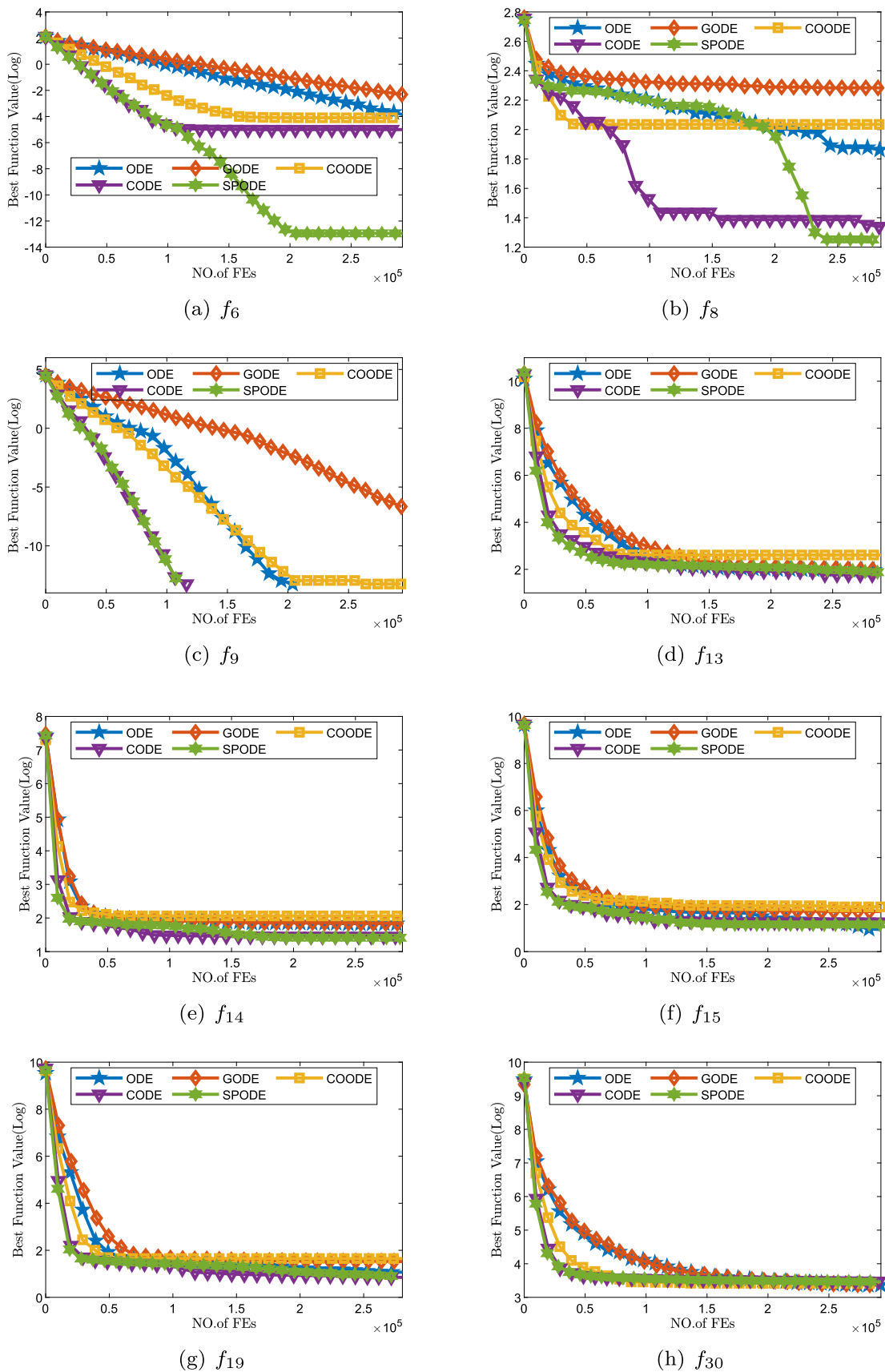
The Wilcoxon test results show that SPODE significantly finds more accurate solutions than ODE, QODE, QRODE, GODE, COODE, EODE, REODE, and CODE for more than half of the functions. Compared with CODE, SPODE has significantly improved performance on 16 functions, but decreased performance on 4 functions. Compared with ODE, SPODE has significantly improved performance on 18 functions, but decreased performance on 7 functions. In other words, SPODE performs much better than other ODE variants. As shown in Table 4, the Friedman test results with Tukey–Kramer’s post hoc are provided. It can be seen from Table 4 that SPODE ranks first among all algorithms, and the outperformance over QODE, GODE, EODE and REODE is statistically significant. The top three algorithms are SPODE, CODE and QRODE. In summary, SPODE outperforms all comparative algorithms on the CEC 2017 at 50-D. Moreover, we analyzed the results of Table 3 based on the function’s properties. Compared with the results at 30-D, SPODE finds more accurate solutions at 50-D. SPODE significantly exhibits better performance in approximately half of the functions. In summary, the proposed algorithm shows a potent competitive edge on the CEC 2017 test suite at 30-D and 50-D.

To further study the convergence behavior of ODE variants, Figs. 6 and 7 show convergence graphs for eight CEC 2017 functions at 50-D. In Fig. 6, we can observe that SPODE achieves better solutions on these six functions. Similar convergence behavior can also be obtained in Figs. 4 and 6. SPODE demonstrates superior performance on both 30-D and 50-D. A different situation is shown in Fig. 7. SPODE obtains the best solution on F6, F8, F14, F16, F19 and F30. Although SPODE is trapped in a local optimum on F9. SPODE still provides the fastest convergence speed. Thus,



**Fig. 4** Convergence curves of SPODE, QODE, QRODE, EODE and REODE on eight 30-D functions: Shifted and Rotated Lunacek Bi\_Rastrigin Function, Shifted and Rotated Levy Function, Shifted and

Rotated Schwefel’s Function, Hybrid Function 4 ( $N = 4$ ), Hybrid Function 5 ( $N = 4$ ), Hybrid Function 6 ( $N = 4$ ), Hybrid Function 6 ( $N = 4$ ) and Composition Function 10 ( $N = 3$ ), respectively



**Fig. 5** Convergence curves of SPODE, ODE, COODE, CODE and GODE on eight 30-D functions: Shifted and Rotated Lunacek Bi\_Rastrigin Function, Shifted and Rotated Levy Function, Shifted and

Rotated Schwefel’s Function, Hybrid Function 4 ( $N = 4$ ), Hybrid Function 5 ( $N = 4$ ), Hybrid Function 6 ( $N = 4$ ), Hybrid Function 6 ( $N = 4$ ) and Composition Function 10 ( $N = 3$ ), respectively

**Table 3** Results of ODE variants on CEC 2017 at 50-D

	SPODE	ODE	QODE	QRODE	CODE	EODE	REODE	CODE	
F1	Mean	2.14E+03	3.06E+03	2.09E+04	2.78E+05	<b>1.18E+03</b>	3.34E+04	5.95E+04	1.16E+05
	Std	1.86E+03	3.75E+03 (=)	2.41E+04 (+)	1.58E+05 (+)	<b>1.71E+03</b> (-)	2.63E+04 (+)	7.87E+04 (+)	5.72E+05 (=)
F3	Mean	<b>3.89E+04</b>	3.06E+05	1.18E+05	2.77E+05	1.25E+05	2.75E+05	2.56E+05	5.77E+04
	Std	<b>7.89E+03</b>	2.62E+04 (+)	1.39E+04 (+)	2.74E+04 (+)	1.25E+05 (+)	2.88E+04 (+)	2.59E+04 (+)	8.89E+03 (+)
F4	Mean	1.25E+02	9.92E+01	9.45E+01	1.08E+02	<b>6.50E+01</b>	9.70E+01	1.01E+02	1.43E+02
	Std	4.60E+01	2.58E+01 (-)	2.91E+01 (-)	<b>1.14E+01</b> (=)	4.71E+01 (-)	2.72E+01 (-)	2.63E+01 (-)	4.29E+01 (=)
F5	Mean	<b>3.86E+01</b>	2.51E+02	3.76E+02	3.91E+02	1.57E+02	3.81E+02	3.78E+02	6.27E+01
	Std	<b>8.55E+00</b>	7.33E+01 (+)	1.58E+01 (+)	1.35E+01 (+)	7.47E+01 (+)	1.15E+01 (+)	1.49E+01 (+)	5.91E+01 (+)
F6	Mean	<b>2.87E-03</b>	3.64E-02	2.02E-01	7.01E-01	2.31E-02	2.61E-01	2.52E-01	2.85E-02
	Std	<b>8.86E-03</b>	1.76E-02 (+)	3.90E-02 (+)	1.11E-01 (+)	6.64E-02 (+)	7.45E-02 (+)	5.29E-02 (+)	5.04E-02 (+)
F7	Mean	1.36E+02	3.25E+02	4.40E+02	4.45E+02	2.54E+02	4.39E+02	4.39E+02	<b>8.99E+01</b>
	Std	8.95E+01	3.35E+01 (+)	<b>1.43E+01</b> (+)	1.75E+01 (+)	8.32E+01 (+)	1.65E+01 (+)	1.83E+01 (+)	4.22E+01 (=)
F8	Mean	4.55E+01	2.61E+02	3.79E+02	3.87E+02	1.66E+02	3.83E+02	3.83E+02	<b>4.45E+01</b>
	Std	3.51E+01	4.39E+01 (+)	1.60E+01 (+)	1.49E+01 (+)	1.16E+02 (+)	1.42E+01 (+)	1.17E+01 (+)	<b>1.08E+01</b> (=)
F9	Mean	1.44E+00	<b>1.51E-02</b>	5.00E-01	2.51E+00	1.99E+00	4.92E-01	5.40E-01	5.09E+00
	Std	1.78E+00	<b>8.29E-02</b> (-)	2.80E-01 (-)	1.07E+00 (+)	3.37E+00 (=)	2.35E-01 (-)	2.55E-01 (-)	8.73E+00 (+)
F10	Mean	1.26E+04	1.05E+04	1.29E+04	8.76E+03	1.03E+04	1.32E+04	1.32E+04	<b>7.97E+03</b>
	Std	5.06E+02	1.23E+03 (-)	<b>3.09E+02</b> (+)	1.82E+03 (-)	1.99E+03 (-)	6.61E+02 (+)	5.05E+02 (+)	3.75E+03 (-)
F11	Mean	<b>7.05E+01</b>	1.52E+02	1.73E+02	1.78E+02	1.30E+02	1.68E+02	1.70E+02	9.63E+01
	Std	2.15E+01	1.99E+01 (+)	<b>7.40E+00</b> (+)	8.14E+00 (+)	7.68E+01 (+)	1.02E+01 (+)	1.02E+01 (+)	2.82E+01 (+)
F12	Mean	1.06E+06	2.60E+07	8.33E+06	1.94E+07	<b>8.29E+04</b>	8.34E+06	8.49E+06	1.15E+06
	Std	6.44E+05	9.92E+06 (+)	3.73E+06 (+)	8.18E+06 (+)	<b>4.32E+04</b> (-)	4.12E+06 (+)	3.60E+06 (+)	6.40E+05 (=)
F13	Mean	8.24E+02	<b>7.98E+02</b>	1.18E+03	2.81E+03	1.12E+04	1.14E+03	1.17E+03	1.68E+03
	Std	8.83E+02	3.89E+02 (-)	1.88E+02 (+)	1.71E+02 (+)	1.08E+04 (+)	<b>1.45E+02</b> (+)	1.81E+02 (+)	9.78E+02 (+)
F14	Mean	<b>6.29E+01</b>	1.59E+02	1.45E+02	1.53E+02	1.54E+03	1.44E+02	1.44E+02	1.10E+02
	Std	1.71E+01	<b>6.60E+00</b> (+)	8.00E+00 (+)	7.54E+00 (+)	6.91E+03 (+)	9.26E+00 (+)	7.81E+00 (+)	2.88E+01 (+)
F15	Mean	9.95E+01	<b>9.52E+01</b>	1.49E+02	1.72E+02	5.52E+02	1.53E+02	1.50E+02	3.21E+02
	Std	3.95E+01	2.51E+01 (=)	1.14E+01 (+)	1.39E+01 (+)	1.32E+03 (+)	1.10E+01 (+)	1.06E+01 (+)	1.92E+02 (+)
F16	Mean	<b>5.47E+02</b>	2.32E+03	2.94E+03	3.02E+03	2.82E+03	2.93E+03	2.95E+03	8.44E+02
	Std	4.09E+02	8.05E+02 (+)	<b>1.81E+02</b> (+)	2.28E+02 (+)	7.71E+02 (+)	2.36E+02 (+)	2.06E+02 (+)	3.33E+02 (+)
F17	Mean	1.15E+03	7.50E+02	1.75E+03	1.84E+03	1.68E+03	1.80E+03	1.82E+03	<b>3.98E+02</b>
	Std	2.42E+02	3.03E+02 (-)	1.37E+02 (+)	1.47E+02 (+)	5.21E+02 (+)	2.04E+02 (+)	<b>1.20E+02</b> (+)	1.76E+02 (-)
F18	Mean	3.30E+04	2.29E+04	2.26E+03	1.13E+04	1.63E+05	2.11E+03	1.98E+03	5.96E+04
	Std	3.01E+04	1.09E+04 (=)	7.53E+02 (-)	6.27E+02 (-)	3.70E+05 (=)	9.15E+02 (-)	<b>5.94E+02</b> (-)	3.54E+04 (+)
F19	Mean	<b>2.93E+01</b>	5.31E+01	8.14E+01	8.77E+01	9.20E+01	8.09E+01	8.14E+01	7.21E+02
	Std	1.59E+01	1.23E+01 (+)	5.03E+00 (+)	6.29E+00 (+)	4.53E+01 (+)	<b>4.75E+00</b> (+)	5.12E+00 (+)	1.45E+03 (+)

Table 3 continued

	SPODE	ODE	QODE	QRODE	GODE	COODE	EODE	REODE	CODE	
F20	Mean	8.81E+02	6.47E+02	1.63E+03	1.22E+03	6.60E+02	1.61E+03	1.63E+03	<b>2.51E+02</b>	
	Std	3.39E+02	3.23E+02 (-)	1.47E+02 (+)	4.41E+02 (+)	2.89E+02 (-)	4.10E+02 (+)	<b>1.40E+02</b> (+)	1.42E+02 (-)	
F21	Mean	<b>2.37E+02</b>	4.71E+02	5.84E+02	5.80E+02	5.84E+02	3.74E+02	5.84E+02	2.45E+02	
	Std	<b>6.78E+00</b>	5.31E+01 (+)	1.38E+01 (+)	1.53E+01 (+)	1.34E+01 (+)	1.01E+02 (+)	1.01E+01 (+)	1.39E+01 (+)	
F22	Mean	1.19E+04	1.40E+04	2.50E+03	<b>1.00E+02</b>	8.71E+03	1.01E+04	1.33E+04	6.75E+03	
	Std	2.31E+03	3.91E+02 (+)	4.94E+03 (-)	<b>5.81E-01</b> (-)	3.13E+03 (-)	1.79E+03 (-)	3.52E+02 (+)	3.30E+03 (-)	
F23	Mean	<b>4.59E+02</b>	8.15E+02	8.03E+02	7.97E+02	8.07E+02	5.90E+02	7.98E+02	4.73E+02	
	Std	<b>1.30E+01</b>	1.85E+01 (+)	1.45E+01 (+)	1.33E+01 (+)	1.65E+01 (+)	8.47E+01 (+)	1.52E+01 (+)	1.50E+01 (+)	
F24	Mean	<b>5.33E+02</b>	8.68E+02	8.61E+02	8.64E+02	8.69E+02	7.41E+02	8.62E+02	5.54E+02	
	Std	<b>8.27E+00</b>	1.89E+01 (+)	1.37E+01 (+)	1.31E+01 (+)	1.25E+01 (+)	1.21E+02 (+)	1.38E+01 (+)	1.63E+01 (+)	
F25	Mean	5.33E+02	4.88E+02	4.81E+02	4.82E+02	4.83E+02	5.13E+02	4.81E+02	5.35E+02	
	Std	2.90E+01	1.86E+01 (-)	2.92E+00 (-)	3.52E+00 (-)	2.51E+00 (-)	3.72E+01 (-)	2.91E+00 (-)	<b>1.45E-01</b> (-)	
F26	Mean	<b>1.52E+03</b>	3.27E+03	4.79E+03	4.81E+03	4.86E+03	2.76E+03	4.79E+03	1.73E+03	
	Std	1.14E+02	7.24E+02 (+)	1.42E+02 (+)	1.61E+02 (+)	<b>1.13E+02</b> (+)	7.73E+02 (+)	1.61E+02 (+)	1.67E+02 (+)	
F27	Mean	5.66E+02	5.43E+02	5.19E+02	5.18E+02	5.34E+02	7.59E+02	5.21E+02	5.97E+02	
	Std	6.44E+01	1.32E+01 (=)	9.90E+00 (-)	1.11E+01 (-)	1.17E+01 (-)	1.75E+02 (+)	<b>8.28E+00</b> (-)	5.35E+01 (+)	
F28	Mean	4.98E+02	5.15E+02	6.50E+02	4.93E+02	5.14E+02	5.30E+02	6.41E+02	4.95E+02	
	Std	1.91E+01	2.82E+02 (+)	6.00E+02 (+)	1.83E+02 (-)	2.30E+02 (+)	1.47E+02 (=)	4.75E+02 (+)	2.13E+01 (=)	
F29	Mean	6.50E+02	2.18E+03	1.96E+03	1.94E+03	2.02E+03	1.28E+03	2.01E+03	<b>5.12E+02</b>	
	Std	2.80E+02	1.64E+02 (+)	1.73E+02 (+)	1.68E+02 (+)	1.95E+02 (+)	5.99E+02 (+)	<b>1.28E+02</b> (+)	1.90E+02 (=)	
F30	Mean	<b>6.02E+05</b>	1.33E+06	8.13E+05	7.73E+05	1.03E+06	8.19E+05	7.85E+05	6.07E+05	
	Std	<b>2.63E+04</b>	1.77E+05 (+)	5.35E+04 (+)	5.98E+04 (+)	1.35E+05 (+)	3.20E+05 (+)	6.93E+04 (+)	2.79E+04 (=)	
+/-/-			18/4/7	23/0/6	21/1/7	22/1/6	20/3/6	24/0/5	23/0/6	16/9/4

Bold values represent the best results in each row



**Table 4** Friedman test with Tukey–Kramer’s post hoc for ODE variants on CEC 2017 at 50-D

	Algorithm	Average ranking	$z$ value	$p$ value	Adj. $p$ value (Tukey–Kramer)	Sig.	Test statistics	
1	SPODE	<b>3.00</b>						
2	ODE	5.00	−4.00E+00	1.34E−04	9.43E−02	No	N	29
3	QODE	5.43	−4.87E+00	2.87E−06	1.38E−02	Yes	Chi-Square	42.03
4	QRODE	4.33	−2.67E+00	1.14E−02	6.01E−01	No	df	8
5	GODE	7.10	−8.20E+00	1.00E−15	2.18E−07	Yes	$p$ value	2.01E−07
6	COODE	4.80	−3.60E+00	6.12E−04	1.91E−01	No	Sig.	Yes
7	EODE	5.77	−5.53E+00	8.96E−08	2.25E−03	Yes		
8	REODE	5.63	−5.27E+00	3.78E−07	4.80E−03	Yes		
9	CODE	3.93	−1.87E+00	6.99E−02	9.19E−01	No		

SPODE has offered superior results than that of the eight ODE variants on both 30-D and 50-D.

### Run-time complexity

In this subsection, we used the widely known run-time complexity criterion to measure the efficiency of different algorithms [33]. In Tables 5 and 6, the run-time complexity results on the CEC 2017 at 30-D and 50-D are introduced. In Table 5, we can see that SPOBL slightly consumes more time than other OBL variants, and we can find a similar tendency on the CEC 2017 at 50-D in Table 6. SPODE has two main aspects that differ from other OBL variants in each iteration. First, the subpopulation strategy needs to select subpopulation based on the jumping rate, which is non-existent in other population-based OBL variants, so this strategy brings extra run-time. Moreover, the indexes of surviving individuals are calculated in the parameter control mechanism to guide the next generation of parameters, and this mechanism is executed in each iteration. As a result, SPOBL slightly consumes more run-time than other OBL variants, but the increased run-time is negligible compared to the performance improvement.

### Evaluation on the components in SPODE

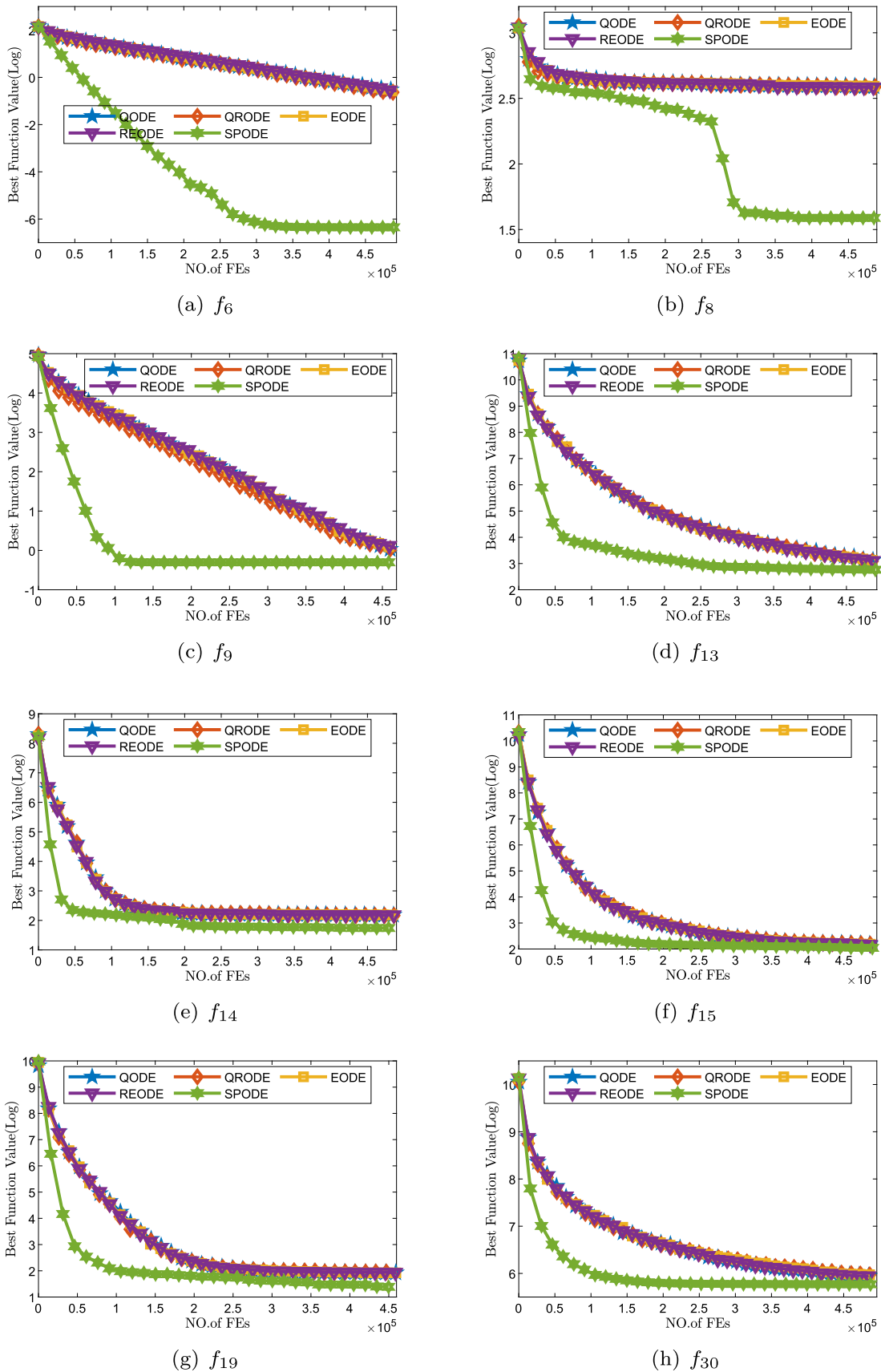
The new OBL variant is proposed in this paper includes two strategies: subpopulation-based opposition and self-adaptive parameter control. In this part, to compare the influence of different strategies on the proposed algorithm. SPODEwosa represents SPODE without a self-adaptive control strategy. SPODEwosp denotes SPODE without the subpopulation-based opposition.

Table 7 provides the results of mean, standard deviation and Wilcoxon test of the algorithms. SPODE achieves higher accuracy for 18 functions, SPODEwosa for 5 functions and SPODEwosp for 6 functions. For unimodal functions (F1, F3), SPODE significantly enhances the performance than its competitors. For multimodal functions (F4–F10), SPODE

achieves the best performance on half of the functions, and the dominance of SPODE’s performance on multimodal functions is not significant compared to other types of functions. This indicates that SPODE’s exploration ability is not enhanced despite its hybrid subpopulation and self-adaptive parameter strategies. For hybrid functions and composition functions (F11–F30), SPODE has achieved impressive performance improvements on these functions, except for F18 and F20. This means that SPODE has obtained a balance between exploration and exploitation on these functions. The average ranks of these algorithms on the 29 functions are 1.36, 2.16 and 2.36, respectively. It is clear that SPODEwosa performs much better than SPODEwosp.

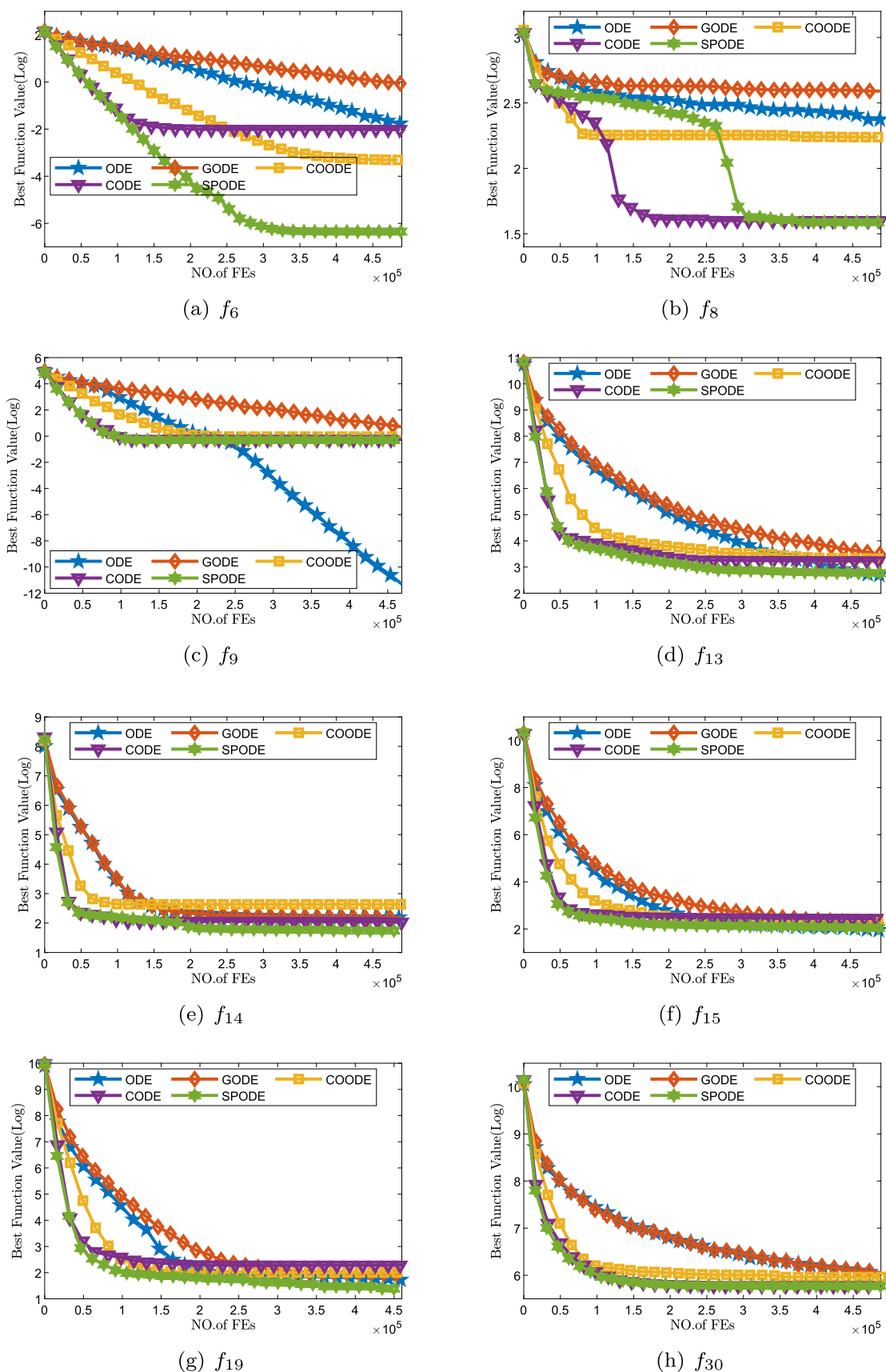
From the results of the Wilcoxon test listed in Table 7, compared with SPODEwosa, SPODE has significantly improved performance for 21 functions. Compared with SPODEwosp, SPODE has significantly improved performance for 19 functions, but decreased performance for 6 functions. SPODEwosp statistically outperforms SPODE on multimodal, hybrid and composition functions (F4, F9, F18, F25, F27 and F28), which means that the self-adaptive parameter control strategy can achieve a good ratio between exploration and exploitation on complex functions. So, SPODE is significantly better than SPODEwosp and SPODEwosa.

Figure 8 shows the convergence curves of SPODE, SPODEwosa and SPODEwosp. From the curve of F6, SPODE finds the global optimal, but both SPODEwosa and SPODEwosp fall into the local optimum. From the other subfigures, similar trends can be observed on F8, F11 and F29. SPODE and SPODEwosa achieve a similar performance. SPODEwosp converges to the local optimum. Both SPODE and SPODEwosa converge faster in the early stage and achieve the global optimal in the later stage, but SPODE always converges a little more rapidly than SPODEwosp. Moreover, it reveals that the influence of the subpopulation-based strategy on the proposed algorithm is more significant than that of the self-adaptive parameter control strategy. The main reason is that the subpopulation-based strategy can



**Fig. 6** Convergence curves of SPODE, QODE, QRODE, EODE and REODE on eight 50-D functions: Shifted and Rotated Lunacek Bi\_Rastrigin Function, Shifted and Rotated Levy Function, Shifted and

Rotated Schwefel’s Function, Hybrid Function 4 ( $N = 4$ ), Hybrid Function 5 ( $N = 4$ ), Hybrid Function 6 ( $N = 4$ ), Hybrid Function 6 ( $N = 4$ ) and Composition Function 10 ( $N = 3$ ), respectively



**Fig. 7** Convergence curves of SPODE, ODE, COODE, CODE and GODE on eight 50-D functions: Shifted and Rotated Lunacek Bi\_Rastrigin Function, Shifted and Rotated Levy Function, Shifted and

Rotated Schwefel’s Function, Hybrid Function 4 ( $N = 4$ ), Hybrid Function 5 ( $N = 4$ ), Hybrid Function 6 ( $N = 4$ ), Hybrid Function 6 ( $N = 6$ ) and Composition Function 10 ( $N = 3$ ), respectively

**Table 5** Run-time complexity for DE/rand/1/bin with OBL variants on CEC 2017 at 30-D

D	T0	T1	DE/rand/1/bin		OBL		QOBL		GOBL			
			SPOBL T2	(T2-T1)/T0	T2	(T2-T1)/T0	T2	(T2-T1)/T0	T2	(T2-T1)/T0	T2	(T2-T1)/T0
30	0.0654	0.4144	2.5265	32.2941	1.8594	22.0945	2.21690	27.5599	2.1615	26.7129	2.3864	30.1519
			COOBL		EOBL		REOBL		COBL			
			T2	(T2-T1)/T0	T2	(T2-T1)/T0	T2	(T2-T1)/T0	T2	(T2-T1)/T0	T2	(T2-T1)/T0
			1.9575	23.5936	2.2661	28.3133	2.1959	27.2395	1.8671	22.2121		

**Table 6** Run-time complexity for DE/rand/1/bin with OBL variants on CEC 2017 at 50-D

D	T0	T1	DE/rand/1/bin		OBL		QOBL		GOBL			
			SPOBL T2	(T2-T1)/T0	T2	(T2-T1)/T0	T2	(T2-T1)/T0	T2	(T2-T1)/T0	T2	(T2-T1)/T0
50	0.0654	0.9455	4.5556	55.1999	3.5410	39.6861	4.1796	49.4507	4.0763	47.8713	4.2874	51.0981
			COOBL		EOBL		REOBL		COBL			
			T2	(T2-T1)/T0	T2	(T2-T1)/T0	T2	(T2-T1)/T0	T2	(T2-T1)/T0	T2	(T2-T1)/T0
			3.7301	42.5773	4.2847	51.0570	4.1357	48.7784	3.5266	39.4658		

reduce the redundancy of calculation when solving complex functions.

### Setting of newly parameter

The jumping rate  $J$  is the most important parameter in the ODE and its variants. It affects the frequency of the opposition. ODE will degenerate to DE if  $J = 0$ . In the subpopulation-based strategy,  $J$  acts on individuals,  $\mu_J$  represents the mean value of  $J$ . Moreover, the Gaussian distribution with  $\mu_J$  as the mean generates a jumping rate for each individual. Thus,  $\mu_J$  affects the subpopulation size according to each individual's jumping rate. This paper proposes a self-adaptive formula for  $\mu_J$  to better adapt the algorithm during the evolutionary process. Furthermore, the arithmetic mean is extended to the generalized Lehmer mean in the Eq. (15). In the generalized Lehmer mean, the degree of this mean is expressed by the value of  $p$ . In this subsection, a proper setting of this new parameter is provided.

In Table 8, the results of SPODE with different  $p$  and time-varying strategies are introduced. SPODE with  $p = 0.5$  significantly outperforms other strategies on 19, 22, 22 and 21 functions. The performance superiority of SPODE with  $p = 0.5$  gradually becomes more significant as the value of  $p$  increases. Compared with the time-varying strategy, generalized Lehmer mean with different  $p$  has significantly improved the performance on this test suite. In the previous section,  $p = 0.5$  indicates the value of  $\mu_J$  rapidly decreases, which means that subpopulation size gets smaller and smaller during the evolutionary process. When the evolution enters the later stage, the opposition will waste the fitness evaluations, so SPODE with a negative strategy ( $p = 0.5$ ) has a more significant advantage than other strategies.

The box plots of fitness error values of all strategies on six functions are depicted in Fig. 9 to show the distribution of SPOBL solutions under different strategies. These six functions are from different categories. All of them have the same trends that the performance superiority of SPODE gradually becomes more significant and stable as the value of  $p$  increases. From the figure, we can see SPOBL with  $p = 0.5$  demonstrates the capability to find promising solutions.

### Comparison for LSHADE with OBL variants

In the previous part, we combined SPOBL with DE to obtain SPODE. To further verify the performance of SPOBL with DE variants, we combined SPOBL with two state-of-the-art DE variant: LSHADE [40] and jSO [41], and compared the effects of OBL variants on the performance of LSHADE and jSO.

Since DE was proposed, many famous DE variants have been proposed. SHADE was proposed and won 3rd place in the CEC 2013 competition [42], and LSHADE won

1st place in the CEC 2014 competition [40]. SHADE, as a basic powerful DE variant, has been extensively studied by researchers, such as SPS-LSHADE-EIG [43], LSHADE-ND [44], iLSHADE [45], LSHADE-EpSin [46], jSO [41], L\_covnSHADE [47], LSHADE-cnEpSin [48], and LSHADE-RSP [49]. The jSO is an improved version of the iLSHADE algorithm [45], mainly with a new weighted version of mutation strategy. The new mutation strategy is called DE/current-to-pBest-w/1. At the early stage of the evolutionary, the smaller  $F$  is used, while the higher  $F$  is used at the later stage. For illustration simplicity, in this section LSHADE-SPOBL and LSHADE-OBL represent the algorithm after embedding SPOBL and OBL into LSHADE.

In this part, the performance results of LSHADE with OBL variants on CEC 2017 at 30-D and 50-D are presented. Tables 9 and 11 provide the results of mean, std and Wilcoxon test of algorithms at 30-D and 50-D, respectively. The results of the Friedman test are presented on Tables 10 and 12. From the tables, the following have been observed.

1. For the unimodal functions (F1, F3), all LSHADE-OBL variants achieve the best solution for all the functions. There is no significant difference in the performance of these algorithms on these functions.
2. For the multimodal functions (F4–F10), LSHADE-SPOBL finds the best performance on 3 and 2 functions at 30-D and 50-D, respectively. LSHADE performs better than SPOBL on most functions, while the other LSHADE-OBL variants are dominated by LSHADE-SPOBL and the LSHADE on multimodal functions.
3. For the hybrid functions (F11–F20), LSHADE-SPOBL keeps the outperformance and superiority of the performance on all functions, except on F16, F17 and F20. Moreover, the significance of the performance of LSHADE-SPOBL is enhanced as the dimensionality increases. LSHADE always surpasses SPOBL on F16, F17, and F20. It indicates that LSHADE-SPOBL weakens the performance of the LSHADE on these functions.
4. For the composition functions (F21–F30), LSHADE-SPOBL obtains the best accuracy for 3 and 4 functions at 30-D and 50-D, respectively. LSHADE for 1 and 2 functions. On composition functions, the dominance of LSHADE-SPOBL as well as the LSHADE decreases. It means that no single algorithm can achieve the best performance on all functions.

From Table 9, it can be seen that LSHADE-SPOBL performs better than other LSHADE-OBL variants. To be precise, LSHADE-SPOBL significantly finds more accurate solutions than LSHADE-OBL, LSHADE-QOBL, LSHADE-QROBL, LSHADE-GOBL, LSHADE-COBL, LSHADE-EOBL, LSHADE-REOBL, and LSHADE-COBL for more than half of the functions. Compared with LSHADE-

**Table 7** Results of SPODE, SPODEwosa and SPODEwosp on CEC 2017 at 50-D

	SPODE		SPODEwosa		vs. SPODE	SPODEwosp		vs. SPODE
	Mean	Std	Mean	Std		Mean	Std	
F1	<b>2.14E+03</b>	<b>1.86E+03</b>	2.07E+05	4.03E+05	+	5.71E+03	8.93E+03	=
F3	<b>3.89E+04</b>	<b>7.89E+03</b>	5.05E+04	1.04E+04	+	<b>2.77E+05</b>	2.70E+04	+
F4	1.25E+02	4.60E+01	1.57E+02	4.48E+01	+	<b>9.25E+01</b>	<b>3.99E+01</b>	-
F5	<b>3.86E+01</b>	<b>8.55E+00</b>	4.93E+01	1.05E+01	+	3.05E+02	5.76E+01	+
F6	<b>2.87E-03</b>	<b>8.86E-03</b>	1.91E-02	2.43E-02	+	9.67E-02	6.23E-02	+
F7	1.36E+02	8.95E+01	<b>9.62E+01</b>	4.81E+01	=	3.66E+02	<b>4.59E+01</b>	+
F8	<b>4.55E+01</b>	3.51E+01	4.76E+01	<b>1.06E+01</b>	+	3.04E+02	7.50E+01	+
F9	1.44E+00	1.78E+00	8.70E+00	1.84E+01	+	<b>1.55E-01</b>	<b>2.37E-01</b>	-
F10	1.26E+04	5.06E+02	<b>1.24E+04</b>	<b>4.50E+02</b>	=	1.32E+04	8.79E+02	+
F11	<b>7.05E+01</b>	2.15E+01	1.28E+02	3.58E+01	+	1.69E+02	<b>1.09E+01</b>	+
F12	<b>1.06E+06</b>	<b>6.44E+05</b>	1.74E+06	9.23E+05	+	7.25E+06	3.08E+06	+
F13	<b>8.24E+02</b>	8.83E+02	2.00E+03	1.50E+03	+	1.08E+03	<b>3.22E+02</b>	+
F14	<b>6.29E+01</b>	1.71E+01	1.11E+02	2.97E+01	+	1.46E+02	<b>6.62E+00</b>	+
F15	<b>9.95E+01</b>	3.95E+01	3.43E+02	1.70E+02	+	1.08E+02	<b>2.74E+01</b>	=
F16	<b>5.47E+02</b>	<b>4.09E+02</b>	7.49E+02	5.57E+02	=	2.77E+03	5.85E+02	+
F17	<b>1.15E+03</b>	<b>2.42E+02</b>	1.16E+03	2.87E+02	=	1.19E+03	5.57E+02	=
F18	3.30E+04	3.01E+04	4.87E+04	3.58E+04	+	<b>2.13E+03</b>	<b>1.04E+03</b>	-
F19	<b>2.93E+01</b>	<b>1.59E+01</b>	8.49E+02	1.77E+03	+	6.54E+01	1.63E+01	+
F20	8.81E+02	<b>3.39E+02</b>	<b>8.49E+02</b>	3.65E+02	=	9.23E+02	5.20E+02	=
F21	<b>2.37E+02</b>	<b>6.78E+00</b>	2.44E+02	1.02E+01	+	5.57E+02	3.24E+01	+
F22	1.19E+04	2.31E+03	<b>1.06E+04</b>	4.30E+03	=	1.33E+04	<b>4.46E+02</b>	+
F23	<b>4.59E+02</b>	1.30E+01	4.71E+02	1.59E+01	+	8.01E+02	<b>1.28E+01</b>	+
F24	<b>5.33E+02</b>	<b>8.27E+00</b>	5.47E+02	1.25E+01	+	8.58E+02	1.93E+01	+
F25	5.33E+02	2.90E+01	5.34E+02	3.76E+01	=	<b>4.82E+02</b>	<b>3.52E+00</b>	-
F26	<b>1.52E+03</b>	<b>1.14E+02</b>	1.71E+03	1.29E+02	+	4.27E+03	5.46E+02	+
F27	5.66E+02	6.44E+01	6.03E+02	3.62E+01	+	<b>5.16E+02</b>	<b>9.71E+00</b>	-
F28	4.98E+02	<b>1.91E+01</b>	5.22E+02	2.14E+01	+	<b>4.70E+02</b>	5.30E+01	-
F29	6.50E+02	2.80E+02	<b>5.41E+02</b>	3.18E+02	=	1.98E+03	<b>1.70E+02</b>	+
F30	<b>6.02E+05</b>	<b>2.63E+04</b>	6.24E+05	4.02E+04	+	7.87E+05	7.31E+04	+
+/-/-			21/8/0			19/4/6		
Average rank	<b>1.36</b>		2.16			2.46		

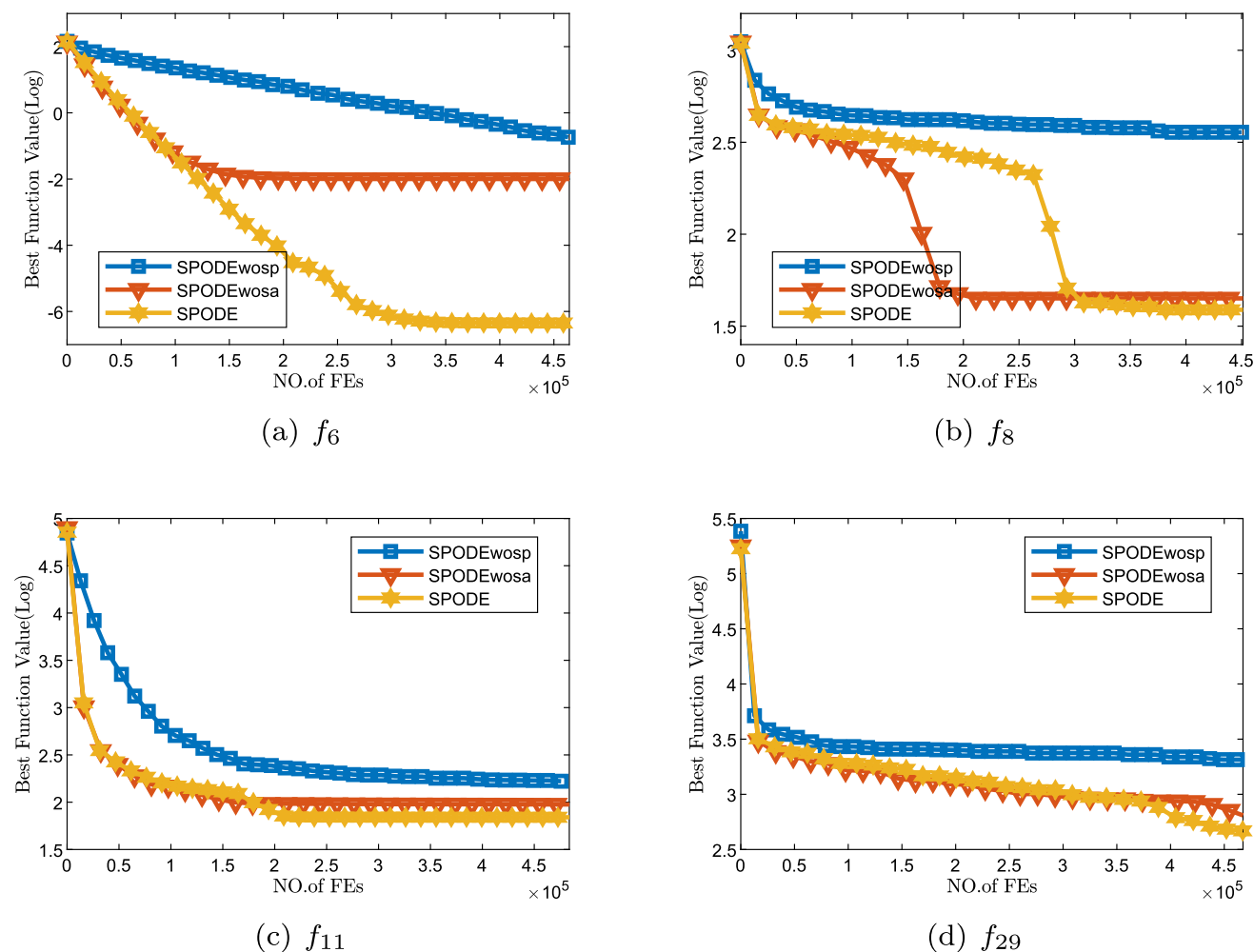
Bold values represent the best results in each row

COBL, LSHADE-SPOBL has significantly improved performance on 17 functions. Compared with LSHADE-GOBL, LSHADE-SPOBL has significantly improved performance on 23 functions. Compared with LSHADE-COBL, LSHADE-SPOBL has significantly improved performance on 20 functions. Although there is no obvious difference between LSHADE-SPOBL and LSHADE on most functions, the performance of LSHADE-SPOBL is still significantly better than that of LSHADE on the 5 functions. In other words, LSHADE-SPOBL performs much better than other LSHADE-OBL variants.

As shown in Table 10, the Friedman test results with Tukey–Kramer’s post hoc are provided. It can be seen from Table 10 that LSHADE-SPOBL ranks first among all algorithms, and the outperformance over OBL, QOBL, GOBL,

COOBL, EOBL, REOBL, and COBL is statistically significant. The top three algorithms are LSHADE-SPOBL, LSHADE, and LSHADE-QROBL. All in all, LSHADE-SPOBL is better than all comparison algorithms in performance on CEC 2017 at 30-D. In Table 12, a similar situation can be found at 50-D.

Moreover, we analyzed the results of Tables 9 and 11 based on the function’s properties. In the unimodal functions (F1, F3), the performance gap between each algorithm is not significant. However, the performance of LSHADE-SPOBL is better than other algorithms on hybrid functions (F11–F20) and composition functions (F21–F30). LSHADE assisted by SPOBL performs better than LSHADE on F13, F19, F22, F23, F24, F26 and F27. It reveals that LSHADE-SPOBL achieves a well-balanced state between exploration



**Fig. 8** Convergence curves of SPODE, SPODEwosa and SPODEwosp at four 50-D functions: Shifted and Rotated Lunacek Bi\_Rastrigin Function, Shifted and Rotated Levy Function, Hybrid Function 2 ( $N = 3$ ) and Composition Function 9 ( $N = 3$ ), respectively

and exploitation, and finds the best solution than other algorithms when dealing with complex problems.

In this part, the performance results of jSO with different OBL variants on CEC 2017 at 30-D and 50-D are presented. Tables 13 and 15 provide the results of mean, standard deviation and Wilcoxon test of algorithms at 30-D and 50-D, respectively. From the tables, we can observe the following.

1. For unimodal functions (F1, F3), most jSO-OBL variants provide similar performance on F1 and F3. jSO-SPOBL is slight better than jSO-OBL, jSO-GOBL and jSO-COOBL on F3 at 50-D. Thus, the scalability of jSO-SPOBL is enhanced by the dimensionality increases.
2. For multimodal functions (F4-F10), jSO-SPOBL finds the best solution for 5 and 6 functions at 30-D and 50-D, respectively. jSO-SPOBL significantly exhibits better performance than other algorithms on the most functions. jSO-SPOBL outperforms the comparison algorithm in terms of both mean and std.

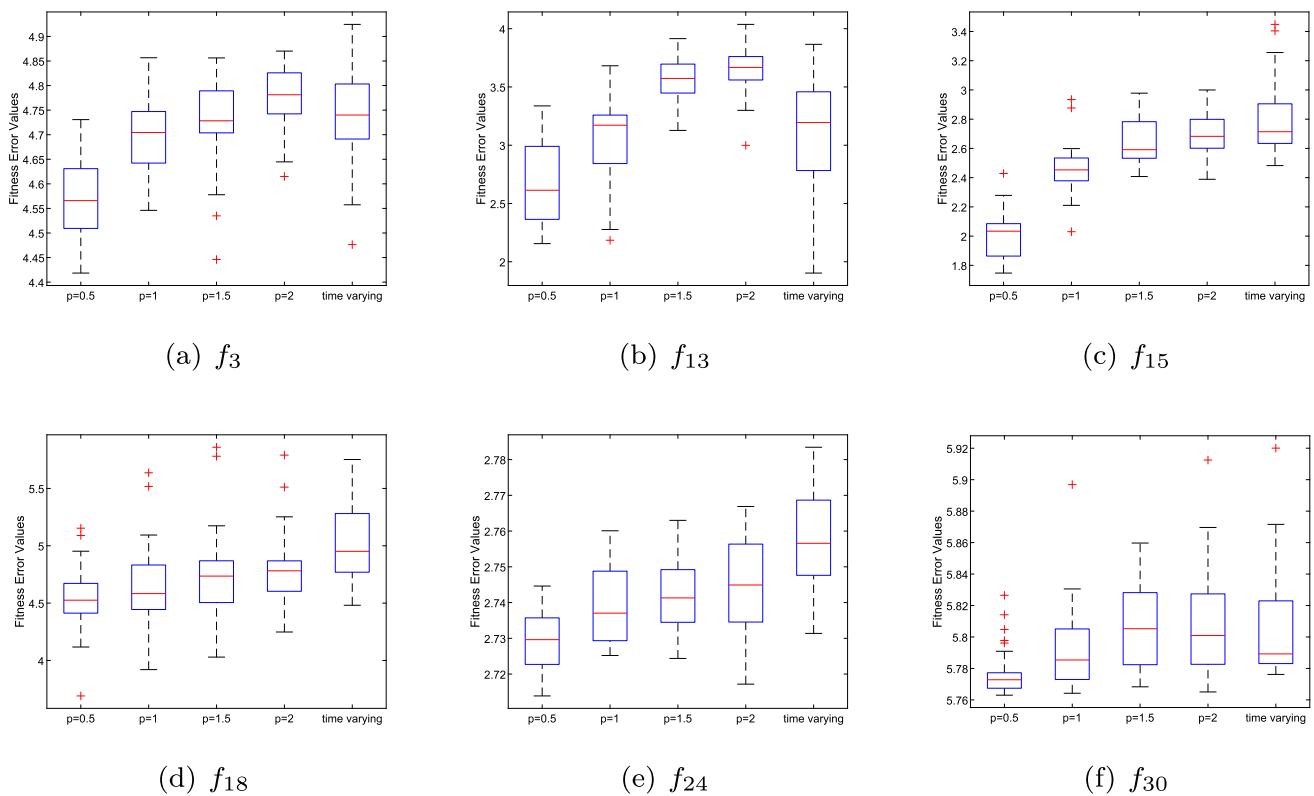
3. For hybrid functions (F11–F20), jSO-SPOBL finds the best solution for 6 and 3 functions at 30-D and 50-D, respectively. jSO statistically performs better than jSO-SPOBL on F12, F18 and F19. jSO-SPOBL statistically surpasses jSO on F13, F16, F17 and F20. Although the number of best value of the mean values of jSO-SPOBL on these functions is not as many as that of jSO, it does not hurt the overall performance of jSO-SPOBL on these functions.
4. For composition functions (F21–F30), jSO-SPOBL provides the best mean values for 5 and 2 functions at 30-D and 50-D, respectively. jSO-SPOBL is superior to jSO, jSO-OBL, jSO-QOBL, jSO-QROBL, jSO-GOBL, jSO-COOBL, jSO-EOBL, jSO-REOBL and jSO-COBL on 3, 5, 8, 7, 7, 8, 6, 7 and 5 functions at 30-D, respectively. In addition, similar results to LSHADE at 50-D are obtained in this section.

**Table 8** Results of SPODE with different value of  $p$  and time-varying strategies on CEC 2017 at 50-D

	$p = 0.5$			$p = 1$			$p = 1.5$			$p = 2$			Time varying		
	Mean	Std	Wil	Mean	Std	Wil	Mean	Std	Wil	Mean	Std	Wil	Mean	Std	Wil
F1	<b>3.06E+03</b>	<b>3.69E+03</b>	+	3.22E+06	1.38E+07	+	3.12E+06	6.60E+06	+	7.63E+06	2.02E+07	+	2.16E+07	4.92E+07	+
F3	<b>3.73E+04</b>	<b>6.92E+03</b>	+	5.00E+04	8.35E+03	+	5.39E+04	1.08E+04	+	6.07E+04	9.30E+03	+	5.51E+04	1.09E+04	+
F4	<b>1.17E+02</b>	5.98E+01	=	1.43E+02	4.84E+01	=	1.72E+02	4.73E+01	+	1.75E+02	<b>4.50E+01</b>	+	1.40E+02	4.90E+01	=
F5	<b>3.52E+01</b>	<b>8.01E+00</b>	+	4.95E+01	1.09E+01	+	5.85E+01	3.11E+01	+	5.16E+01	1.15E+01	+	6.23E+01	1.17E+01	+
F6	<b>2.88E-03</b>	<b>8.70E-03</b>	+	1.93E-02	2.19E-02	+	1.01E-01	1.07E-01	+	1.57E-01	1.89E-01	+	6.47E-01	5.86E-01	+
F7	1.07E+02	6.92E+01	-	9.94E+01	4.69E+01	-	<b>8.90E+01</b>	<b>6.77E+00</b>	-	9.68E+01	3.17E+01	-	1.05E+02	1.62E+01	-
F8	<b>4.46E+01</b>	3.18E+01	+	5.78E+01	3.95E+01	+	5.27E+01	1.31E+01	+	6.14E+01	<b>1.31E+01</b>	+	6.35E+01	1.45E+01	+
F9	<b>7.63E-01</b>	<b>1.05E+00</b>	+	6.31E+00	5.26E+00	+	1.13E+01	8.95E+00	+	1.64E+01	1.10E+01	+	5.76E+01	4.92E+01	+
F10	1.25E+04	<b>5.75E+02</b>	-	1.01E+04	2.64E+03	-	<b>8.72E+03</b>	3.25E+03	-	9.42E+03	3.16E+03	-	1.04E+04	3.10E+03	-
F11	<b>7.38E+01</b>	<b>2.13E+01</b>	+	1.03E+02	2.92E+01	+	1.26E+02	3.38E+01	+	1.34E+02	3.20E+01	+	1.55E+02	4.74E+01	+
F12	<b>1.15E+06</b>	<b>5.15E+05</b>	=	1.38E+06	6.23E+05	=	2.03E+06	9.26E+05	+	2.14E+06	8.06E+05	+	2.13E+06	8.77E+05	+
F13	<b>6.83E+02</b>	<b>5.64E+02</b>	+	1.61E+03	1.26E+03	+	4.18E+03	1.87E+03	+	4.97E+03	2.08E+03	+	2.20E+03	2.09E+03	+
F14	<b>5.51E+01</b>	<b>9.30E+00</b>	+	1.06E+02	2.76E+01	+	1.53E+02	9.96E+01	+	1.84E+02	1.25E+02	+	5.54E+02	7.53E+02	+
F15	<b>1.10E+02</b>	<b>4.71E+01</b>	+	3.11E+02	1.52E+02	+	4.66E+02	1.91E+02	+	5.13E+02	1.80E+02	+	7.49E+02	6.14E+02	+
F16	7.35E+02	5.32E+02	=	<b>4.29E+02</b>	<b>1.53E+02</b>	=	6.29E+02	1.96E+02	=	5.80E+02	2.39E+02	=	7.60E+02	2.82E+02	=
F17	1.11E+03	<b>2.92E+02</b>	-	6.33E+02	3.67E+02	-	5.79E+02	4.21E+02	-	<b>5.11E+02</b>	3.69E+02	-	6.67E+02	4.19E+02	-
F18	<b>4.20E+04</b>	<b>3.13E+04</b>	=	6.92E+04	9.13E+04	=	9.63E+04	1.58E+05	+	8.86E+04	1.15E+05	+	1.33E+05	1.07E+05	+
F19	<b>2.90E+01</b>	<b>1.48E+01</b>	+	9.29E+02	1.68E+03	+	1.95E+03	1.90E+03	+	3.07E+03	3.74E+03	+	5.43E+03	3.49E+03	+
F20	7.57E+02	3.57E+02	-	3.96E+02	4.31E+02	-	2.41E+02	2.83E+02	-	<b>1.56E+02</b>	<b>1.96E+02</b>	-	3.67E+02	4.10E+02	-
F21	<b>2.36E+02</b>	<b>7.15E+00</b>	+	2.43E+02	9.42E+00	+	2.47E+02	9.16E+00	+	2.49E+02	1.22E+01	+	2.65E+02	1.23E+01	+
F22	1.22E+04	<b>2.40E+03</b>	-	8.89E+03	3.27E+03	-	9.38E+03	2.48E+03	-	<b>7.28E+03</b>	3.58E+03	-	8.69E+03	4.66E+03	-
F23	<b>4.60E+02</b>	<b>1.15E+01</b>	+	4.71E+02	1.34E+01	+	4.75E+02	1.49E+01	+	4.82E+02	1.75E+01	+	4.95E+02	2.03E+01	+
F24	<b>5.37E+02</b>	<b>1.14E+01</b>	+	5.48E+02	1.26E+01	+	5.54E+02	1.32E+01	+	5.55E+02	1.68E+01	+	5.72E+02	1.90E+01	+
F25	<b>5.23E+02</b>	<b>2.18E+01</b>	+	5.38E+02	2.73E+01	+	5.45E+02	2.58E+01	+	5.50E+02	2.79E+01	+	5.49E+02	2.71E+01	+
F26	<b>1.56E+03</b>	1.61E+02	+	1.75E+03	1.76E+02	+	1.76E+03	<b>1.51E+02</b>	+	1.87E+03	1.66E+02	+	2.19E+03	1.87E+02	+
F27	<b>5.59E+02</b>	<b>4.80E+01</b>	+	6.15E+02	9.00E+01	+	6.38E+02	6.28E+01	+	6.31E+02	7.22E+01	+	6.64E+02	8.67E+01	+
F28	<b>5.02E+02</b>	<b>1.41E+01</b>	+	5.23E+02	2.13E+01	+	5.37E+02	3.03E+01	+	5.50E+02	2.98E+01	+	5.26E+02	3.04E+01	+
F29	6.34E+02	3.04E+02	-	4.34E+02	1.24E+02	-	<b>4.34E+02</b>	<b>8.20E+01</b>	=	4.94E+02	1.13E+02	=	5.73E+02	2.29E+02	=
F30	<b>6.00E+05</b>	<b>2.25E+04</b>	+	6.22E+05	4.13E+04	+	6.42E+05	3.96E+04	+	6.46E+05	5.25E+04	+	6.40E+05	5.18E+04	+
+/-/-				19/4/6			22/2/5			22/2/5			21/3/5		
Average rank	<b>1.96</b>			2.30			2.93			3.56			4.23		

Bold values represent the best results in each row





**Fig. 9** Boxplots of SPODE with different  $p$  and time-varying strategies at six functions at 50-D

Some observations can be obtained from the Wilcoxon test results in Tables 13 and 15. jSO-SPOBL performs better than other jSO-OBL variants. To be precise, jSO-SPOBL significantly finds more accurate solutions than jSO-OBL, jSO-QOBL, jSO-QROBL, jSO-GOBL, jSO-COOBL, jSO-EOBL, jSO-REOBL, and jSO-COBL for more than half of the functions. Compared with jSO-QOBL, jSO-SPOBL has significantly improved performance on 22 functions. Compared with jSO-COOBL, jSO-SPOBL has significantly improved performance on 23 functions. Compared with jSO-GOBL, jSO-SPOBL has significantly improved performance on 21 functions. Although there is no obvious difference between jSO-SPOBL and jSO on most functions, the performance of jSO-SPOBL is still significantly better than that of jSO on the 4 problems. In other words, jSO-SPOBL performs much better than all other jSO-OBL variants.

As shown in Table 14, the results of the Friedman test with Tukey–Kramer’s post hoc are provided. It can be seen from Table 14 that jSO-SPOBL ranks first among all algorithms, and the outperformance over jSO-OBL variants is statistically significant. The top three algorithms are jSO-SPOBL, jSO, jSO-EOBL. In Table 16, it can be concluded that jSO-SPOBL still surpasses other jSO-OBL variants, except for jSO-QOBL, jSO-QROBL, jSO-EOBL and jSO-REOBL. jSO-SPOBL performs better than these four jSO-OBL vari-

ants on 22, 19, 18 and 20 functions at 30-D, respectively. jSO-SPOBL statistically finds the best performance of these four jSO-OBL variants on 12, 11, 13 and 11 functions at 50-D, respectively. However, this does not affect the significant performance of SPOBL among all OBL variants, which indicates that QOBL, QROBL, EOBL and REOBL embedding into jSO exhibit excellent performance with increasing dimensionality. Although SPOBL ranks first, the outperformance over jSO-OBL, jSO-GOBL, jSO-COOBL and jSO-COBL is statistically significant. The top three algorithms are jSO-SPOBL, jSO-QROBL and jSO-REOBL. Thus, jSO-SPOBL is better than all comparison algorithms in performance on CEC 2017 at 30-D and 50-D.

These improvements reveal that SPOBL benefits the performance of jSO and LSHADE. It can be explained by the following reasons. On the one hand, SPOBL employs a sub-population strategy to select proper individuals to participate in the opposition, which can increase the probability of selecting individuals suitable for the opposition using the jumping rate of each individual during the evolutionary process. On the other hand, the self-adaptive parameter mechanism makes full use of the previous generation’s information, since the parameters of the surviving individuals in the previous iteration are very valuable experience for parameter control. Therefore, SPOBL has good scalability which adapts to dif-

**Table 9** Results of LSHADE with OBL variants on CEC 2017 at 30-D

	LSHADE-SPOBL	LSHADE	LSHADE-OBL	LSHADE-QOBL	LSHADE-QOBL	LSHADE-GOBL	LSHADE-COBL	LSHADE-EOBL	LSHADE-REOBL	LSHADE-COBL
F1	Mean	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Std	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)
F3	Mean	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.36E+03	0.00E+00	0.00E+00
	Std	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	1.11E-14 (=)	9.70E+03 (=)	0.00E+00 (=)	0.00E+00 (=)
F4	Mean	5.86E+01	5.86E+01	5.86E+01	5.86E+01	5.86E+01	<b>2.00E+01</b>	5.86E+01	5.86E+01	5.86E+01
	Std	2.78E-14	<b>2.41E-14</b> (=)	3.01E-14 (=)	3.41E-14 (=)	7.84E+00	3.18E+00 (-)	3.20E-14 (=)	3.11E-14 (=)	3.28E-06 (-)
F5	Mean	6.62E+00	<b>6.19E+00</b>	8.75E+00	7.29E+00	7.52E+00	4.38E+01	7.02E+00	7.33E+00	1.90E+01
	Std	<b>1.35E+00</b>	1.62E+00 (=)	3.65E+00 (+)	1.40E+00 (+)	1.66E+00 (+)	1.75E+01 (+)	1.58E+00 (=)	1.55E+00 (+)	9.24E+00 (+)
F6	Mean	<b>0.00E+00</b>	<b>0.00E+00</b>	7.59E-08	<b>0.00E+00</b>	4.53E-08	1.86E-07	3.15E-08	2.82E-08	3.30E-08
	Std	1.97E-08	<b>0.00E+00</b> (=)	2.68E-07 (+)	2.71E-08 (=)	1.65E-07 (+)	3.05E-07 (+)	1.55E-07 (=)	1.54E-07 (=)	1.56E-07 (=)
F7	Mean	<b>3.66E+01</b>	3.75E+01	3.85E+01	3.85E+01	3.85E+01	6.29E+01	3.80E+01	3.79E+01	4.62E+01
	Std	1.31E+00	<b>1.29E+00</b> (+)	2.80E+00 (+)	1.57E+00 (+)	1.33E+00 (+)	1.21E+01 (+)	1.47E+00 (+)	1.81E+00 (+)	5.86E+00 (+)
F8	Mean	6.96E+00	<b>6.56E+00</b>	9.18E+00	7.88E+00	7.94E+00	3.94E+01	7.74E+00	8.18E+00	1.74E+01
	Std	1.59E+00	<b>1.27E+00</b> (=)	3.72E+00 (+)	1.82E+00 (+)	1.89E+00 (+)	1.32E+01 (+)	1.45E+00 (+)	1.63E+00 (+)	8.84E+00 (+)
F9	Mean	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Std	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)
F10	Mean	<b>1.44E+03</b>	1.45E+03	2.38E+03	1.58E+03	1.52E+03	1.71E+03	1.54E+03	1.55E+03	1.78E+03
	Std	2.25E+02	2.00E+02 (=)	8.45E+02 (+)	2.27E+02 (+)	<b>1.88E+02</b> (=)	3.62E+02 (+)	2.16E+02 (+)	2.06E+02 (+)	6.91E+02 (+)
F11	Mean	1.65E+01	2.60E+01	2.16E+01	3.24E+01	2.56E+01	<b>1.49E+01</b>	2.58E+01	2.16E+01	2.49E+01
	Std	2.29E+01	2.81E+01 (=)	2.52E+01 (+)	2.90E+01 (+)	2.76E+01 (=)	<b>6.10E+00</b> (-)	2.76E+01 (+)	2.54E+01 (+)	2.73E+01 (+)
F12	Mean	9.52E+02	1.02E+03	1.21E+03	1.10E+03	1.09E+03	9.74E+02	1.04E+03	<b>9.40E+02</b>	1.11E+03
	Std	3.72E+02	4.08E+02 (=)	3.88E+02 (+)	3.55E+02 (=)	4.06E+02 (=)	3.93E+02 (=)	<b>3.34E+02</b> (=)	4.28E+02 (=)	4.17E+02 (=)
F13	Mean	<b>1.61E+01</b>	1.64E+01	1.74E+01	1.75E+01	1.75E+01	2.42E+01	1.87E+01	1.71E+01	1.86E+01
	Std	6.17E+00	6.09E+00 (=)	7.73E+00 (=)	6.44E+00 (=)	6.70E+00 (=)	9.10E+00 (+)	1.17E+01 (=)	6.07E+00 (=)	<b>5.92E+00</b> (=)
F14	Mean	<b>2.06E+01</b>	2.18E+01	2.36E+01	2.24E+01	2.20E+01	7.06E+03	2.23E+01	2.28E+01	2.33E+01
	Std	5.00E+00	1.30E+00 (=)	1.49E+00 (+)	<b>1.14E+00</b> (+)	3.14E+00 (+)	2.95E+04 (+)	3.16E+00 (+)	1.50E+00 (+)	4.24E+00 (+)
F15	Mean	<b>2.74E+00</b>	3.06E+00	6.45E+00	5.15E+00	5.09E+00	6.73E+00	4.44E+00	4.28E+00	4.85E+00
	Std	<b>1.49E+00</b>	1.83E+00 (=)	3.21E+00 (+)	2.91E+00 (+)	2.98E+00 (+)	3.08E+00 (+)	1.96E+00 (+)	1.98E+00 (+)	2.42E+00 (+)
F16	Mean	7.44E+01	<b>4.67E+01</b>	2.34E+02	1.40E+02	1.28E+02	7.75E+02	1.42E+02	1.52E+02	1.53E+02
	Std	6.61E+01	<b>3.89E+01</b> (-)	1.51E+02 (+)	1.07E+02 (+)	1.04E+02 (+)	3.38E+02 (+)	1.12E+02 (+)	1.06E+02 (+)	1.34E+02 (+)
F17	Mean	3.49E+01	<b>3.21E+01</b>	6.20E+01	3.96E+01	3.85E+01	4.32E+01	3.88E+01	3.91E+01	4.08E+01
	Std	6.34E+00	<b>6.12E+00</b> (-)	5.37E+01 (+)	6.79E+00 (+)	7.57E+00 (+)	1.47E+02 (+)	6.42E+00 (+)	8.11E+00 (+)	3.01E+01 (=)
F18	Mean	2.20E+01	<b>2.18E+01</b>	2.36E+01	2.25E+01	2.29E+01	1.76E+05	2.34E+01	2.31E+01	2.41E+01
	Std	<b>1.10E+00</b>	1.62E+00 (=)	2.16E+00 (+)	1.21E+00 (+)	1.84E+00 (+)	4.87E+05 (+)	1.94E+00 (+)	2.13E+00 (+)	3.94E+00 (+)
F19	Mean	<b>4.97E+00</b>	5.17E+00	6.60E+00	6.85E+00	6.97E+00	9.96E+00	6.67E+00	7.12E+00	5.77E+00
	Std	1.81E+00	1.44E+00 (=)	1.79E+00 (+)	1.92E+00 (+)	2.08E+00 (+)	3.82E+00 (+)	2.01E+00 (+)	2.00E+00 (+)	<b>1.41E+00</b> (+)

Table 9 continued

	LSHADE-SPOBL	LSHADE	LSHADE-OBL	LSHADE-QOBL	LSHADE-QOBL	LSHADE-GOBL	LSHADE-COBL	LSHADE-EOBL	LSHADE-REOBL	LSHADE-COBL
F20	Mean	<b>2.92E+01</b>	1.04E+02	4.19E+01	4.15E+01	6.49E+01	2.25E+02	4.36E+01	4.40E+01	5.09E+01
	Std	<b>5.42E+00</b> (–)	7.31E+01 (+)	9.38E+00 (+)	8.99E+00 (+)	4.38E+01 (+)	1.34E+02 (+)	9.56E+00 (+)	1.15E+01 (+)	4.37E+01 (=)
F21	Mean	2.07E+02	<b>2.04E+02</b>	2.08E+02	2.08E+02	2.09E+02	2.41E+02	2.08E+02	2.08E+02	2.16E+02
	Std	1.28E+00	2.15E+01 (=)	1.83E+00 (+)	1.69E+00 (+)	1.53E+00 (+)	1.57E+01 (+)	1.70E+00 (+)	1.71E+00 (+)	6.71E+00 (+)
F22	Mean	1.00E+02	1.00E+02	1.00E+02	1.00E+02	1.00E+02	1.00E+02	1.00E+02	1.00E+02	1.00E+02
	Std	1.00E–13 (=)	1.00E–13 (=)	1.00E–13 (=)	1.00E–13 (=)	1.00E–13 (=)	1.00E–13 (=)	1.11E–13 (=)	1.00E–13 (=)	1.00E–13 (=)
F23	Mean	<b>3.48E+02</b>	3.51E+02	3.51E+02	3.51E+02	3.51E+02	3.95E+02	3.50E+02	3.51E+02	3.68E+02
	Std	2.67E+00	<b>2.21E+00</b> (+)	5.74E+00 (+)	2.92E+00 (+)	2.88E+00 (+)	1.65E+01 (+)	2.64E+00 (+)	3.03E+00 (+)	1.17E+01 (+)
F24	Mean	<b>4.24E+02</b>	4.27E+02	4.27E+02	4.27E+02	4.27E+02	4.60E+02	4.27E+02	4.27E+02	4.40E+02
	Std	<b>1.38E+00</b>	3.85E+00 (+)	1.81E+00 (+)	1.98E+00 (+)	2.14E+00 (+)	1.32E+01 (+)	1.78E+00 (+)	1.99E+00 (+)	1.09E+01 (+)
F25	Mean	3.87E+02	3.87E+02	3.87E+02	3.87E+02	3.87E+02	<b>3.79E+02</b>	3.87E+02	3.87E+02	3.87E+02
	Std	<b>1.97E–02</b>	2.41E–02 (–)	3.26E–02 (=)	2.29E–02 (=)	2.44E–02 (=)	1.22E+00 (–)	2.79E–02 (–)	2.56E–02 (=)	2.40E–02 (+)
F26	Mean	9.18E+02	<b>8.39E+02</b>	9.54E+02	9.59E+02	9.79E+02	1.32E+03	9.51E+02	9.46E+02	1.10E+03
	Std	3.76E+01	<b>3.45E+01</b> (+)	3.89E+01 (+)	4.32E+01 (+)	4.00E+01 (+)	1.86E+02 (+)	3.89E+01 (+)	3.98E+01 (+)	9.56E+01 (+)
F27	Mean	<b>4.96E+02</b>	5.02E+02	5.02E+02	5.04E+02	5.05E+02	4.98E+02	5.05E+02	5.03E+02	4.99E+02
	Std	7.18E+00	6.13E+00 (+)	5.72E+00 (+)	5.12E+00 (+)	<b>4.65E+00</b> (+)	9.45E+00 (+)	5.27E+00 (+)	6.22E+00 (+)	8.19E+00 (+)
F28	Mean	3.37E+02	3.23E+02	3.28E+02	3.26E+02	3.40E+02	3.93E+02	<b>3.18E+02</b>	3.42E+02	3.28E+02
	Std	5.31E+01	4.88E+01 (=)	4.67E+01 (=)	4.66E+01 (=)	5.48E+01 (+)	7.79E+01 (+)	<b>4.36E+01</b> (=)	5.52E+01 (=)	4.88E+01 (=)
F29	Mean	4.34E+02	<b>4.32E+02</b>	4.40E+02	4.38E+02	4.44E+02	5.75E+02	4.40E+02	4.39E+02	4.36E+02
	Std	1.01E+01	<b>6.04E+00</b> (–)	1.01E+01 (+)	7.23E+00 (+)	1.05E+01 (+)	1.62E+02 (+)	7.49E+00 (+)	7.66E+00 (+)	1.06E+01 (=)
F30	Mean	1.98E+03	2.02E+03	1.99E+03	1.99E+03	2.01E+03	<b>6.22E+02</b>	2.00E+03	2.00E+03	2.01E+03
	Std	<b>4.47E+01</b>	6.45E+01 (=)	7.18E+01 (+)	5.49E+01 (=)	6.48E+01 (+)	2.04E+03 (–)	6.94E+01 (=)	5.10E+01 (=)	7.29E+01 (+)
+/=/–		5/19/5	20/8/1	18/11/0	16/13/0	23/6/0	20/5/4	17/11/1	18/11/0	17/11/1

Bold values represent the best results in each row

**Table 10** Friedman test with Tukey–Kramer’s post hoc for LSHADE with OBL variants on CEC 2017 at 30-D

Algorithm	Average ranking	<i>z</i> value	<i>p</i> value	Adj. <i>p</i> value (Tukey–Kramer)	Sig.	Test statistics
1 LSHADE-SPOBL	<b>2.72</b>					
2 LSHADE	3.25	-8.73E-01	2.73E-01	9.99E-01	No	N 29
3 LSHADE-OBL	6.82	-6.71E+00	6.71E-11	9.33E-07	Yes	Chi-Square 96.26
4 LSHADE-QOBL	5.25	-4.15E+00	7.40E-05	1.82E-02	Yes	df 9
5 LSHADE-QROBL	4.80	-3.41E+00	1.19E-03	1.17E-01	No	<i>p</i> value 8.95E-17
6 LSHADE-GOBL	7.48	-7.80E+00	2.45E-14	1.29E-07	Yes	Sig. Yes
7 LSHADE-COOBL	7.68	-8.13E+00	1.81E-15	1.27E-07	Yes	
8 LSHADE-EOBL	5.25	-4.15E+00	7.40E-05	1.82E-02	Yes	
9 LSHADE-REOBL	5.03	-3.79E+00	3.02E-04	4.77E-02	Yes	
10 LSHADE-COBL	6.72	-6.55E+00	1.98E-10	1.90E-06	Yes	

ferent dimensional problems and portability which embeds into any DE variant.

**Comparison for jSO with OBL variants**

In this part, the performance results of jSO with different OBL variants on CEC 2017 at 30-D and 50-D are presented. Tables 13 and 15 provide the results of mean, standard deviation and Wilcoxon test of algorithms at 30-D and 50-D, respectively. From the tables, we can observe the following.

1. For unimodal functions (F1, F3), most jSO-OBL variants provide similar performance on F1 and F3. jSO-SPOBL is slight better than jSO-OBL, jSO-GOBL and jSO-COOBL on F3 at 50-D. Thus, the scalability of jSO-SPOBL is enhanced by the dimensionality increases.
2. For multimodal functions (F4-F10), jSO-SPOBL finds the best solution for 5 and 6 functions at 30-D and 50-D, respectively. jSO-SPOBL significantly exhibits better performance than other algorithms on the most functions. jSO-SPOBL outperforms the comparison algorithm in terms of both mean and std.
3. For hybrid functions (F11-F20), jSO-SPOBL finds the best solution for 6 and 3 functions at 30-D and 50-D, respectively. jSO statistically performs better than jSO-SPOBL on F12, F18 and F19. jSO-SPOBL statistically surpasses jSO on F13, F16, F17 and F20. Although the number of best value of the mean values of jSO-SPOBL on these functions is not as many as that of jSO, it does not hurt the overall performance of jSO-SPOBL on these functions.
4. For composition functions (F21-F30), jSO-SPOBL provides the best mean values for 5 and 2 functions at 30-D and 50-D, respectively. jSO-SPOBL is superior to jSO, jSO-OBL, jSO-QOBL, jSO-QROBL, jSO-GOBL, jSO-COOBL, jSO-EOBL, jSO-REOBL and jSO-COBL on 3,

5, 8, 7, 7, 8, 6, 7 and 5 functions at 30-D, respectively. In addition, similar results to LSHADE at 50-D are obtained in this section.

Some observations can be obtained from the Wilcoxon test results in Tables 13 and 15. jSO-SPOBL performs better than other jSO-OBL variants. To be precise, jSO-SPOBL significantly finds more accurate solutions than jSO-OBL, jSO-QOBL, jSO-QROBL, jSO-GOBL, jSO-COOBL, jSO-EOBL, jSO-REOBL, and jSO-COBL for more than half of the functions. Compared with jSO-QOBL, jSO-SPOBL has significantly improved performance on 22 functions. Compared with jSO-COOBL, jSO-SPOBL has significantly improved performance on 23 functions. Compared with jSO-GOBL, jSO-SPOBL has significantly improved performance on 21 functions. Although there is no obvious difference between jSO-SPOBL and jSO on most functions, the performance of jSO-SPOBL is still significantly better than that of jSO on the 4 problems. In other words, jSO-SPOBL performs much better than all other jSO-OBL variants.

As shown in Table 14, the results of the Friedman test with Tukey–Kramer’s post hoc are provided. It can be seen from Table 14 that jSO-SPOBL ranks first among all algorithms, and the outperformance over jSO-OBL variants is statistically significant. The top three algorithms are jSO-SPOBL, jSO, jSO-EOBL. In Table 16, it can be concluded that jSO-SPOBL still surpasses other jSO-OBL variants, except for jSO-QOBL, jSO-QROBL, jSO-EOBL and jSO-REOBL. jSO-SPOBL performs better than these four jSO-OBL variants on 22, 19, 18 and 20 functions at 30-D, respectively. jSO-SPOBL statistically finds the best performance of these four jSO-OBL variants on 12, 11, 13 and 11 functions at 50-D, respectively. However, this does not affect the significant performance of SPOBL among all OBL variants, which indicates that QOBL, QROBL, EOBL and REOBL embedding into jSO exhibit excellent performance with

**Table 11** Results of LSHADE with OBL variants on CEC 2017 at 50-D

	LSHADE-SPOBL	LSHADE	LSHADE-OBL	LSHADE-QOBL	LSHADE-QOBL	LSHADE-GOBL	LSHADE-COBL	LSHADE-EOBL	LSHADE-REOBL	LSHADE-COBL
F1	Mean	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Std	<b>4.91E-15</b>	7.21E-15 (=)	1.25E-14 (=)	8.12E-15 (=)	6.97E-15 (=)	2.94E-14 (=)	6.81E-15 (=)	8.08E-15 (=)	7.16E-15 (=)
F3	Mean	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Std	8.28E-14	7.27E-14 (=)	3.50E-13 (=)	8.79E-14 (=)	7.30E-14 (=)	1.42E-08 (=)	<b>6.59E-14</b> (=)	7.06E-14 (=)	1.26E-13 (=)
F4	Mean	6.34E+01	6.67E+01	<b>4.44E+01</b>	7.57E+01	6.60E+01	5.72E+01	8.05E+01	8.16E+01	5.97E+01
	Std	<b>4.19E+01</b>	4.84E+01 (=)	4.74E+01 (=)	4.75E+01 (=)	5.04E+01 (=)	4.83E+01 (=)	4.99E+01 (=)	4.88E+01 (=)	4.77E+01 (=)
F5	Mean	1.35E+01	<b>1.17E+01</b>	2.12E+01	1.36E+01	1.42E+01	5.45E+01	1.45E+01	1.45E+01	3.57E+01
	Std	2.47E+00	2.31E+00 (-)	7.17E+00 (+)	2.53E+00 (=)	<b>2.23E+00</b> (=)	1.75E+01 (+)	2.99E+00 (=)	2.57E+00 (=)	1.40E+01 (+)
F6	Mean	2.23E-07	<b>7.35E-08</b>	5.08E-04	8.84E-05	4.20E-07	3.57E-04	2.44E-07	4.37E-05	9.61E-05
	Std	4.48E-07	<b>1.69E-07</b> (-)	1.26E-03 (+)	3.36E-04 (=)	6.76E-07 (=)	1.25E-03 (+)	2.98E-07 (=)	2.38E-04 (=)	3.71E-04 (+)
F7	Mean	<b>6.16E+01</b>	6.34E+01	6.79E+01	6.50E+01	6.58E+01	9.80E+01	6.51E+01	6.54E+01	7.50E+01
	Std	<b>1.45E+00</b>	1.96E+00 (+)	4.33E+00 (+)	2.10E+00 (+)	2.16E+00 (+)	1.34E+01 (+)	2.32E+00 (+)	2.06E+00 (+)	1.11E+01 (+)
F8	Mean	1.45E+01	<b>1.12E+01</b>	2.02E+01	1.40E+01	1.48E+01	5.97E+01	1.43E+01	1.44E+01	2.79E+01
	Std	<b>2.07E+00</b>	2.28E+00 (-)	7.84E+00 (+)	3.28E+00 (=)	2.95E+00 (=)	1.94E+01 (+)	2.64E+00 (=)	3.12E+00 (=)	1.13E+01 (+)
F9	Mean	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	2.98E-03	0.00E+00
	Std	0.00E+00	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	2.27E-02 (=)	0.00E+00 (=)	1.63E-02 (=)	0.00E+00 (=)
F10	Mean	<b>3.17E+03</b>	3.19E+03	5.17E+03	3.26E+03	3.30E+03	5.85E+03	3.46E+03	3.47E+03	4.60E+03
	Std	2.87E+02	<b>2.44E+02</b> (=)	1.08E+03 (+)	2.94E+02 (=)	2.98E+02 (=)	1.08E+03 (+)	3.37E+02 (+)	3.30E+02 (+)	1.48E+03 (+)
F11	Mean	<b>5.14E+01</b>	5.32E+01	6.64E+01	5.54E+01	5.59E+01	5.92E+01	5.45E+01	5.42E+01	5.57E+01
	Std	1.39E+01	<b>7.57E+00</b> (=)	1.23E+01 (+)	8.20E+00 (+)	1.13E+01 (+)	1.57E+01 (+)	9.74E+00 (=)	1.13E+01 (=)	1.01E+01 (+)
F12	Mean	2.11E+03	<b>2.11E+03</b>	2.68E+03	2.26E+03	2.16E+03	3.41E+03	2.24E+03	2.26E+03	2.18E+03
	Std	4.37E+02	5.05E+02 (=)	8.38E+02 (+)	4.92E+02 (=)	4.25E+02 (=)	2.21E+03 (+)	6.30E+02 (=)	<b>3.91E+02</b> (=)	3.94E+02 (=)
F13	Mean	<b>4.78E+01</b>	7.17E+01	7.45E+01	6.38E+01	6.70E+01	7.40E+01	6.04E+01	6.69E+01	6.96E+01
	Std	<b>1.88E+01</b>	3.57E+01 (+)	3.16E+01 (+)	3.69E+01 (=)	2.99E+01 (+)	2.76E+01 (+)	2.77E+01 (+)	3.73E+01 (=)	3.04E+01 (+)
F14	Mean	3.06E+01	<b>3.02E+01</b>	3.79E+01	3.31E+01	3.33E+01	2.31E+04	3.19E+01	3.25E+01	3.96E+01
	Std	3.34E+00	<b>3.18E+00</b> (=)	6.54E+00 (+)	5.09E+00 (=)	4.48E+00 (+)	9.31E+04 (+)	3.27E+00 (=)	3.18E+00 (+)	5.69E+00 (+)
F15	Mean	<b>4.01E+01</b>	4.14E+01	5.62E+01	4.68E+01	4.55E+01	5.84E+01	4.37E+01	4.21E+01	5.51E+01
	Std	1.02E+01	1.01E+01 (=)	1.70E+01 (+)	1.25E+01 (+)	1.09E+01 (=)	1.80E+01 (+)	<b>1.00E+01</b> (=)	1.01E+01 (=)	1.74E+01 (+)
F16	Mean	<b>4.13E+02</b>	4.23E+02	5.42E+02	5.22E+02	4.58E+02	1.18E+03	4.85E+02	5.28E+02	6.75E+02
	Std	1.32E+02	1.22E+02 (=)	2.23E+02 (+)	1.61E+02 (+)	<b>8.45E+01</b> (=)	4.02E+02 (+)	1.24E+02 (+)	1.14E+02 (+)	2.25E+02 (+)
F17	Mean	2.78E+02	<b>2.62E+02</b>	4.96E+02	3.38E+02	3.33E+02	8.89E+02	3.82E+02	3.48E+02	3.12E+02
	Std	7.70E+01	<b>6.85E+01</b> (=)	1.76E+02 (+)	8.79E+01 (+)	8.65E+01 (+)	3.72E+02 (+)	9.96E+01 (+)	8.47E+01 (+)	1.72E+02 (=)
F18	Mean	<b>3.82E+01</b>	4.43E+01	5.82E+01	4.77E+01	4.21E+01	6.14E+01	4.36E+01	4.79E+01	5.94E+01
	Std	1.06E+01	1.34E+01 (=)	2.31E+01 (+)	1.66E+01 (+)	1.18E+01 (=)	2.26E+01 (+)	<b>9.21E+00</b> (+)	1.93E+01 (+)	2.30E+01 (+)
F19	Mean	<b>2.32E+01</b>	3.06E+01	4.52E+01	3.24E+01	3.24E+01	4.54E+01	3.15E+01	3.17E+01	3.73E+01
	Std	<b>6.69E+00</b>	1.10E+01 (+)	1.53E+01 (+)	8.28E+00 (+)	1.00E+01 (+)	1.62E+01 (+)	9.66E+00 (+)	8.80E+00 (+)	8.46E+00 (+)

Table 11 continued

	LSHADE-SPOBL	LSHADE	LSHADE-OBL	LSHADE-QOBL	LSHADE-QOBL	LSHADE-GOBL	LSHADE-COBL	LSHADE-EOBL	LSHADE-REOBL	LSHADE-COBL
F20	Mean	<b>1.74E+02</b>	4.00E+02	2.38E+02	2.24E+02	3.56E+02	6.78E+02	2.83E+02	2.65E+02	2.02E+02
	Std	<b>6.26E+01 (=)</b>	2.19E+02 (+)	8.96E+01 (+)	9.11E+01 (=)	2.26E+02 (+)	3.26E+02 (+)	9.01E+01 (+)	9.51E+01 (+)	1.41E+02 (=)
F21	Mean	<b>2.13E+02</b>	2.22E+02	2.16E+02	2.16E+02	2.17E+02	2.55E+02	2.15E+02	2.15E+02	2.30E+02
	Std	2.57E+00 (=)	9.12E+00 (+)	2.77E+00 (+)	2.75E+00 (+)	3.01E+00 (+)	1.21E+01 (+)	2.76E+00 (+)	<b>2.23E+00 (+)</b>	1.05E+01 (+)
F22	Mean	1.91E+03	2.21E+03	1.00E+02	2.44E+02	<b>1.00E+02</b>	7.14E+03	2.11E+03	7.93E+02	4.14E+03
	Std	1.80E+03 (+)	3.06E+03 (=)	1.39E+00 (-)	7.89E+02 (-)	<b>2.23E-13 (-)</b>	1.46E+03 (+)	1.89E+03 (+)	1.50E+03 (=)	2.02E+03 (+)
F23	Mean	<b>4.30E+02</b>	4.44E+02	4.33E+02	4.34E+02	4.36E+02	4.90E+02	4.34E+02	4.34E+02	4.63E+02
	Std	4.72E+00	1.38E+01 (+)	4.94E+00 (=)	4.44E+00 (+)	4.40E+00 (+)	2.59E+01 (+)	5.60E+00 (+)	3.69E+00 (+)	1.08E+01 (+)
F24	Mean	<b>5.07E+02</b>	5.08E+02	5.17E+02	5.09E+02	5.11E+02	5.58E+02	5.10E+02	5.10E+02	5.29E+02
	Std	2.94E+00	8.14E+00 (+)	2.50E+00 (+)	2.88E+00 (+)	3.32E+00 (+)	2.65E+01 (+)	2.74E+00 (+)	3.70E+00 (+)	1.66E+01 (+)
F25	Mean	4.82E+02	4.85E+02	4.84E+02	4.87E+02	4.82E+02	<b>4.81E+02</b>	4.85E+02	4.83E+02	4.84E+02
	Std	4.46E+00	1.53E+01 (=)	1.53E+01 (=)	2.09E+01 (=)	7.45E+00 (=)	1.13E+01 (-)	1.49E+01 (=)	4.71E+00 (=)	1.03E+01 (=)
F26	Mean	<b>1.17E+03</b>	1.36E+03	1.19E+03	1.19E+03	1.23E+03	1.62E+03	1.18E+03	1.18E+03	1.38E+03
	Std	5.22E+01	1.03E+02 (+)	4.89E+01 (=)	5.54E+01 (=)	5.55E+01 (+)	2.21E+02 (+)	5.94E+01 (=)	4.58E+01 (=)	1.51E+02 (+)
F27	Mean	5.34E+02	5.49E+02	5.42E+02	5.35E+02	5.41E+02	<b>5.00E+02</b>	5.36E+02	5.37E+02	5.48E+02
	Std	1.49E+01	2.75E+01 (+)	2.50E+01 (=)	1.63E+01 (=)	2.40E+01 (=)	<b>3.80E-04 (-)</b>	1.91E+01 (=)	1.98E+01 (=)	2.72E+01 (+)
F28	Mean	<b>4.70E+02</b>	4.94E+02	4.78E+02	<b>4.70E+02</b>	4.72E+02	4.77E+02	4.78E+02	4.83E+02	4.81E+02
	Std	2.28E+01 (=)	2.19E+01 (+)	2.43E+01 (=)	2.10E+01 (=)	2.20E+01 (+)	2.30E+01 (=)	2.37E+01 (=)	2.46E+01 (=)	2.45E+01 (=)
F29	Mean	3.60E+02	3.82E+02	3.67E+02	3.63E+02	3.77E+02	6.25E+02	3.66E+02	3.67E+02	3.76E+02
	Std	1.42E+01	2.56E+01 (+)	1.34E+01 (+)	1.48E+01 (=)	1.58E+01 (+)	2.15E+02 (+)	1.27E+01 (=)	<b>1.23E+01 (+)</b>	2.89E+01 (+)
F30	Mean	6.45E+05	6.92E+05	6.82E+05	6.65E+05	6.71E+05	<b>4.15E+04</b>	6.66E+05	6.75E+05	6.45E+05
	Std	6.71E+04 (=)	7.26E+04 (+)	7.56E+04 (+)	9.45E+04 (=)	6.99E+04 (=)	8.80E+04 (-)	8.48E+04 (=)	9.42E+04 (=)	6.20E+04 (=)
+/-		5/20/4	23/6/0	12/16/1	8/20/1	21/7/1	22/4/3	12/17/0	12/17/0	19/10/0

Bold values represent the best results in each row

**Table 12** Friedman test with Tukey–Kramer’s post hoc for LSHADE with OBL variants on CEC 2017 at 50-D

Algorithm	Average ranking	$z$ value	$p$ value	Adj. $p$ value(Tukey– Kramer)	Sig.	Test statistics		
1	LSHADE-SPOBL	<b>2.38</b>						
2	LSHADE	2.93	-9.00E-01	2.66E-01	1.00E+00	No		
3	LSHADE-OBL	7.87	-8.97E+00	1.31E-18	1.27E-07	Yes	N	29
4	LSHADE-QOBL	5.07	-4.39E+00	2.60E-05	1.64E-02	Yes	Chi-Square	130.83
5	LSHADE-QROBL	4.58	-3.60E+00	6.12E-04	3.20E-01	No	df	9
6	LSHADE-GOBL	6.87	-7.34E+00	8.20E-13	1.39E-07	Yes	$p$ value	8.00E-24
7	LSHADE-COOBL	8.02	-9.22E+00	1.41E-19	1.27E-07	Yes	Sig.	Yes
8	LSHADE-EOBL	4.90	-4.12E+00	8.28E-05	3.85E-02	Yes		
9	LSHADE-REOBL	5.48	-5.07E+00	1.03E-06	3.91E-03	Yes		
10	LSHADE-COBL	6.90	-7.39E+00	5.48E-13	5.86E-07	Yes		

increasing dimensionality. Although SPOBL ranks first, the outperformance over jSO-OBL, jSO-GOBL, jSO-COOBL and jSO-COBL is statistically significant. The top three algorithms are jSO-SPOBL, jSO-QROBL and jSO-REOBL. Thus, jSO-SPOBL is better than all comparison algorithms in performance on CEC 2017 at 30-D and 50-D.

These improvements reveal that SPOBL benefits the performance of jSO and LSHADE. It can be explained by the following reasons. On the one hand, SPOBL employs a subpopulation strategy to select proper individuals to participate in the opposition, which can increase the probability of selecting individuals suitable for the opposition using the jumping rate of each individual during the evolutionary process. On the other hand, the self-adaptive parameter mechanism makes full use of the previous generation’s information, since the parameters of the surviving individuals in the previous iteration are very valuable experience for parameter control. Therefore, SPOBL has good scalability which adapts to different dimensional problems and portability which embeds into any DE variant.

### Real-world constrained optimization problems

To validate the effectiveness of SPOBL, it is essential that the performance is evaluated on real-world optimization problems and is also compared against the popular existing algorithms. The CEC 2020 test suite contains non-linear and non-convex constrained optimization problems [32]. Moreover, the majority of problems of CEC 2020 originate from real-world applications.

In this subsection, SPOBL is compared with three up-to-date metaheuristics which are all published on the proceedings of CEC 2020 and GECCO 2020.

1. SASS: Self-Adaptive Spherical Search Algorithm [50].

2. *sCMaGES*: Modified Covariance Matrix Adaptation Evolution Strategy [51].

3. *COLSHADE*: LSHADE for Constrained Optimization with Lévy Flight [52].

Among these three algorithms, SASS, *sCMaGES* and *COLSHADE* rank first, second and third, respectively. SASS is an excellent constraint optimization algorithm, and it is an interesting problem to embed SPOBL into SASS and observe the performance improvement. Besides, this paper discusses a new OBL, so there is not much discussion on constraint handling techniques. Since SPOBL is an OBL variant and is not an algorithm for solving constrained optimization problems, this section embeds SPOBL into SASS and the algorithm is denoted as SASS-SPOBL. In this subsection, the above-mentioned algorithms are run 25 times on each problem. The max fitness evaluations and performance evaluations are the same as [32]. The mean, std, median, best and worst values of fitness are shown in the following subsection.

### Planetary gear train design optimization problem

This problem is concerned with the design of the gear train to determine the number of teeth on each gear to obtain a velocity ratio between the input and output shafts [53]. The main objective of this problem is to minimize the maximum error in the gear ratio. Six integer variables and 11 constraints are contained in this problem. More details of this problem can be found in [32].

In Table 17, the outcomes of planetary gear train design optimization problem are reported for all algorithms. It can be seen that the performance of SASS-SPOBL is better than other algorithms. The top three mean fitness values are 0.529216, 0.530809 and 0.541026, which are SASS-SPOBL, *sCMaGES* and *COLSHADE*. SASS-SPOBL can yield a significant minimum fitness value of 0.525769, which is lower

Table 13 Results of jSO with OBL variants on CEC 2017 at 30-D

	jSO-SPOBL	jSO	jSO-OBL	jSO-QOBL	jSO-QROBL	jSO-GOBL	jSO-COOBL	jSO-EOBL	jSO-REOBL	jSO-COBL
F1	Mean	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Std	0.00E+00	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)
F3	Mean	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Std	0.00E+00	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)
F4	Mean	5.87E+01	5.86E+01	5.86E+01	5.86E+01	5.86E+01	<b>1.96E+01</b>	5.86E+01	5.86E+01	5.86E+01
	Std	7.78E-01	2.54E-14 (=)	3.60E-14 (-)	2.42E-14 (=)	4.06E-14 (-)	4.64E+00 (-)	<b>2.13E-14 (=)</b>	2.67E-14 (=)	3.87E-05 (-)
F5	Mean	<b>8.16E+00</b>	9.07E+00	2.44E+01	1.18E+01	1.15E+01	5.45E+01	1.02E+01	1.08E+01	1.81E+01
	Std	<b>2.48E+00</b>	2.93E+00 (=)	9.29E+00 (+)	4.03E+00 (+)	3.58E+00 (+)	1.64E+01 (+)	3.04E+00 (+)	2.91E+00 (+)	1.00E+01 (+)
F6	Mean	<b>0.00E+00</b>	<b>0.00E+00</b>	5.89E-08	5.51E-08	9.88E-08	3.09E-07	5.26E-08	6.14E-08	<b>0.00E+00</b>
	Std	<b>1.92E-08</b>	2.71E-08 (=)	2.15E-07 (+)	2.27E-07 (+)	2.54E-07 (+)	5.07E-07 (+)	1.76E-07 (+)	2.83E-07 (+)	2.68E-08 (=)
F7	Mean	<b>3.88E+01</b>	3.92E+01	4.89E+01	4.15E+01	4.02E+01	7.71E+01	4.06E+01	4.03E+01	4.68E+01
	Std	<b>2.54E+00</b>	2.80E+00 (=)	7.70E+00 (+)	6.32E+00 (+)	2.70E+00 (+)	1.46E+01 (+)	3.50E+00 (+)	2.93E+00 (+)	1.03E+01 (+)
F8	Mean	<b>8.04E+00</b>	9.32E+00	2.21E+01	1.15E+01	1.14E+01	5.05E+01	9.61E+00	1.06E+01	1.68E+01
	Std	<b>2.37E+00</b>	2.50E+00 (+)	6.29E+00 (+)	3.38E+00 (+)	3.71E+00 (+)	1.80E+01 (+)	2.85E+00 (+)	3.52E+00 (+)	9.94E+00 (+)
F9	Mean	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Std	0.00E+00	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)
F10	Mean	<b>1.79E+03</b>	1.90E+03	3.36E+03	2.72E+03	2.99E+03	3.74E+03	2.56E+03	2.35E+03	3.07E+03
	Std	4.49E+02	<b>4.48E+02 (=)</b>	5.65E+02 (+)	8.49E+02 (+)	7.48E+02 (+)	8.30E+02 (+)	6.36E+02 (+)	7.02E+02 (+)	6.18E+02 (+)
F11	Mean	<b>4.44E+00</b>	4.74E+00	1.27E+01	1.20E+01	5.29E+00	1.25E+01	9.77E+00	1.01E+01	8.44E+00
	Std	8.37E+00	8.33E+00 (=)	2.03E+01 (+)	2.03E+01 (+)	8.63E+00 (=)	<b>5.71E+00 (+)</b>	1.77E+01 (=)	1.73E+01 (+)	1.24E+01 (+)
F12	Mean	<b>1.74E+02</b>	2.08E+02	4.45E+02	3.25E+02	4.27E+02	5.34E+02	3.32E+02	3.35E+02	4.87E+02
	Std	<b>1.20E+02</b>	1.32E+02 (=)	2.42E+02 (+)	1.99E+02 (+)	2.22E+02 (+)	2.70E+02 (+)	1.94E+02 (+)	2.29E+02 (+)	2.44E+02 (+)
F13	Mean	1.76E+01	1.74E+01	1.77E+01	1.82E+01	2.17E+01	2.50E+01	<b>1.69E+01</b>	1.79E+01	1.74E+01
	Std	4.88E+00	5.86E+00 (=)	6.61E+00 (=)	5.49E+00 (=)	6.75E+00 (+)	6.71E+00 (+)	6.35E+00 (=)	5.46E+00 (=)	<b>4.41E+00 (=)</b>
F14	Mean	2.18E+01	<b>2.16E+01</b>	2.27E+01	2.27E+01	2.27E+01	3.44E+01	2.26E+01	2.25E+01	2.50E+01
	Std	2.91E+00	<b>1.11E+00 (-)</b>	3.60E+00 (+)	1.31E+00 (+)	1.38E+00 (+)	6.93E+00 (+)	1.32E+00 (=)	1.38E+00 (=)	4.37E+00 (+)
F15	Mean	1.32E+00	<b>1.25E+00</b>	3.20E+00	2.37E+00	3.78E+00	7.15E+00	2.40E+00	2.40E+00	2.71E+00
	Std	1.05E+00	<b>9.96E-01 (=)</b>	2.12E+00 (+)	1.77E+00 (+)	1.82E+00 (+)	4.05E+00 (+)	1.42E+00 (+)	1.45E+00 (+)	1.65E+00 (+)
F16	Mean	<b>4.23E+01</b>	4.36E+01	3.78E+02	1.42E+02	1.66E+02	6.74E+02	1.59E+02	1.26E+02	2.44E+02
	Std	<b>5.24E+01</b>	6.66E+01 (=)	3.09E+02 (+)	1.21E+02 (+)	1.31E+02 (+)	2.95E+02 (+)	1.32E+02 (+)	1.17E+02 (+)	2.31E+02 (+)
F17	Mean	<b>4.08E+01</b>	4.20E+01	9.29E+01	5.43E+01	5.94E+01	2.16E+02	5.34E+01	5.67E+01	5.02E+01
	Std	<b>9.00E+00</b>	9.31E+00 (=)	5.54E+01 (+)	1.49E+01 (+)	1.44E+01 (+)	1.62E+02 (+)	1.34E+01 (+)	1.22E+01 (+)	2.28E+01 (+)
F18	Mean	<b>2.04E+01</b>	2.09E+01	2.19E+01	2.13E+01	2.19E+01	2.45E+01	2.12E+01	2.11E+01	2.17E+01
	Std	2.83E+00	<b>4.94E-01 (=)</b>	1.21E+00 (+)	7.52E-01 (+)	9.99E-01 (+)	2.47E+00 (+)	5.48E-01 (+)	5.78E-01 (+)	9.80E-01 (+)
F19	Mean	5.87E+00	5.99E+00	7.33E+00	8.94E+00	8.45E+00	1.13E+01	8.72E+00	8.42E+00	<b>5.71E+00</b>
	Std	1.97E+00	<b>1.94E+00 (=)</b>	2.47E+00 (+)	2.29E+00 (+)	2.57E+00 (+)	4.09E+00 (+)	2.33E+00 (+)	2.25E+00 (+)	2.04E+00 (=)



Table 13 continued

		jSO-SPOBL	jSO	jSO-OBL	jSO-QOBL	jSO-QROBL	jSO-GOBL	jSO-COOBL	jSO-EOBL	jSO-REOBL	jSO-COBL
F20	Mean	<b>4.29E+01</b>	4.33E+01	1.41E+02	6.07E+01	6.06E+01	1.20E+02	2.70E+02	5.79E+01	5.63E+01	6.50E+01
	Std	<b>9.79E+00</b>	1.12E+01 (=)	1.07E+02 (+)	2.09E+01 (+)	1.64E+01 (+)	8.66E+01 (+)	1.68E+02 (+)	1.76E+01 (+)	1.77E+01 (+)	6.41E+01 (=)
F21	Mean	<b>2.09E+02</b>	2.10E+02	2.21E+02	2.12E+02	2.11E+02	2.12E+02	2.51E+02	2.11E+02	2.11E+02	2.22E+02
	Std	<b>2.13E+00</b>	2.25E+00 (=)	7.63E+00 (+)	3.91E+00 (+)	3.45E+00 (+)	4.04E+00 (+)	1.98E+01 (+)	3.52E+00 (+)	3.27E+00 (+)	1.01E+01 (+)
F22	Mean	1.00E+02	1.00E+02	1.00E+02	1.00E+02	1.00E+02	1.00E+02	1.00E+02	1.00E+02	1.00E+02	1.00E+02
	Std	<b>5.86E-14</b>	1.00E-13 (=)	1.00E-13 (=)	1.00E-13 (=)	1.00E-13 (=)	1.00E-13 (=)	1.17E-13 (=)	1.11E-13 (=)	1.00E-13 (=)	1.00E-13 (=)
F23	Mean	<b>3.55E+02</b>	3.56E+02	3.64E+02	3.59E+02	3.59E+02	3.57E+02	4.00E+02	3.57E+02	3.58E+02	3.73E+02
	Std	5.31E+00	<b>4.63E+00</b> (=)	1.09E+01 (+)	6.38E+00 (+)	5.66E+00 (+)	5.62E+00 (+)	2.00E+01 (+)	5.97E+00 (=)	6.75E+00 (+)	1.28E+01 (+)
F24	Mean	<b>4.29E+02</b>	4.31E+02	4.37E+02	4.32E+02	4.33E+02	4.33E+02	4.72E+02	4.31E+02	4.33E+02	4.41E+02
	Std	3.23E+00	<b>2.93E+00</b> (+)	6.78E+00 (+)	3.47E+00 (+)	3.19E+00 (+)	3.48E+00 (+)	1.88E+01 (+)	3.25E+00 (+)	2.98E+00 (+)	1.03E+01 (+)
F25	Mean	3.87E+02	3.87E+02	3.87E+02	3.87E+02	3.87E+02	3.87E+02	<b>3.79E+02</b>	3.87E+02	3.87E+02	3.87E+02
	Std	7.74E-03	6.70E-03 (=)	1.13E-02 (=)	7.64E-03 (=)	8.39E-03 (=)	9.44E-03 (=)	1.23E+00 (-)	7.15E-03 (=)	<b>6.27E-03</b> (=)	1.37E-02 (+)
F26	Mean	9.66E+02	1.01E+03	<b>6.90E+02</b>	1.03E+03	1.04E+03	1.03E+03	1.46E+03	1.02E+03	1.04E+03	1.14E+03
	Std	<b>5.05E+01</b>	5.37E+01 (+)	4.30E+02 (=)	5.57E+01 (+)	6.51E+01 (+)	6.70E+01 (+)	1.73E+02 (+)	7.16E+01 (+)	5.35E+01 (+)	1.18E+02 (+)
F27	Mean	<b>4.92E+02</b>	4.98E+02	5.02E+02	4.99E+02	5.00E+02	4.99E+02	4.98E+02	5.01E+02	4.99E+02	4.94E+02
	Std	9.42E+00	6.93E+00 (+)	7.28E+00 (+)	6.94E+00 (+)	7.04E+00 (+)	6.68E+00 (+)	1.01E+01 (+)	<b>6.35E+00</b> (+)	6.71E+00 (+)	9.90E+00 (=)
F28	Mean	3.11E+02	3.13E+02	3.13E+02	3.24E+02	3.11E+02	<b>3.04E+02</b>	3.40E+02	3.06E+02	3.20E+02	3.08E+02
	Std	3.30E+01	3.60E+01 (=)	3.60E+01 (=)	4.58E+01 (+)	3.36E+01 (=)	<b>2.23E+01</b> (-)	7.02E+01 (+)	2.54E+01 (=)	4.34E+01 (+)	2.88E+01 (=)
F29	Mean	<b>4.50E+02</b>	4.52E+02	4.72E+02	4.71E+02	4.66E+02	4.71E+02	6.10E+02	4.76E+02	4.70E+02	4.80E+02
	Std	2.39E+01	<b>1.62E+01</b> (=)	2.08E+01 (+)	3.22E+01 (+)	2.36E+01 (+)	2.55E+01 (+)	1.91E+02 (+)	3.22E+01 (+)	2.65E+01 (+)	7.03E+01 (=)
F30	Mean	1.97E+03	1.97E+03	1.97E+03	1.97E+03	1.97E+03	1.98E+03	1.80E+04	1.98E+03	1.97E+03	<b>1.88E+03</b>
	Std	<b>8.15E+00</b>	1.03E+01 (=)	3.49E+01 (=)	2.50E+01 (+)	3.05E+01 (+)	2.07E+01 (+)	1.19E+05 (+)	3.04E+01 (+)	2.68E+01 (=)	3.00E+02 (-)
+/-		4/24/1		19/9/1	22/7/0	19/10/0	21/6/2	23/4/2	18/11/0	20/9/0	16/11/2

Bold values represent the best results in each row

**Table 14** Friedman test with Tukey–Kramer’s post hoc for jSO with OBL variants on CEC 2017 at 30-D

	Algorithm	Average ranking	$z$ value	$p$ value	Adj. $p$ value (Tukey–Kramer)	Sig.	Test statistics
1	jSO-SPOBL	<b>2.52</b>					
2	jSO	2.97	-7.36E-01	3.04E-01	1.00E+00	No	N 29
3	jSO-OBL	7.17	-7.61E+00	1.07E-13	1.29E-07	Yes	Chi-Square 113.08
4	jSO-QOBL	6.20	-6.03E+00	5.16E-09	1.00E-05	Yes	df 9
5	jSO-QROBL	5.25	-4.47E+00	1.81E-05	4.78E-03	Yes	$p$ value 3.46E-20
6	jSO-GOBL	6.43	-6.41E+00	4.80E-10	1.74E-06	Yes	Sig. Yes
7	jSO-COOBL	8.38	-9.60E+00	3.88E-21	1.27E-07	Yes	
8	jSO-EOBL	4.95	-3.98E+00	1.44E-04	2.20E-02	Yes	
9	jSO-REOBL	5.08	-4.20E+00	5.89E-05	1.15E-02	Yes	
10	jSO-COBL	6.05	-5.78E+00	2.20E-08	3.00E-05	Yes	

than that of other algorithms. It means that SASS-SPOBL has a better performance than its competitors on this problem.

**Four-stage gear box problem**

This problem’s essential purpose is to minimize the gear box weight. The mixed-integer nonlinear programming formulation of this problem contains 22 design variables and 86 nonlinear constraints.

Table 18 shows the result of a four-stage gearbox problem. The best fitness values obtained by SASS-SPOBL and COLSHADE can reach 35.35923. Besides, the best and std values found by SASS-SPOBL are smaller than those obtained by other metaheuristics. It indicates that COLSHADE is the best algorithm for solving this problem. Besides, it also shows that SASS-SPOBL cannot significantly outperform the comparison algorithms in all problems due to the complexity of the real-world problems.

**SOPWM for 13-level inverters**

Synchronous optimal pulse-width modulation (SOP) is an up-and-coming technique for medium-voltage high-power drives. An optimization technique is used to minimize the harmonic distortion of the machine stator current while determining the switching angles. SOPWM can be expressed as a scaleable constrained optimization problem. For different levels of frequency inverters, the SOPWM problem is formulated by the following [32].

**Minimize:**

$$f = \frac{\sqrt{\sum_k (k^{-4}) \left(\sum_{i=1}^N s(i) \cos(k\alpha_i)\right)^2}}{6\sqrt{\sum_k k^{-4}}}$$

where,  $k = 5, 7, 11, 13 \dots 97$ ,  $N = \left\lfloor \frac{6f_{s,\max}}{f_m} \right\rfloor$  and (23)  
 $s = [1, 1, 1, -1, 1, -1, 1, -1, 1, 1, 1, 1]$ .

**subject to:**

$$g_i = \alpha_{i+1} - \alpha_i - 10^{-5} > 0, i = 1, 2, \dots N - 1$$

$$h_1 = 6m - \sum_{i=1}^N s(i) \cos(\alpha_i) = 0 \tag{24}$$

**with bounds:**

$$0 < \alpha_i < \frac{\pi}{2}, i = 1, 2, \dots N \tag{25}$$

Table 19 shows the best results by these algorithms. The four statistical values found by SASS-SPOBL are superior to the other algorithms, except for the best value. Moreover, the std value of SASS-SPOBL is smaller than that of the rest approaches, which means that SASS-SPOBL is more stable than other competitors. Therefore, the SASS-SPOBL is a very promising method to handle this problem.

**Conclusions**

In the OBL family, the opposition operator is very important. The opposition operator randomly maps the population to another location in the search space to increase the convergence speed of the algorithm. Many OBL variants are population based and can only ensure either exploration or exploitation. When they converge to the later stage, they will waste the fitness evaluations. Many strategies aim to reduce waste. However, the individual’s information on the opposition operator has a great influence on the effect of the operator, but there is no research concentrating on it. To achieve a better balance between exploration and exploitation, we proposed a novel OBL variant called SPOBL, which is subpopulation-based opposition. However, in the past years, many OBL variants have been proposed to enhance the performance of metaheuristic algorithms. The population-based opposition still has many disadvantages: (1) waste the fitness evaluations, and (2) select suitable parameters for OBL. The main idea of this paper is to overcome the above shortcomings. To reduce the calculation redundancy

**Table 15** Results of jSO with OBL variants on CEC 2017 at 50-D

	jSO-SPOBL	jSO	jSO-OBL	jSO-QOBL	jSO-QROBL	jSO-GOBL	jSO-COOBL	jSO-EOBL	jSO-REOBL	jSO-COBL
F1	Mean	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.06E-08	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Std	7.21E-15	3.71E-10 (=)	1.03E-14 (=)	1.28E-14 (=)	1.45E-08 (+)	2.76E-08 (=)	1.21E-14 (=)	1.37E-14 (=)	2.58E-14 (=)
F3	Mean	0.00E+00	0.00E+00	0.00E+00	0.00E+00	3.29E-08	1.81E-08	0.00E+00	0.00E+00	0.00E+00
	Std	<b>6.90E-14</b>	4.16E-12 (=)	1.00E-13 (=)	1.07E-13 (=)	3.85E-08 (+)	1.99E-08 (+)	9.55E-14 (=)	9.36E-14 (=)	3.63E-13 (=)
F4	Mean	<b>2.85E+01</b>	3.98E+01	3.72E+01	4.27E+01	7.90E+01	6.75E+01	6.28E+01	5.27E+01	4.62E+01
	Std	4.48E+01	<b>3.00E+01</b> (=)	3.73E+01 (+)	4.53E+01 (=)	4.69E+01 (+)	5.01E+01 (=)	5.53E+01 (=)	5.13E+01 (=)	3.90E+01 (=)
F5	Mean	<b>1.10E+01</b>	1.93E+01	3.44E+01	1.60E+01	1.92E+01	7.94E+01	1.57E+01	1.47E+01	3.33E+01
	Std	3.50E+00	<b>2.87E+00</b> (+)	9.73E+00 (+)	4.33E+00 (+)	5.14E+00 (+)	2.42E+01 (+)	6.69E+00 (+)	4.28E+00 (+)	1.56E+01 (+)
F6	Mean	<b>0.00E+00</b>	1.31E-06	9.75E-06	9.75E-07	4.09E-06	9.03E-06	7.92E-07	1.04E-06	6.48E-07
	Std	<b>4.95E-08</b>	2.39E-06 (+)	2.97E-05 (+)	2.16E-06 (+)	5.03E-06 (+)	1.38E-05 (+)	1.20E-06 (+)	1.56E-06 (+)	9.13E-07 (+)
F7	Mean	<b>6.20E+01</b>	6.87E+01	8.10E+01	6.60E+01	7.85E+01	1.28E+02	6.82E+01	6.71E+01	7.75E+01
	Std	<b>2.95E+00</b>	5.11E+00 (+)	9.22E+00 (+)	5.73E+00 (+)	1.34E+01 (+)	3.19E+01 (+)	7.77E+00 (+)	4.85E+00 (+)	1.25E+01 (+)
F8	Mean	<b>1.16E+01</b>	2.48E+01	3.39E+01	1.67E+01	2.17E+01	7.13E+01	1.59E+01	1.51E+01	2.83E+01
	Std	<b>2.46E+00</b>	1.70E+01 (+)	8.37E+00 (+)	4.36E+00 (+)	7.87E+00 (+)	2.40E+01 (+)	5.13E+00 (+)	3.91E+00 (+)	1.40E+01 (+)
F9	Mean	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Std	0.00E+00	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)
F10	Mean	<b>4.18E+03</b>	4.81E+03	5.89E+03	5.40E+03	5.67E+03	6.47E+03	5.60E+03	5.48E+03	5.53E+03
	Std	1.57E+03	1.43E+03 (=)	7.53E+02 (+)	1.35E+03 (=)	<b>5.68E+02</b> (+)	1.46E+03 (+)	1.43E+03 (+)	1.33E+03 (+)	8.22E+02 (+)
F11	Mean	3.07E+01	2.97E+01	3.19E+01	3.09E+01	3.32E+01	3.94E+01	2.97E+01	<b>2.77E+01</b>	3.59E+01
	Std	4.69E+00	<b>4.07E+00</b> (=)	5.20E+00 (=)	4.09E+00 (=)	5.39E+00 (=)	1.08E+01 (+)	4.71E+00 (=)	4.15E+00 (-)	6.25E+00 (+)
F12	Mean	1.81E+03	<b>1.50E+03</b>	2.64E+03	1.94E+03	2.60E+03	2.57E+03	2.08E+03	1.99E+03	2.05E+03
	Std	5.07E+02	<b>3.49E+02</b> (-)	5.51E+02 (+)	4.72E+02 (=)	6.99E+02 (+)	8.75E+02 (+)	5.05E+02 (=)	5.14E+02 (=)	5.38E+02 (=)
F13	Mean	4.42E+01	5.43E+01	6.73E+01	4.81E+01	9.30E+01	7.04E+01	4.49E+01	3.94E+01	5.22E+01
	Std	<b>1.94E+01</b>	3.63E+01 (+)	3.06E+01 (+)	2.84E+01 (+)	3.23E+01 (+)	2.79E+01 (+)	2.39E+01 (+)	2.08E+01 (=)	2.49E+01 (+)
F14	Mean	2.74E+01	2.77E+01	2.94E+01	<b>2.53E+01</b>	2.93E+01	4.54E+01	2.57E+01	2.57E+01	3.16E+01
	Std	2.11E+00	2.54E+00 (=)	3.42E+00 (+)	<b>1.79E+00</b> (-)	3.11E+00 (+)	7.98E+00 (+)	1.96E+00 (-)	2.56E+00 (-)	6.13E+00 (+)
F15	Mean	2.77E+01	2.57E+01	3.56E+01	2.72E+01	3.86E+01	3.36E+01	2.61E+01	2.59E+01	3.21E+01
	Std	3.61E+00	<b>2.81E+00</b> (=)	7.97E+00 (+)	4.24E+00 (=)	9.89E+00 (+)	5.99E+00 (+)	3.74E+00 (=)	4.80E+00 (=)	6.59E+00 (+)
F16	Mean	<b>3.55E+02</b>	6.42E+02	9.32E+02	6.02E+02	6.62E+02	1.33E+03	5.60E+02	6.17E+02	8.92E+02
	Std	1.40E+02	1.91E+02 (+)	3.77E+02 (+)	1.99E+02 (+)	1.50E+02 (+)	4.28E+02 (+)	1.48E+02 (+)	<b>1.27E+02</b> (+)	3.63E+02 (+)
F17	Mean	<b>2.74E+02</b>	3.67E+02	8.09E+02	4.50E+02	5.27E+02	1.07E+03	4.50E+02	4.74E+02	5.97E+02
	Std	1.12E+02	<b>1.04E+02</b> (+)	2.94E+02 (+)	1.63E+02 (+)	1.41E+02 (+)	4.06E+02 (+)	1.34E+02 (+)	1.24E+02 (+)	2.51E+02 (+)
F18	Mean	2.77E+01	<b>2.59E+01</b>	3.91E+01	2.69E+01	4.07E+01	7.90E+04	2.78E+01	2.75E+01	3.26E+01
	Std	5.71E+00	<b>2.14E+00</b> (-)	6.86E+00 (+)	2.76E+00 (=)	7.76E+00 (+)	4.33E+05 (+)	3.62E+00 (=)	2.66E+00 (=)	5.66E+00 (+)
F19	Mean	1.94E+01	<b>1.59E+01</b>	1.74E+01	1.80E+01	2.32E+01	2.36E+01	1.81E+01	1.80E+01	2.02E+01
	Std	3.21E+00	2.51E+00 (-)	<b>2.40E+00</b> (-)	3.59E+00 (=)	3.17E+00 (+)	6.42E+00 (+)	3.34E+00 (=)	3.49E+00 (=)	3.65E+00 (=)

Table 15 continued

	jSO-SPOBL	jSO	jSO-OBL	jSO-QOBL	jSO-QROBL	jSO-GOBL	jSO-COOBL	jSO-EOBL	jSO-REOBL	jSO-COBL
F20	Mean <b>1.26E+02</b>	2.79E+02	6.67E+02	2.06E+02	2.05E+02	6.22E+02	9.78E+02	2.50E+02	2.29E+02	3.76E+02
	Std <b>7.52E+01</b>	1.08E+02 (+)	3.01E+02 (+)	1.10E+02 (=)	8.86E+01 (+)	2.62E+02 (+)	3.46E+02 (+)	1.40E+02 (+)	1.08E+02 (+)	1.93E+02 (+)
F21	Mean <b>2.12E+02</b>	2.21E+02	2.32E+02	2.18E+02	2.18E+02	2.20E+02	2.81E+02	2.18E+02	2.19E+02	2.35E+02
	Std <b>2.41E+00</b>	3.19E+00 (+)	1.17E+01 (+)	4.77E+00 (+)	5.91E+00 (+)	4.48E+00 (+)	2.82E+01 (+)	4.04E+00 (+)	5.35E+00 (+)	1.54E+01 (+)
F22	Mean 4.42E+03	1.98E+03	3.07E+03	<b>1.00E+02</b>	1.00E+02	1.00E+02	6.97E+03	5.39E+03	9.66E+02	5.58E+03
	Std 2.03E+03	2.42E+03 (-)	3.48E+03 (=)	2.12E-13 (-)	<b>1.15E-13</b> (-)	2.76E-13 (-)	1.54E+03 (+)	2.05E+03 (+)	2.09E+03 (-)	2.06E+03 (+)
F23	Mean 4.40E+02	<b>4.33E+02</b>	4.60E+02	4.47E+02	4.49E+02	4.52E+02	5.21E+02	4.49E+02	4.46E+02	4.68E+02
	Std 9.39E+00	8.72E+00 (-)	1.19E+01 (+)	9.41E+00 (+)	<b>7.24E+00</b> (+)	7.82E+00 (+)	2.77E+01 (+)	1.03E+01 (+)	7.86E+00 (+)	1.84E+01 (+)
F24	Mean 5.13E+02	<b>5.06E+02</b>	5.26E+02	5.17E+02	5.16E+02	5.18E+02	5.88E+02	5.16E+02	5.17E+02	5.36E+02
	Std 6.76E+00	5.22E+00 (-)	1.39E+01 (+)	4.17E+00 (+)	4.14E+00 (+)	6.70E+00 (+)	2.75E+01 (+)	<b>3.48E+00</b> (+)	6.02E+00 (+)	1.63E+01 (+)
F25	Mean 4.80E+02	4.92E+02	4.81E+02	4.80E+02	4.80E+02	4.83E+02	<b>4.79E+02</b>	4.82E+02	4.80E+02	4.81E+02
	Std 2.28E+00	1.88E+01 (+)	3.01E+00 (=)	5.57E-02 (=)	1.62E-02 (=)	4.74E+00 (+)	2.99E+00 (-)	4.00E+00 (=)	<b>1.38E-02</b> (=)	3.61E+00 (=)
F26	Mean 1.21E+03	<b>1.11E+03</b>	1.38E+03	1.27E+03	1.29E+03	1.30E+03	1.90E+03	1.27E+03	1.26E+03	1.42E+03
	Std 1.01E+02	6.77E+01 (-)	9.89E+01 (+)	<b>5.70E+01</b> (+)	7.87E+01 (+)	9.57E+01 (+)	2.52E+02 (+)	8.94E+01 (=)	6.19E+01 (=)	1.64E+02 (+)
F27	Mean 5.16E+02	5.29E+02	5.30E+02	5.18E+02	5.19E+02	5.31E+02	<b>5.02E+02</b>	5.24E+02	5.21E+02	5.15E+02
	Std 1.45E+01	1.93E+01 (+)	2.00E+01 (+)	1.02E+01 (=)	1.08E+01 (=)	2.25E+01 (+)	<b>9.31E+00</b> (-)	1.92E+01 (=)	1.15E+01 (=)	1.83E+01 (=)
F28	Mean 4.59E+02	4.59E+02	4.60E+02	4.59E+02	4.59E+02	4.59E+02	4.66E+02	<b>4.59E+02</b>	<b>4.59E+02</b>	4.59E+02
	Std 8.92E+00	3.82E-13 (-)	8.92E+00 (+)	<b>2.80E-13</b> (-)	3.73E-13 (=)	3.55E-13 (-)	2.04E+01 (=)	3.92E-13 (=)	4.26E-13 (=)	3.15E-13 (=)
F29	Mean <b>4.11E+02</b>	4.84E+02	4.83E+02	4.22E+02	4.17E+02	5.03E+02	7.20E+02	4.24E+02	4.18E+02	4.67E+02
	Std 3.21E+01	4.50E+01 (+)	6.27E+01 (+)	3.89E+01 (=)	<b>3.16E+01</b> (=)	8.06E+01 (+)	2.48E+02 (+)	4.39E+01 (=)	3.84E+01 (=)	9.67E+01 (+)
F30	Mean 5.99E+05	5.96E+05	6.50E+05	6.25E+05	6.08E+05	6.24E+05	<b>4.39E+04</b>	6.36E+05	6.20E+05	4.39E+05
	Std 3.88E+04	<b>2.29E+04</b> (=)	6.08E+04 (+)	4.97E+04 (=)	3.43E+04 (=)	5.20E+04 (=)	1.16E+05 (-)	6.45E+04 (=)	5.26E+04 (=)	2.81E+05 (=)
+/-		12/9/8	22/6/1	12/14/3	11/15/3	24/3/2	22/4/3	13/15/1	11/15/3	19/10/0

Bold values represent the best results in each row

**Table 16** Friedman test with Tukey–Kramer’s post hoc for jSO with OBL variants on CEC 2017 at 50-D

	Algorithm	Average ranking	<i>z</i> value	<i>p</i> value	Adj. <i>p</i> value (Tukey–Kramer)	Sig.	Test statistics
1	jSO-SPOBL	<b>2.95</b>					
2	jSO	4.53	-2.59E+00	1.39E-02	5.27E-01	No	
3	jSO-OBL	7.75	-7.85E+00	1.60E-14	1.35E-07	Yes	N 29
4	jSO-QOBL	4.38	-2.35E+00	2.55E-02	6.68E-01	No	Chi-Square 127.85
5	jSO-QROBL	3.28	-5.45E-01	3.44E-01	1.00E+00	No	df 9
6	jSO-GOBL	7.87	-8.05E+00	3.51E-15	1.30E-07	Yes	<i>p</i> value 3.27E-23
7	jSO-COOBL	8.48	-9.05E+00	6.28E-19	1.27E-07	Yes	Sig. Yes
8	jSO-EOBL	4.98	-3.33E+00	1.57E-03	1.75E-01	No	
9	jSO-REOBL	4.18	-2.02E+00	5.21E-02	8.31E-01	No	
10	jSO-COBL	6.58	-5.95E+00	8.42E-09	6.30E-05	Yes	

**Table 17** Result of planetary gear train design optimization problem with SASS-SPOBL, SASS, sCMAgES and COLSHADE

	RC22	SASS-SPOBL	SASS [50]	sCMAgES [51]	COLSHADE [52]
Best_f		0.525769	0.525769	0.525967	0.525769
Median_f		<b>0.527347</b>	0.645573	0.53	0.53
Mean_f		<b>0.529216</b>	1.001524	0.530809	0.541026
Worst_f		<b>0.536523</b>	3.521656	0.543846	0.746667
Std_f		<b>0.003361</b>	0.725184	0.004348	0.043451

Bold values represent the best results in each row

**Table 18** Result of four-stage gear box problem with SASS-SPOBL, SASS, sCMAgES and COLSHADE

	RC26	SASS-SPOBL	SASS [50]	sCMAgES [51]	COLSHADE [52]
Best_f		<b>35.35923</b>	36.2504	36.24853	<b>35.35923</b>
Median_f		36.54604	38.1297	48.61283	<b>36.24929</b>
Mean_f		37.35082	38.5141	53.71013	<b>36.61098</b>
Worst_f		<b>40.08065</b>	45.50641	120.3174	40.93115
Std_f		<b>1.380578</b>	2.114822	17.8764	1.395912

Bold values represent the best results in each row

**Table 19** Result of SOPWM for 13-level Inverters with SASS-SPOBL, SASS, sCMAgES and COLSHADE

	RC50	SASS-SPOBL	SASS [50]	sCMAgES [51]	COLSHADE [52]
Best_f		0.015857	<b>0.015115</b>	0.017985	0.02045
Median_f		<b>0.019572</b>	0.019581	0.080702	0.039494
Mean_f		<b>0.021683</b>	0.023637	0.083849	0.065091
Worst_f		<b>0.034203</b>	0.061767	0.150263	0.238732
Std_f		<b>0.005484</b>	0.010305	0.026915	0.049192

Bold values represent the best results in each row

of fitness evaluations, we presented a subpopulation-based opposition instead of population-based opposition. Also, we applied a self-adaptive parameter control strategy with generalized Lehmer mean in SPOBL to balance between exploration and exploitation during the evolutionary process.

The performance of the proposed algorithm is verified on the CEC 2017 and CEC 2020 test suites. Eight state-of-the-art OBL variants are compared in experiments to illustrate the effectiveness of the proposed algorithm. The results confirm that SPOBL has a significant improvement compared with other OBL variants, but also show that SPOBL can be applied

to any advanced DE variant to improve the corresponding algorithm’s performance.

Future research directions: (1) apply SPOBL to multi-objective EAs, (2) Apply SPOBL to other metaheuristic algorithms and compare their performance, (3) analyze the convergence behavior of subpopulation-based opposition and (4) design a novel OBL variant which aims to balance between exploration and exploitation.

**Acknowledgements** The work was supported by the National Natural Science Foundation of China (Nos. 11961001 and 61561001),

China. The Construction Project of First-class Subjects in Ningxia Higher Education (No. NXYLXK2017B09), China. The Major Proprietary Funded Project of North Minzu University (No. ZDZX201901), China. The Graduate Innovation Project of North Minzu University (No. YCX21152), China.

## Declarations

**Conflict of interest** We declare that we have no financial and personal relationships with other people or organizations that can inappropriately influence our work, there is no professional or other personal interest of any nature or kind in any product, service and/or company that could be construed as influencing the position presented in, or the review of, the manuscript entitled, “Self-adaptive opposition-based differential evolution with subpopulation strategy for numerical and engineering optimization problems”.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Cheng R, He C, Jin Y, Yao X (2018) Model-based evolutionary algorithms: a short survey. *Complex Intell Syst* 4(4):283–292. <https://doi.org/10.1007/s40747-018-0080-1>
- Huang C, Li Y, Yao X (2020) A survey of automatic parameter tuning methods for metaheuristics. *IEEE Trans Evolut Comput* 24(2):201–216. <https://doi.org/10.1109/tevc.2019.2921598>
- Osaba E, Villar-Rodríguez E, Ser JD, Nebro AJ, Molina D, La Torre A, Suganthan PN, Coello CAC, Herrera F (2021) A tutorial on the design, experimentation and application of metaheuristic algorithms to real-world optimization problems. *Swarm Evolut Comput* 64:100888. <https://doi.org/10.1016/j.swevo.2021.100888>
- Wang L, Gao Y, Gao S, Yong X (2021) A new feature selection method based on a self-variant genetic algorithm applied to android malware detection. *Symmetry* 13(7):1290. <https://doi.org/10.3390/sym13071290>
- Xu J, Jin Y, Du W (2021) A federated data-driven evolutionary algorithm for expensive multi-/many-objective optimization. *Complex Intell Syst*. <https://doi.org/10.1007/s40747-021-00506-7>
- Ser JD, Osaba E, Molina D, Yang XS, Salcedo-Sanz S, Camacho D, Das S, Suganthan PN, Coello CAC, Herrera F (2019) Bio-inspired computation: where we stand and what’s next. *Swarm Evolut Comput* 48:220–250. <https://doi.org/10.1016/j.swevo.2019.04.008>
- Storn R, Price K (1995) Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces. *Tech. Rep. TR-95-012*, International Computer Science, Berkeley, California
- Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *J Global Optim* 11(4):341–359. <https://doi.org/10.1023/a:1008202821328>
- Das S, Suganthan PN (2011) Differential evolution: a survey of the state-of-the-art. *IEEE Trans Evolut Comput* 15(1):4–31. <https://doi.org/10.1109/tevc.2010.2059031>
- Das S, Mullick SS, Suganthan P (2016) Recent advances in differential evolution—an updated survey. *Swarm Evolut Comput* 27:1–30. <https://doi.org/10.1016/j.swevo.2016.01.004>
- Zhang Y, wei Gong D, zhi Gao X, Tian T, yan Sun X (2020) Binary differential evolution with self-learning for multi-objective feature selection. *Inform Sci* 507:67–85. <https://doi.org/10.1016/j.ins.2019.08.040>
- Alswaitti M, Albughdadi M, Isa NAM (2019) Variance-based differential evolution algorithm with an optional crossover for data clustering. *Appl Soft Comput* 80:1–17. <https://doi.org/10.1016/j.asoc.2019.03.013>
- Aladeemy M, Adwan L, Booth A, Khasawneh MT, Poranki S (2020) New feature selection methods based on opposition-based learning and self-adaptive cohort intelligence for predicting patient no-shows. *Appl Soft Comput*. <https://doi.org/10.1016/j.asoc.2019.105866>
- Hussain A, Muhammad YS (2019) Trade-off between exploration and exploitation with genetic algorithm using a novel selection operator. *Complex Intell Syst* 6(1):1–14. <https://doi.org/10.1007/s40747-019-0102-7>
- Rahnamayan S, Tizhoosh HR, Salama MMA (2008) Opposition-based differential evolution. *IEEE Trans Evolut Comput* 12(1):64–79. <https://doi.org/10.1109/tevc.2007.894200>
- Tizhoosh H (2005) Opposition-based learning: a new scheme for machine intelligence. In: *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC’06)*. IEEE. <https://doi.org/10.1109/cimca.2005.1631345>
- Ergezer M, Simon D (2014) Mathematical and experimental analyses of oppositional algorithms. *IEEE Trans Cybern* 44(11):2178–2189. <https://doi.org/10.1109/tcyb.2014.2303117>
- Ergezer M, Simon D, Du D (2009) Oppositional biogeography-based optimization. In: *2009 IEEE International Conference on Systems, Man and Cybernetics*. IEEE. <https://doi.org/10.1109/icsmc.2009.5346043>
- Rahnamayan S, Wang GG, Ventresca M (2012) An intuitive distance-based explanation of opposition-based sampling. *Appl Soft Comput* 12(9):2828–2839. <https://doi.org/10.1016/j.asoc.2012.03.034>
- Al-Qunaiier FS, Tizhoosh HR, Rahnamayan S (2010) Opposition based computing—a survey. In: *The 2010 International Joint Conference on Neural Networks (IJCNN)*. IEEE. <https://doi.org/10.1109/ijcnn.2010.5596906>
- Xu Q, Wang L, Wang N, Hei X, Zhao L (2014) A review of opposition-based learning from 2005 to 2012. *Eng Appl Artif Intell* 29:1–12. <https://doi.org/10.1016/j.engappai.2013.12.004>
- Mahdavi S, Rahnamayan S, Deb K (2018) Opposition based learning: a literature review. *Swarm Evolut Comput* 39:1–23. <https://doi.org/10.1016/j.swevo.2017.09.010>
- Esmailzadeh A, Rahnamayan S (2011) Opposition-based differential evolution with protective generation jumping. In: *2011 IEEE Symposium on Differential Evolution (SDE)*. IEEE. <https://doi.org/10.1109/sde.2011.5952059>
- Rahnamayan S, Tizhoosh HR, Salama MM (2007) Quasi-oppositional differential evolution. In: *2007 IEEE Congress on Evolutionary Computation*. IEEE. <https://doi.org/10.1109/cec.2007.4424748>
- Wang H, Wu Z, Rahnamayan S (2010) Enhanced opposition-based differential evolution for solving high-dimensional continuous

- optimization problems. *Soft Comput* 15(11):2127–2140. <https://doi.org/10.1007/s00500-010-0642-7>
26. Xu, Wang L, He B, Wang N (2011) Modified opposition-based differential evolution for function optimization. *J Comput Inform Syst* 7(5):1582–1591
  27. Seif Z, Ahmadi M (2015) An opposition-based algorithm for function optimization. *Eng Appl Artif Intell* 37:293–306. <https://doi.org/10.1016/j.engappai.2014.09.009>
  28. Rahnamayan S, Jesuthasan J, Bourennani F, Salehinejad H, Naterer GF (2014) Computing opposition by involving entire population. In: 2014 IEEE Congress on Evolutionary Computation (CEC). IEEE. <https://doi.org/10.1109/cec.2014.6900329>
  29. Alkayem NF, Shen L, Asteris PG, Sokol M, Xin Z, Cao M (2021) A new self-adaptive quasi-oppositional stochastic fractal search for the inverse problem of structural damage assessment. *Alexand Eng J*. <https://doi.org/10.1016/j.aej.2021.06.094>
  30. Xu Y, Yang X, Yang Z, Li X, Wang P, Ding R, Liu W (2021) An enhanced differential evolution algorithm with a new oppositional-mutual learning strategy. *Neurocomputing* 435:162–175. <https://doi.org/10.1016/j.neucom.2021.01.003>
  31. Choi TJ, Togelius J, Cheong YG (2021) A fast and efficient stochastic opposition-based learning for differential evolution in numerical optimization. *Swarm Evolut Computat*. <https://doi.org/10.1016/j.swevo.2020.100768>
  32. Kumar A, Wu G, Ali MZ, Mallipeddi R, Suganthan PN, Das S (2020) A test-suite of non-convex constrained optimization problems from the real-world and some baseline results. *Swarm Evolut Comput*. <https://doi.org/10.1016/j.swevo.2020.100693>
  33. Awad N, Ali M, JLBQ, Suganthan P (2016) Problem definitions and evaluation criteria for the cec 2017 special session and competition on single objective bound constrained real-parameter numerical optimization, technical report, ntu, singapore. Tech. rep
  34. 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. <https://doi.org/10.1016/j.cor.2015.09.006>
  35. Zhang J, Sanderson A (2009) JADE: adaptive differential evolution with optional external archive. *IEEE Trans Evol Computat* 13(5):945–958. <https://doi.org/10.1109/tevc.2009.2014613>
  36. Wang H, Wu Z, Rahnamayan S, Wang J (2010) Diversity analysis of opposition-based differential evolution—an experimental study. In: *Advances in Computation and Intelligence*, pp. 95–102. Springer Berlin Heidelberg. [https://doi.org/10.1007/978-3-642-16493-4\\_10](https://doi.org/10.1007/978-3-642-16493-4_10)
  37. Wang W, Wang H, Sun H, Rahnamayan S (2016) Using opposition-based learning to enhance differential evolution: A comparative study. In: 2016 IEEE Congress on Evolutionary Computation (CEC). IEEE. <https://doi.org/10.1109/cec.2016.7743780>
  38. Derrac J, García 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. <https://doi.org/10.1016/j.swevo.2011.02.002>
  39. García S, Fernández 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. *Inform Sci* 180(10):2044–2064. <https://doi.org/10.1016/j.ins.2009.12.010>
  40. 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. <https://doi.org/10.1109/cec.2014.6900380>
  41. Brest J, Maucec MS, Boskovic B (2017) Single objective real-parameter optimization: Algorithm jSO. In: 2017 IEEE Congress on Evolutionary Computation (CEC). IEEE. <https://doi.org/10.1109/cec.2017.7969456>
  42. Tanabe R, Fukunaga A (2013) Success-history based parameter adaptation for differential evolution. In: 2013 IEEE Congress on Evolutionary Computation. IEEE. <https://doi.org/10.1109/cec.2013.6557555>
  43. Guo SM, Tsai JSH, Yang CC, Hsu PH (2015) A self-optimization approach for l-SHADE incorporated with eigenvector-based crossover and successful-parent-selecting framework on CEC 2015 benchmark set. In: 2015 IEEE Congress on Evolutionary Computation (CEC). IEEE. <https://doi.org/10.1109/cec.2015.7256999>
  44. Sallam KM, Sarker RA, Essam DL, Elsayed SM (2015) Neurodynamic differential evolution algorithm and solving CEC2015 competition problems. In: 2015 IEEE Congress on Evolutionary Computation (CEC). IEEE. <https://doi.org/10.1109/cec.2015.7257003>
  45. Brest J, Maucec MS, Boskovic B (2016) iL-SHADE: Improved l-SHADE algorithm for single objective real-parameter optimization. In: 2016 IEEE Congress on Evolutionary Computation (CEC). IEEE. <https://doi.org/10.1109/cec.2016.7743922>
  46. Awad NH, Ali MZ, Suganthan PN, Reynolds RG (2016) An ensemble sinusoidal parameter adaptation incorporated with l-SHADE for solving CEC2014 benchmark problems. In: 2016 IEEE Congress on Evolutionary Computation (CEC). IEEE. <https://doi.org/10.1109/cec.2016.7744163>
  47. Awad NH, Ali MZ, Suganthan PN, Reynolds RG, Shatnawi AM (2017) A novel differential crossover strategy based on covariance matrix learning with euclidean neighborhood for solving real-world problems. In: 2017 IEEE Congress on Evolutionary Computation (CEC). IEEE. <https://doi.org/10.1109/cec.2017.7969337>
  48. Awad NH, Ali MZ, Suganthan PN (2017) Ensemble sinusoidal differential covariance matrix adaptation with euclidean neighborhood for solving CEC2017 benchmark problems. In: 2017 IEEE Congress on Evolutionary Computation (CEC). IEEE. <https://doi.org/10.1109/cec.2017.7969336>
  49. 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. <https://doi.org/10.1109/cec.2018.8477977>
  50. Kumar A, Das S, Zelinka I (2020) A self-adaptive spherical search algorithm for real-world constrained optimization problems. In: *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*. ACM. <https://doi.org/10.1145/3377929.3398186>
  51. Kumar A, Das S, Zelinka I (2020) A modified covariance matrix adaptation evolution strategy for real-world constrained optimization problems. In: *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*. ACM. <https://doi.org/10.1145/3377929.3398185>
  52. Gurrola-Ramos J, Hernandez-Aguirre A, Dalmau-Cedeno O (2020) COLSHADE for real-world single-objective constrained optimization problems. In: 2020 IEEE Congress on Evolutionary Computation (CEC). IEEE. <https://doi.org/10.1109/cec48606.2020.9185583>
  53. Dukic M, Dobrosavljevic Z (1990) A method of a spread-spectrum radar polyphase code design. *IEEE J Select Areas Commun* 8(5):743–749. <https://doi.org/10.1109/49.56381>