Artificial Intelligence Review (2022) 55:3937–3978
https://doi.org/10.1007/s10462-021-10099-9

SFSADE: an improved self-adaptive differential evolution algorithm with a shuffled frog-leaping strategy

Qingtao Pan[1] · Jun Tang[1] · Haoran Wang[1] · Hao Li[1] · Xi Chen[1] · Songyang Lao[1]

Published online: 17 November 2021

Abstract
The differential evolution (DE) algorithm is an efficient random search algorithm based on swarm intelligence for solving optimization problems. It has the advantages of easy implementation, fast convergence, strong optimization ability and good robustness. However, the performance of DE is very sensitive to the design of different operators and the setting of control parameters. To solve these key problems, this paper proposes an improved self-adaptive differential evolution algorithm with a shuffled frog-leaping strategy (SFSADE). It innovatively incorporates the idea of the shuffled frog-leaping algorithm into DE, and at the same time, it cleverly introduces a new strategy of classification mutation, and also designs a new adaptive adjustment mechanism for control parameters. In addition, we have carried out a large number of simulation experiments on the 25 benchmark functions of CEC 2005 and two nonparametric statistical tests to comprehensively evaluate the performance of SFSADE. Finally, the results of simulation experiments and nonparametric statistical tests show that SFSADE is very effective in improving DE, and significantly improves the overall diversity of the population in the process of dynamic evolution. Compared with other advanced DE variants, its global search speed and optimization performance also has strong competitiveness.

Keywords Differential evolution · Numerical optimization · Self-adaptive mechanism · Shuffled frog leaping

1 Introduction

Differential evolution (DE) was jointly proposed by (Storn and Price 1997) to solve Chebyshev polynomials. It is a simple and efficient evolutionary algorithm for solving global optimization problems in a continuous search space (Li and Wang 2020). Essentially, DE is a random search algorithm based on a population, which mainly simulates the biological evolution process through the three operators of mutation, crossover and selection to evolve the initial solution to the global optimal solution (Wang

✉ Jun Tang
tangjun06@nudt.edu.cn

[1] College of Systems Engineering, National University of Defense Technology, Hunan 410073, China

et al. 2011). In the past 20 years of development, DE has attracted increasing attention from researchers and has been successfully applied to various scientific and engineering fields, such as pattern recognition (Maulik and Saha 2009; Koloseni et al. 2012; Cai and Wang 2015), image processing (Das and Konar 2009; Su et al. 2012), collision avoidance treatment (Tang et al. 2016; Sun et al. 2017; Tang 2019), and engineering design (Guo et al. 2001; Zhao et al. 2014; Sakr et al. 2017), and has achieved good application effects.

When DE is used to solve specific optimization problems, the design of different operators (i.e., mutation, crossover and selection) and the adjustment of control parameters (i.e., population size NP, mutation factor F and crossover probability CR) are very important (Parouha and Verma 2021). They directly determine the result of DE solution to a large extent. For different optimization problems, operator design and parameter setting are often different (Wu et al. 2016). However, improper settings may lead to premature or slow convergence of algorithm, easily falling into local optimization and other many problems. Therefore, researchers usually need to try the above settings repeatedly to better apply DE to solve numerical optimization problems (Zhu et al. 2020).

Therefore, in order to solve the above problem, we summarize a large number of enhanced DE variants proposed by researchers in recent years, such as FADE (with fuzzy logic self-adaptive strategy) (Liu and Lampinen 2005), TDE (with trigonometric mutation operation) (Fan and Lampinen 2003), EDA (with distributed estimation strategy) (Sun et al. 2005), jDE (with self-adaptive parameters) (Brest et al. 2006), AnDE (with new mutation and selection operation and combining simulated annealing ideal) (Das et al. 2007), JADE (with "current-to-pbest/1" mutation operation and self-adaptive parameters) (Zhang and Sanderson 2009), SaDE (with self-adaptive mutation strategies and parameters) (Qin et al. 2009b), CoDE (with composite generation strategies and control parameters) (Wang et al. 2011), SspDE (with self-adaptive strategies and control parameters) (Pan et al. 2011), ESADE (with "current-to-pbest/1" mutation operation and self-adaptive control parameters) (Guo et al. 2014), DMPSADE (with self-adaptive discrete mutation control parameters) (Fan and Yan 2015), MPEDE (with multiple mutation strategies and self-adaptive parameters) (Wu et al. 2016), DEMPSO (combining particle swarm optimization ideal) (Mao et al. 2017), SpDE (with multiple subpopulations and phase-mutations strategy) (Pan et al. 2018), HyGADE (combining Genetic Algorithm ideal) (Chaudhary et al. 2019), and SAMDE (with self-adaptive multipopulation strategy) (Zhu et al. 2020). Through much experimental research and theoretical analysis, the efficiency and performance of DE variants in solving problems have been greatly improved. In summary, the above improvements to DE can be simply divided into the following three types: redesigning the operations, adaptively adjusting control parameters, and incorporating other intelligent optimization algorithms (Li et al. 2020a). As a very successful algorithm, SAMDE (Zhu et al. 2020), integrates three effective ways of improvement. In order to increase the diversity of the population and improve the optimization performance and speed of the algorithm, this is a very effective attempt in which the entire population is decomposed into multiple subpopulations in DE. These subpopulations are continuously decomposed and merged in the evolution process, and different mutation strategies and control parameter adaptive adjustment mechanisms are used (MA et al. 2009). Although some effective rules on those settings have been summarized based on a large number of studies (Das and Suganthan 2011), their universality and adaptability need to be further improved and it is still very difficult to design a DE algorithm with better comprehensive performance.

In this paper, to further improve the solution performance of the DE algorithm, accelerate the convergence speed, prevent falling into local optimization and improve the stability, on the basis of SAMDE (Zhu et al. 2020) and p-ADE (Bi and Xiao 2012), we propose an improved self-adaptive differential evolution algorithm with a shuffled frog-leaping strategy, called SFSADE. We introduce the mutation strategy based on the classification idea and design a new parameter adaptive adjustment mechanism. The most important aspect is to form multiple subpopulations that evolve independently by using the idea of shuffled frog-leaping. The specific improvement measures include the following four aspects:

(1) Introducing the shuffled frog-leaping strategy, in which dividing meme groups are based on individual fitness values, thereby forming multiple subgroups to evolve independently, improving the diversity and convergence speed of the population;
(2) Designing a new mutation strategy. At the same time, the global optimal solution in the evolution process and the local optimal solution of each meme group are used to avoid search blindness caused by the random selection of individuals in the difference vector;
(3) Using classification mutation strategy. Choosing different evolution directions for individuals with different fitness values to further balance the "explore" and "exploit" capability of the algorithm;
(4) Incorporating a new parameter self-adaptive strategy. According to the fitness value of the individual and the evolutionary number, an adaptive parameter scheme is provided for each individual. In addition, the SFSADE algorithm proposed in this paper have carries out 10-, 30- and 50-dimensional tests on the 25 benchmark functions of CEC 2005 (Suganthan et al. 2005) and also performed two nonparametric statistical test, Wilcoxon signed rank test and Friedman test. Comparing its performance with the results of 7 other advanced algorithms at home and abroad mentioned in reference (Zhu et al. 2020), SFSADE has the great advantage and competitiveness.

The remaining sections of this paper are organized as follows: Sect. 2 introduces the basic DE and Shuffled Frog-Leaping algorithm. A review of the related work of differential evolution algorithms is presented in Sect. 3. In Sect. 4, the proposed algorithm SFSADE is introduced in detail. Section 5 shows the relevant experimental results and analysis. Finally, the conclusions are drawn in Sect. 6.

2 Differential evolution and shuffled frog leaping

2.1 Differential evolution

DE is a heuristic random parallel search algorithm based on swarm intelligence, which is mainly used to solve global optimization problems with continuous variables. Similar to other evolutionary algorithms, it mainly includes mutation, crossover and selection operations and performs the evolutionary search process through continuous iterative calculation. The five key operations of DE are introduced as follows.

2.1.1 Initialization operation

Assume that constrained numerical optimization problems (CNOPs) are defined as follows (Parouha and Verma 2021).

$$Min\, f(X) \quad X \subseteq \left[X_{\min}, X_{\max}\right],\ X = \left(X_1, X_2, \ldots, X_n\right) \tag{1}$$

In the initial evolution number $t = 0$, the individuals $\boldsymbol{X_{i,t}} = \left(x_{i,t}^1,\ x_{i,t}^2, \ldots,\ x_{i,t}^j,\ \ldots, x_{i,t}^D\right)$ in the initial population of DE are generated by Eq. (2), which is randomly generated according to the lower bound $\boldsymbol{X_{min}} = \left\{x_{min}^1, x_{min}^2, \ldots, x_{min}^n\right\}$ and upper bound $\boldsymbol{X_{max}} = \left\{x_{max}^1, x_{max}^2, \ldots, x_{max}^n\right\}$ of the optimization problem and is evenly distributed in the feasible solution space (Das and Suganthan 2011). The dimensionality D of the individual variables is equal to the dimension n of the decision variable $\boldsymbol{X}$ in the objective function, and the number of individuals in the population is NP (Fan et al. 2019).

$$x_{i,t}^j = x_{min}^j + \mathrm{rand}(0,1)\left(x_{max}^j - x_{min}^j\right) j = 1, 2, \ldots, D \quad i = 1, 2, \ldots, NP \tag{2}$$

where $x_{i,t}^j$ is the value of the j^{th} dimension of the i^{th} individual in the t^{th} generation, and rand(0, 1) represents a random variable uniformly distributed in the range [0, 1].

2.1.2 Mutation operation

In the current t^{th} evolutionary number, the i^{th} individual $\boldsymbol{x_{i,t}}$ in the parent population is also called the target individual vector, and its corresponding mutant individual vector $\boldsymbol{V_{i,t+1}} = \left(v_{i,t+1}^1,\ v_{i,t+1}^2, \ldots, v_{i,t+1}^D\right)$ is generated by the mutation operator. The most basic mutation strategy "DE/rand/1" is shown in Eq. (3), which is widely used, simple and robust (Zhang and Duan 2015).

$$\boldsymbol{V_{i,t+1}} = \boldsymbol{X_{r1,t}} + F \cdot \left(\boldsymbol{X_{r2,t}} - \boldsymbol{X_{r3,t}}\right) \tag{3}$$

where $r_1,\ r_2, r_3 \in \{1, 2, \ldots, NP\}$ are randomly selected positive integers that are different from each other, representing the indexes of three randomly selected different individuals in the parent population, and r_1, r_2, r_3 are different from the current target individual vector index$i$. $\boldsymbol{X_{ri,t}}$ is called the basis vector. $\boldsymbol{X_{r2,t}} - \boldsymbol{X_{r3,t}}$ are called the difference vector. F is the mutation factor or scaling factor, and it is one of the main control parameters of the DE algorithm. It usually takes a value in the range[0, 2], which controls the scaling of the difference vector and determines the degree of influence on the basis vector.

Some well-known mutation strategies in the literature are summarized as follows (Zhu et al. 2020):

"DE/rand/2"

$$\boldsymbol{V_{i,t+1}} = \boldsymbol{X_{r1,t}} + F \cdot \left(\boldsymbol{X_{r2,t}} - \boldsymbol{X_{r3,t}}\right) + F \cdot \left(\boldsymbol{X_{r4,t}} - \boldsymbol{X_{r5,t}}\right) \tag{4}$$

"DE/best/1"

$$\boldsymbol{V_{i,t+1}} = \boldsymbol{X_{best,t}} + F \cdot \left(\boldsymbol{X_{r1,t}} - \boldsymbol{X_{r2,t}}\right) \tag{5}$$

"DE/best/2"

$$\boldsymbol{V}_{i,t+1} = \boldsymbol{X}_{best,t} + F \cdot \left(\boldsymbol{X}_{r1,t} - \boldsymbol{X}_{r2,t}\right) + F \cdot \left(\boldsymbol{X}_{r3,t} - \boldsymbol{X}_{r4,t}\right) \tag{6}$$

"DE/current-to-best/1"

$$\boldsymbol{V}_{i,t+1} = \boldsymbol{X}_{i,t} + F \cdot \left(\boldsymbol{X}_{best,t} - \boldsymbol{X}_{i,t}\right) + F \cdot \left(\boldsymbol{X}_{r1,t} - \boldsymbol{X}_{r2,t}\right) \tag{7}$$

"DE/rand-to-best/1"

$$\boldsymbol{V}_{i,t+1} = \boldsymbol{X}_{r1,t} + F \cdot \left(\boldsymbol{X}_{best,t} - \boldsymbol{X}_{r1,t}\right) + F \cdot \left(\boldsymbol{X}_{r2,t} - \boldsymbol{X}_{r3,t}\right) \tag{8}$$

where the parameters $r_1,\ r_2,\ r_3,\ r_4,\ r_5$ are positive integers in range $[1, NP]$, which are different from the index i and are not equal to each other. $\boldsymbol{X}_{best,t}$ is the individual with the optimal fitness in the t^{th} generation.

Generally, the mutation individuals of the DE algorithm are generated by adding the weighted difference vector between different individuals in the parent population to the base vector, which is equivalent to adding a random disturbance to the base vector. In the early stage of evolution, there is a large difference between individuals, which makes the difference vector greatly disturb the base vector and ensures a strong exploration ability and global search ability. With the increase in evolutionary number, the difference between individuals decreases, which makes the DE algorithm enhance the population exploitation ability and local search ability.

2.1.3 Detection operation

The out-of-bound detection operator is after the mutation operator, which will ensure that the value of each dimension in every variant individual $\boldsymbol{V}_{i,t}$ is within the boundary constraints $\left[\boldsymbol{X}_{min},\ \boldsymbol{X}_{max}\right]$ of the CNOPs [see Eq. (1)]. If the value of the j^{th} dimension in $\boldsymbol{V}_{i,t}$ exceeds the range, the value can be corrected by the following simple and effective Eq. (9), that is, an individual is randomly generated according to the initialization operation (see Sect. 2.1.1) (Fan et al. 2019).

$$V_{i,t}^{j} = x_{\min}^{j} + rand(0,1) \cdot \left(x_{\max}^{j} - x_{\min}^{j}\right) if \begin{cases} V_{i,t}^{j} < x_{\min}^{j} \\ or \\ V_{i,t}^{j} > x_{\max}^{j} \end{cases} \tag{9}$$

2.1.4 Crossover operation

After the detection operation, to further increase the diversity of the population, the target vector individual $\boldsymbol{X}_{i,t}$ and its corresponding mutant individual $\boldsymbol{V}_{i,t+1}$ need to be crossed to generate trial individual $\boldsymbol{U}_{i,t+1} = \left(u_{i,t+1}^{1},\ u_{i,t+1}^{2}, \ldots, u_{i,t+1}^{D}\right)$. The DE algorithm generally employs the following two crossover methods (Zhu et al. 2020).

2.1.4.1 Binomial crossover In binomial crossover, to make the target individuals $X_{i,t}$ evolve, it is necessary to ensure that the components of at least one dimension of experimental individuals $\boldsymbol{U}_{i,t+1}$ are generated by the mutation individual $\boldsymbol{V}_{i,t+1}$, while the components of other dimensions are determined by the parameter CR randomly.

$$U_{i,t+1}^{j}=\begin{cases} V_{i,t+1}^{j} & \text{if } r_j \le CR\ or\ j=j_{rand} \\ X_{i,t+1}^{j} & otherwise \end{cases} \quad j=1,2,\ldots,D \tag{10}$$

where r_j is a decimal generated by the function rand() in the range [0, 1]. j_{rand} is an integer randomly selected from the range $[1, D]$. CR is the cross probability factor.

2.1.4.2 Exponential crossover

$$\begin{cases} U_{i,t+1}^{j}=\begin{cases} V_{i,t+1}^{j} & j=<J>_D,<J+1>_D,\ldots,<J+L-1>_D \\ X_{i,t+1}^{j} & otherwise \end{cases} \\ \begin{cases} L=1,\ \beta=rand(0,1) \\ \text{while}(\beta \le CR\ \&\ L<D) \\ \qquad L=L+1,\quad \beta=\text{rand}(0,1) \\ \text{end} \end{cases} \end{cases} \tag{11}$$

The above Eq. (11) is used for exponential crossover. $J=\text{rand}(1,D)$ means that the components from the J^{th} dimension to the $J+L-1^{th}$ dimension of the trial individual $\boldsymbol{U}_{i,t+1}$ are all composed of mutation individual $\boldsymbol{V}_{i,t+1}$. The components of other dimensions are composed of the target vector individual $\boldsymbol{X}_{i,t+1}$. where the angular brackets $<>_D$ denote a modulo function of modulus D. The difference length L is determined by CR.

2.1.5 Selection operation

After the crossover operation, DE will use the "greedy" strategy to select between the trial individual $\boldsymbol{U}_{i,t+1}$ and the target vector individual $\boldsymbol{X}_{i,t}$. The function value of the two under the constraint [see Eq. (1)] is also called the fitness value, and the smaller value is selected as a parent individual of the next generation, while the larger one is eliminated, as shown in Eq. (12) (Brest et al. 2006).

$$\boldsymbol{X}_{i,t+1}=\begin{cases} \boldsymbol{U}_{i,t+1}, \mathrm{f}\left(\boldsymbol{U}_{i,t+1}\right) < \mathrm{f}\left(\boldsymbol{X}_{i,t}\right) \\ \boldsymbol{X}_{i,t}, \mathrm{f}\left(\boldsymbol{U}_{i,t+1}\right) \ge \mathrm{f}\left(\boldsymbol{X}_{i,t}\right) \end{cases} \tag{12}$$

2.2 Shuffled frog leaping

The shuffled frog-leaping algorithm (SFLA) is a heuristic random search algorithm based on swarm intelligence generated by simulating the swarm information sharing and communication mechanism in the foraging process of frogs. The algorithm was first proposed by Eusuff and Lansey (Eusuff and Lansey 2003); it combines the advantages of the memetic algorithm (MA) (Moscato 1989) and particle swarm optimization (PSO) algorithm

(Kennedy and Eberhart 1995) and has the characteristics of fewer parameter settings, fast solving speed, and strong global optimization ability (Rahimi-Vahed and Mirzaei 2007). There are mainly the following three key operations in SFLA (Santander-Jimenez et al. 2018).

2.2.1 Initialization operation

Initialize the frog population as $\boldsymbol{F} = \left(\boldsymbol{X}_1,\ \boldsymbol{X}_2, \dots, \boldsymbol{X}_{\boldsymbol{NP}}\right)$ with NP individuals in total [see Eq. (2)], and each individual is a randomly generated candidate solution. The i^{th} individual is $\boldsymbol{X}_{\boldsymbol{i},\boldsymbol{t}} = \left(x_{i,t}^1,\ x_{i,t}^2, \dots,\ x_{i,t}^j,\ \dots, x_{i,t}^D\right), 1 \le i \le NP$, where t represents the evolutionary number of each meme group, and D represents the dimension of the solution.

2.2.2 Grouping operation

The grouping operation can divide the population into M subpopulations equally. First, the NP candidate solutions in the population are arranged in descending order according to the fitness value calculated by Eq. (1), and then the first candidate solution is classified into the first subpopulation, and the second candidate solutions are classified into the second subpopulation. By analogy, the remainder is calculated according to Eq. (13), and the NP candidate solutions are allocated to M subpopulations, where s represents the subpopulation number to which the i^{th} candidate solution is allocated (Hsu and Yang 2020).

$$s = i\%M, \quad i = 1, 2, \dots, NP \tag{13}$$

2.2.3 Local update operation

$\boldsymbol{Fg}$ represents the candidate solution with the highest fitness value of the population in all previous generations, $\boldsymbol{Fb}_{\boldsymbol{t}}^{s}$ represents the candidate solution with the highest fitness value in the s^{th} subpopulation in the t^{th} evolutionary generation as an optimal solution, and $\boldsymbol{Fw}_{\boldsymbol{t}}^{s}$ represents the candidate solution with the lowest fitness value in the s^{th} subpopulation in the t^{th} evolutionary generation as a worst solution. In each local iterative calculation of the s^{th} subpopulation, $\boldsymbol{Fw}_{\boldsymbol{t}}^{s}$ is updated according to Eq. (14) (Wang et al. 2019). If $\boldsymbol{Fw}'$ is better than $\boldsymbol{Fw}_{\boldsymbol{t}}^{s}$, that is, the fitness value of $\boldsymbol{Fw}'$ is smaller, then $\boldsymbol{Fw}'$ will replace the current $\boldsymbol{Fw}_{\boldsymbol{t}}^{s}$; otherwise, replace $\boldsymbol{Fb}_{\boldsymbol{t}}^{s}$ in Eq. (14) with $\boldsymbol{Fg}$ to recalculate $\boldsymbol{Fw}'$ and compare. If $\boldsymbol{Fw}'$ is still not better than $\boldsymbol{Fw}_{\boldsymbol{t}}^{s}$, a random solution is generated to replace $\boldsymbol{Fw}_{\boldsymbol{t}}^{s}$ by Eq. (2).

$$\boldsymbol{Fw}' = \boldsymbol{Fw}_{\boldsymbol{t}}^{s} + \text{rand}(0, 1) \cdot \left(\boldsymbol{Fb}_{\boldsymbol{t}}^{s} - \boldsymbol{Fw}_{\boldsymbol{t}}^{s}\right) \tag{14}$$

3 Related work

Different operator design and control parameter schemes will have a great impact on the performance of DE in solving problems. The current improvement in DE can be summarized into the following three types (Li et al. 2020b).

3.1 Operation design

The operations are the core of DE, and many algorithms focus on its improvement. In TDE, (Fan and Lampinen 2003) proposed a triangle mutation strategy. It limits the trial individual to a triangular area to improve the efficiency of mutation operation and has achieved better results in neural network learning. In JADE, (Zhang and Sanderson 2009) adopted a new difference method, “DE/current-to-pbest/1”, which randomly selects one of the top p% excellent individuals to participate in the mutation and increase the diversity of the population. In SspDE, (Pan et al. 2011) used a strategy pool containing four mutation strategies and assigned mutation strategies and control parameter values to each individual at the same time. After several generations of evolution, the mutation strategy and the value of the control parameter that correspond to a better individual are redistributed to each individual based on a larger probability to improve the speed of convergence. In DMPSADE, (Fan and Yan 2015) used a strategy pool containing five mutation strategies, which determines the mutation strategy of each individual according to the cumulative probability of each strategy and the roulette algorithm. In MPEDE, (Wu et al. 2016) proposed an integrated DE that uses three different mutation strategies to form multiple subpopulations for simultaneous evolution. In SpDE, (Pan et al. 2018) divided the entire population into three subpopulations and proposed a new two-stage mutation strategy. The difference between individuals is considered in the evolution process. A better balance has been achieved between exploration and exploitation. In HMCFQDE, (Deng et al. 2021) designed a new hybrid mutation strategy based on the advantage of local neighborhood mutation, which greatly overcomes the problems of low efficiency of the original DE algorithm, insufficient diversity in the later stage, and slow convergence speed.

3.2 Control parameter setting

Before solving a specific problem, the relevant control parameters of DE should be determined first. (Storn and Price 1997) represented the number of populations $NP \in [3, 10]$, and the mutation factor $F \in [0.5, 1.0]$ was usually considered to be a more appropriate parameter scheme. (Gämperle et al. 2002) proved that $NP \in [3, 8]$, $F = 0.6$, and $CR \in [0.3, 0.9]$ were better parameter schemes when initializing the population. In fact, there is no fixed parameter setting scheme that can be well applied to various optimization problems. Different parameter schemes should be adopted according to different problems (Elsayed et al. 2013). Therefore, finding a suitable parameter scheme has attracted the attention of many researchers. In FADE, (Liu and Lampinen 2005) proposed an adaptive adjustment scheme that uses a fuzzy logic controller to continuously adjust the mutation factor F and crossover probability CR through multiple iteration cycles of population individuals and corresponding fitness function values. In jDE, (Brest et al. 2006) encoded the control parameters F and CR into the population individuals and adjusted them adaptively in the continuous iterative calculation. In SaDE, (Qin et al. 2009b) believed that it is possible to adaptively generate new individuals of the next generation and parameter values by learning the outstanding individuals of the past and the corresponding control parameter values. In CoDE, (Wang et al. 2011) created a parameter pool, including three control parameter schemes. By comparing the calculation results of the random combination of three independent trial individuals and the parameter schemes, the optimal scheme and its parameter scheme are determined to enter the next generation of evolution. In MDE, (Zou et al. 2013) used a Gaussian distribution and a uniform distribution to adjust the values of mutation factor F and crossover probability CR, respectively. However, in DMPSADE, (Fan and Yan 2015) used two normal distributions to adaptively adjust the control parameter values and achieved better results. In SAMDE, (Zhu et al. 2020) proposed a new mutation strategy "DE/rand assembly/1", where the basis vector is composed of three randomly selected individuals in proportion. In the evolution process, the entire population is randomly divided into three equal-sized subpopulations, and each subpopulation adopts its own different mutation strategy and evolves independently. At the same time, it can generate adaptive control parameters according to evolutionary number. After each generation, the three subpopulations are mixed to regroup. Compared with other enhanced DE algorithms, SAMDE has a strong competitive performance.

3.3 Hybrid strategy

Currently, the strategy of hybridizing DE with different swarm intelligence optimization algorithms has become one of the most effective ways to enhance its optimization performance. In EDA, (Sun et al. 2005) combined the DE algorithm with the estimation of the distribution algorithm to better solve the global optimization problem in continuous space. In AnDE, (Das et al. 2007) merged the DE algorithm with the simulated annealing algorithm and proposed new individual selection conditions and mutation strategies. The experimental results show that the performance of the algorithm is better than that of the traditional DE algorithm. In ESADE, (Guo et al. 2014) used the parameter setting method and mutation strategy in JADE (Zhang and Sanderson 2009). At the same time, the simulated annealing algorithm is integrated into the selection operator, which greatly enhances the global search capability. In DEMPSO, (Mao et al. 2017) first executed DE to obtain the local optimal solution and the individual optimal solution and then used this information to find the global optimal solution through the iterative loop of the particle swarm optimization algorithm. In HyGADE, (Chaudhary et al. 2019) merged DE and an improved genetic algorithm based on the crossover of three parents. Compared with the basic genetic algorithm, the DE algorithm and the genetic algorithm with an improved crossover operation can obtain a better global optimal solution. In SFDE, (Wang et al. 2019) combined DE with the most basic shuffled frog-leaping algorithm, which improved the diversity of the population and the speed of global search during the entire dynamic evolution process. (Parouha and Verma 2021) designed an advanced hybrid algorithm (haDEPSO), which integrated with the advanced DE and PSO. The convergence characteristic of them provided different approximation to the solution space.

We further summarize the algorithms involved in the three types of improvement and the SFSADE proposed in this paper in detail in Table 1. According to the name of the algorithm, the time of creation, the mutation strategy (the number of mutation schemes and the number of individuals required), whether the control parameters are adaptive, whether the population is divided into sub-populations, whether it is integrated with other algorithms, the core idea of the algorithm, testing function status (number and highest dimension), algorithm performance evaluation indicators (iteration time, function value, average, standard deviation, intermediate value, optimal value, worst value) and statistical test type, a total of 11 aspects are summarized and analyzed. In summary, many enhanced DE algorithms have been studied for the improvement of mutation strategies and the design of self-adaptive adjustment mechanisms of control parameters. Consequently, determining how to develop a better hybrid strategy to improve the performance of DE is a pivotal concern current research. The SFSADE proposed in this paper adopts all the above three improvement

Table 1 Comparison of improved DE algorithms

Improvement type	Algorithm name	Creation time	Mutation strategy	Control parameter	Population division	Algorithm hybrid	Core idea	Test functions	Performance metrics	Statistical test
Operation design	TDE	2003	[One, Three]	Fixed	Single	None	Trigonometric mutation	[2; 30D]	[Time, Value]	None
	JADE	2009	[One, Four]	Dynamic	Single	None	DE/current-to-pbest	[20; 100D]	[Mean, Std]	None
	SspDE	2011	[Four, Five]	Dynamic	Single	None	Trial vector generation	[19, 100D]	[Median]	None
	DMPSADE	2015	[Five, Five]	Dynamic	Single	None	Discrete mutation	[25, 50D]	[Mean, Std]	Kruskal–Wallis
	MPEDE	2016	[Three, Three]	Dynamic	Grouped	None	Multi-population ensemble	[25, 50D]	[Mean, Std]	Wilcoxon rank sum
	SpDE	2018	[Two, Three]	Fixed	Grouped	None	two phases mutation	[2, 200D]	[Value]	None
	HMCFQDE	2021	[Two, Two]	Fixed	Grouped	QEA / CCEA	Hybrid mutation	[6, 1000D]	[Mean, Std, Best, Worst]	None
Control parameter setting	FADE	2005	[Nine, Four]	Dynamic	Single	Mamdani's fuzzy inference	Fuzzv logic controller	[6, 20D]	[Value]	None
	jDE	2006	[One, Three]	Dynamic	Single	None	Self-adapting parameters	[21, 30D]	[Mean, Std]	None
	SaDE	2009	[Two, Four]	Dynamic	Single	None	self-adapted by learning	[26, 30D]	[Mean, Std, Rate]	None
	CoDE	2011	[Three, Four]	Dynamic	Single	None	Trial vector generation	[25, 30D]	[Mean, Std]	None
	MDE	2013	[Two, Four]	Dynamic	Single	None	Adjust by two distributions	[25, 50D]	[Mean, Std]	Wilcoxon rank sum
	SAMDE	2020	[Three, Four]	Dynamic	Grouped	None	Parameter adaptation	[25, 50D]	[Mean, Std]	Multiple Sign

Table 1 (continued)

Improvement type	Algorithm name	Creation time	Mutation strategy	Control parameter	Population division	Algorithm hybrid	Core idea	Test functions	Performance metrics	Statistical test
Hybrid strategy	EDE	2005	[One, Three]	Fixed	Single	EDA	Probability model	[5, 10D]	[Mean, Best]	None
	AnDE	2007	[One, Three]	Dynamic	Single	SA	Conditional acceptance function	[6, 100D]	[Mean, Std]	Unpaired t
	ESADE	2014	[Two, Three]	Dynamic	Single	SA	Parameters generation	[17, 30D]	[Mean, Std, Median]	Wilcoxon signed rank and Friedman
	DEMPSO	2017	[One, Three]	Fixed	Single	PSO	Parameter identification	[3, 10D]	[Value]	None
	HyGADE	2019	[One, Three]	Fixed	Single	EA	Hybrid mutation	[46, 30D]	[Value]	None
	SFDE	2019	[One, Three]	Dynamic	Grouped	SFLA	Subgroup sharing	[20, 30D]	[Mean, Std]	None
	haDEPSO	2021	[Two, Two]	Fixed	Grouped	PSO	Advanced hybrid	[30, 30D]	[Mean, Std, Rate]	One-tailed t and Wilcoxon signed rank
Ours	SFSADE	2021	[Three, Three]	Dynamic	Grouped	SFLA	Adaptive mutation and parameters	[25, 50D]	[Mean, Std]	Wilcoxon signed rank and Friedman

methods. It combines the enhanced DE based on the self-adaptive control parameter adjustment mechanism and classification mutation strategy with the meme group idea in shuffled frog leaping and achieves better results. Compared with the latest advanced DE variants, such as SAMDE (Zhu et al. 2020) and SFDE (Wang et al. 2019), SFSADE has great differences and performance improvements.

4 SFSADE

For the DE algorithm, the most critical problem is how to balance the relationship between "explore" and "explicit" (Deng et al. 2020a). Exploration means that the population is more inclined to increase diversity so that the individuals will expand the search range in

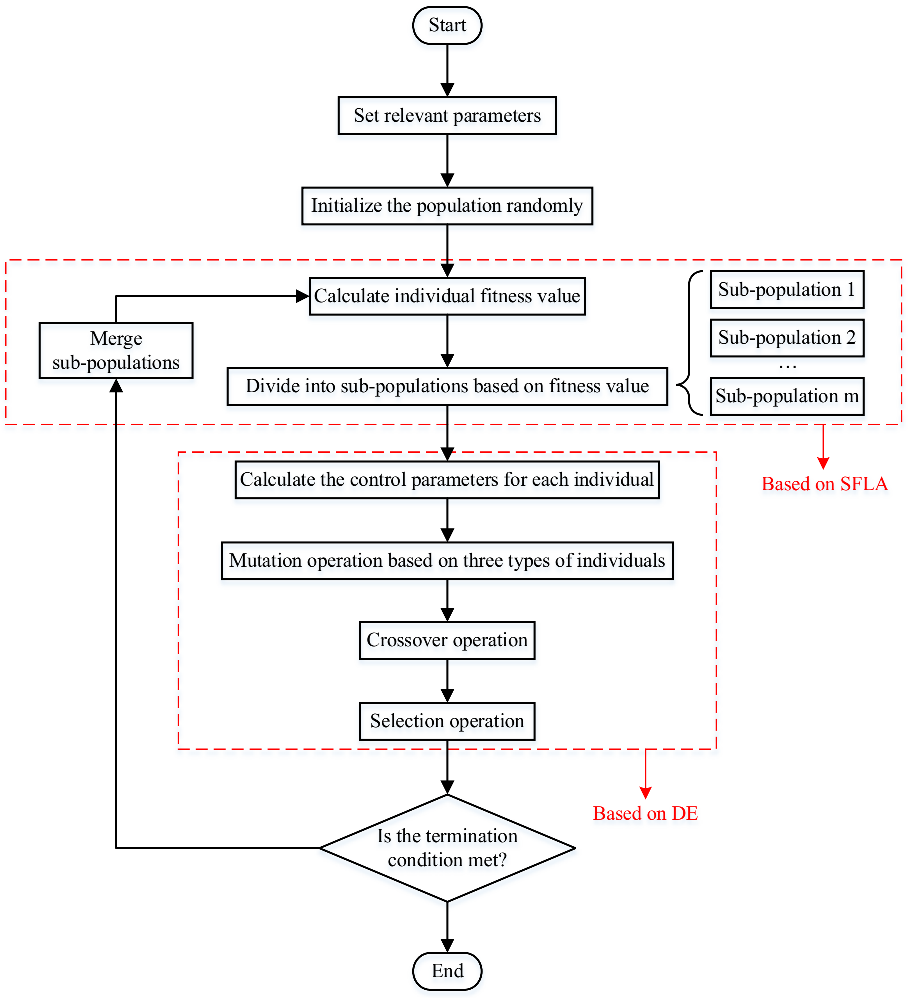

Start
Set relevant parameters
Initialize the population randomly
Calculate individual fitness value
Merge sub-populations
Divide into sub-populations based on fitness value
Sub-population 1
Sub-population 2
…
Sub-population m
Based on SFLA
Calculate the control parameters for each individual
Mutation operation based on three types of individuals
Crossover operation
Selection operation
Based on DE
Is the termination condition met?
End

Fig. 1 Flow chart of the proposed SFSADE

the solution space and improve the search performance and reliability of the algorithm. Explicit means that the population is more inclined to effectively use better individuals and explore better individuals in a local range. To achieve a good balance between them, our proposed SFSADE adopts a variety of strategies to improve DE and innovatively incorporates the idea of the shuffled frog-leaping algorithm. The flow chart of the SFSADE is shown in Fig. 1.

As shown in the figure above, the execution process of SFSADE is generally based on the two algorithms of DE and SFLA. After initializing the parameters and populations of the algorithm, first learn from SFLA and group the populations according to the fitness value of the individuals; then use DE as the core to perform unique mutation, crossover, and selection operations for each individual in different groups; Finally, the groups are merged and looped until the termination condition is met. Through these improvements, the SFSADE can better improve the search efficiency of the population while maintaining the diversity of the population to avoid premature convergence. The specific improvement strategies are described in the following four subsections.

4.1 Shuffled frog-leaping strategy

We are familiar with most swarm intelligence optimization algorithms, such as genetic algorithms (Sampson 1976), ant colony algorithms (Colorni et al. 1991), and particle swarm algorithms (Kennedy and Eberhart 1995). Similar to the most basic DE algorithm, they do not involve grouping operations, but they search and update based on the entire population, and the execution efficiency is low. However, the shuffled frog-leaping algorithm (Eusuff and Lansey 2003) is introduced in the SFSADE proposed in this paper. It divides the population into several small meme groups based on a grouping operation, and each meme group evolves independently of each other in each iteration, which can improve the execution speed of the algorithm. After each iteration, the meme groups will be fused again into a new population to transmit information, which is conducive to searching for the global optimal solution.

In the work of this paper, we will first sort the population according to the fitness values of all individuals, divide the population into a specified number of subpopulations of the same size, expand the diversity of the population, and improve the overall search performance. Then, instead of updating only the worst individual, such as the traditional shuffled frog-leaping algorithm or other enhanced DE algorithms, we will let each individual in each subpopulation perform improved mutation, crossover and other operations to update so that it is helpful for the population to find the global optimal solution. It is worth mentioning that after each iteration, each subpopulation will be merged, and then the grouping evolution in the next generation will be executed according to the process. Through this operation, the information of different subpopulations will be exchanged and shared, which will further enrich the diversity of the population, make it difficult to fall into local extremes and search for the global optimal solution better and faster.

4.2 Classification mutation strategy

The mutation operation in DE is actually a process of local search, where the base vector is the center of the local search, and the difference vector represents the size of the local search range, including the search step and direction. Therefore, the diversity of the population often depends on the diversity of each local search center, namely, the basis vector

(Bi and Xiao 2012). For the usual mutation form, such as "DE/rand/$*$", the basis vector is randomly selected from the population. Although it has good diversity and high search performance, the efficiency is relatively low. Another example is "DE/best/$*$", which uses the best individuals of the population as the basis vector. It can make full use of excellent individuals and has better search efficiency. However, the basis vectors of the entire population lack diversity and tend to converge prematurely (Cai et al. 2021).

In fact, for different basis vectors, the degree of adaptation of the representative individuals may also vary greatly. For the optimization problem [see Eq. (1)], the smaller the fitness value is, the better the fitness, and vice versa. Therefore, different individuals should adopt different schemes in the direction and speed of their mutations to further improve the diversity and overall adaptability of the population at the same time and to search for a better global optimal solution. Based on the above analysis and related research, this paper adopts the following three types of mutation strategies for individuals with different adaptation ranges, as shown in Eqs. (15–17) (Deng et al. 2021).

Strong individuals:$\mathrm{f}\left(\boldsymbol{X}_{i,t}\right) < \mu_{t,s} - 2 \cdot \sigma_{t,s}$

$$\boldsymbol{V}_{i,t} = R_{i,t} \cdot \boldsymbol{X}_{r,t} + G_{i,t} \cdot \left(\boldsymbol{Xgbest}_t - \boldsymbol{X}_{i,t}\right) \tag{15}$$

Ordinary individuals: $\mu_{t,s} - 2 \cdot \sigma_{t,s} < \mathrm{f}\left(\boldsymbol{X}_{i,t}\right) < \mu_{t,s} + 2 \cdot \sigma_{t,s}$

$$\boldsymbol{V}_{i,t} = R_{i,t} \cdot \boldsymbol{X}_{r,t} + S_{i,t} \times \left(\boldsymbol{Xsbest}_t - \boldsymbol{X}_{i,t}\right) + G_{i,t} \cdot \left(\boldsymbol{Xgbest}_t - \boldsymbol{X}_{i,t}\right) \tag{16}$$

Poor individuals: $\mathrm{f}\left(\boldsymbol{X}_{i,t}\right) > \mu_{t,s} + 2 \cdot \sigma_{t,s}$

$$\boldsymbol{V}_{i,t} = R_{i,t} \cdot \boldsymbol{X}_{r,t} + S_{i,t} \cdot \left(\boldsymbol{Xsbest}_t - \boldsymbol{X}_{i,t}\right) \tag{17}$$

where $\mathrm{f}\left(\boldsymbol{X}_{i,t}\right)$ represents the fitness value of the t^{th} generation individual $\boldsymbol{X}_i$ under the constraint [see Eq. (1)], and $\mu_{t,s}$ and $\sigma_{t,s}$, respectively, represent the average and standard deviation of the fitness value of the s^{th} meme group in the t^{th} generation. $\boldsymbol{V}_{i,t}$ represents the target individual corresponding to the current individual $\boldsymbol{X}_{i,t}$, $\boldsymbol{X}_{r,t}$ is an individual that is randomly selected from the population at the time of mutation as the basis vector that is different from $\boldsymbol{X}_{i,t}$, $\boldsymbol{Xsbest}_t$ is the best individual in the current meme group of the t^{th} generation, and $\boldsymbol{Xgbest}_t$ represents the global optimal individual within the t^{th} generation. $R_{i,t}$, $S_{i,t}$, and $G_{i,t}$ are, respectively, defined as random mutation coefficient, grouping mutation coefficient, and global mutation coefficient.

According to Chebyshev's theorem (Chebyshev 1867), regardless of the data distribution,, at least 75% of the data values are within two standard deviations of the average. As shown in Fig. 2, approximately 12.5% of individuals $\boldsymbol{X}_{i,t}$ have a fitness value $\mathrm{f}\left(\boldsymbol{X}_{i,t}\right)$ less than $\mu_{t,s} - 2 \cdot \sigma_{t,s}$. We define them as strong individuals, indicating that they have strong adaptability. These individuals implement the mutation strategy of Eq. (15), that is, try to retain the genes of the best individuals and improve local search capabilities. Similarly, approximately 12.5% of individuals have fitness values greater than $\mu_{t,s} + 2 \cdot \sigma_{t,s}$, and they are defined as poor individuals, indicating that they have weaker adaptability. These individuals implement the mutation strategy of Eq. (17), which promotes their learning from excellent individuals, enhances the individual's exploration ability and global search ability, and accelerates the convergence to the optimal solution. Approximately 75% of individuals have fitness values within the interval $\left[\mu_{t,s} - 2 \cdot \sigma_{t,s},\ \mu_{t,s} + 2 \cdot \sigma_{t,s}\right]$, and we define these individuals as ordinary individuals. These individuals implement the mutation strategy of Eq. (16), that is, while retaining the best genes to learn from them, they also improve the individual's exploration ability and global search ability.

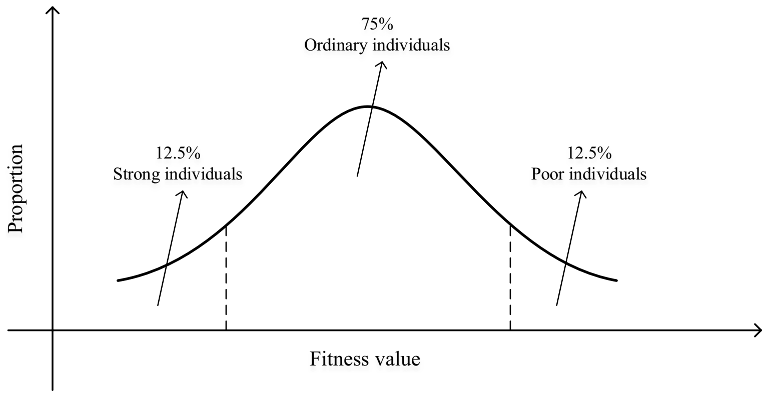


Fig. 2 Distribution of individuals in the population

4.3 Self-adaptive control parameters strategy

In the DE, the setting of control parameters and the adjustment schemes have an important influence on the performance of the algorithm. Inappropriate parameter settings may cause the algorithm to converge too early or too slowly, and it is impossible to search for a better global optimal solution (Deng et al. 2020b). In the earlier enhanced DE, fixed control parameter values were often used. Although this method is simple, it is usually difficult to determine the appropriate parameter values, and it is also not conducive to the performance of the algorithm. The current trend is to use a self-adaptive control parameter strategy, that is, to determine different parameter schemes according to the actual situation of each individual.

Based on the new classification mutation strategy proposed above, in our improved DE algorithm SFSADE, there are mainly four control parameters $R_{i,t}$, $S_{i,t}$, $G_{i,t}$ and $CR_{i,t}$. The first three parameters control the mutation operation, and the last parameter controls the proportion of the mutation individual $\boldsymbol{V}_{i,t}$ in the trial individual $\boldsymbol{U}_{i,t}$. These parameters all affect the convergence speed and population diversity of the algorithm. The increase in $R_{i,t}$ or $CR_{i,t}$ helps to increase the degree of random mutation of the individual, enhances the global exploration ability of the population, and maintains the diversity of the population. Increasing the parameters $S_{i,t}$ or $G_{i,t}$ helps to enhance the local search ability of the individual and the exploitation ability of the population.

In the current enhanced DE, most of the adaptive parameter adjustment strategies are dynamically adjusted according to the evolutionary number of the algorithm. However, the strategy proposed in this paper takes into account the degree of adaptation of different individuals on that basis, that is, adaptively adjusts the control parameters of each individual according to the evolutionary number and the individual's fitness value. See the following Eqs. (18–21) for details(Bi and Xiao 2012).

$$R_{i,t} = R_{min} + \left(R_{max} - R_{min}\right) \cdot \left(\frac{1}{2} \cdot \left(2 - 2^{\frac{t}{T}}\right) + \frac{1}{2} \cdot \left(\frac{f_{i,t} - f_{min}}{f_{max} - f_{min}}\right)\right) \tag{18}$$

$$S_{i,t} = S_{min} + \left(S_{max} - S_{min}\right) \cdot \left(\frac{1}{3} \cdot \left(2^{\frac{t}{T}} - 2\right) + \frac{2}{3} \cdot \left(\frac{f_{max} - f_{i,t}}{f_{max} - f_{min}}\right)\right) \tag{19}$$

$$G_{i,t} = G_{min} + \left(G_{max} - G_{min}\right) \cdot \left(\frac{1}{3} \cdot \left(2^{\frac{t}{T}} - 2\right) + \frac{2}{3} \cdot \left(\frac{f_{max} - f_{i,t}}{f_{max} - f_{min}}\right)\right) \tag{20}$$

$$CR_{i,t} = CR_{min} + \left(CR_{max} - CR_{min}\right) \cdot \left(\frac{1}{2} \cdot \left(2 - 2^{\frac{t}{T}}\right) + \frac{1}{2} \cdot \left(\frac{f_{i,t} - f_{min}}{f_{max} - f_{min}}\right)\right) \tag{21}$$

where $R_{i,t} \in \left[R_{min},\ R_{max}\right]$, $S_{i,t} \in \left[S_{min},\ S_{max}\right]$, $G_{i,t} \in \left[G_{min},\ G_{max}\right]$, $CR_{i,t} \in \left[CR_{min},\ CR_{max}\right]$, t is the current evolutionary number, T is the total evolutionary number, and f_{max} and f_{min} are the fitness values corresponding to the optimal individual and the worst individual of the group when the evolutionary number is t, respectively.

In Fig. 3 above, the change in the four control parameters in Shifted Rosenbrock's function test is shown. For our proposed parameter self-adaptive adjustment mechanism, its advantages can be analyzed from three aspects. The first is the influence of algorithm evolution number on control parameters. With the increase in evolutionary number, the enhancement of the individual's global search ability and population exploration ability should gradually transition to increasing the individual's local search ability and population exploitation ability so that the algorithm can reach the global optimal solution faster in the later stage. Therefore, the values of these four control parameters decrease as the number of algorithm iterations increases. The second is the influence of the degree of adaptation of each individual on the control parameters. For individuals with poor fitness in the population, the degree of mutation and crossover should be increased to promote their progress in a better search direction. Finally, there is the weight relationship between evolutionary numbers and individual fitness values. For $R_{i,t}$ and $CR_{i,t}$, because they are more inclined to determine the degree of random evolution of the individual, it may be possible to make the weights of the two equal. $S_{i,t}$ and $G_{i,t}$

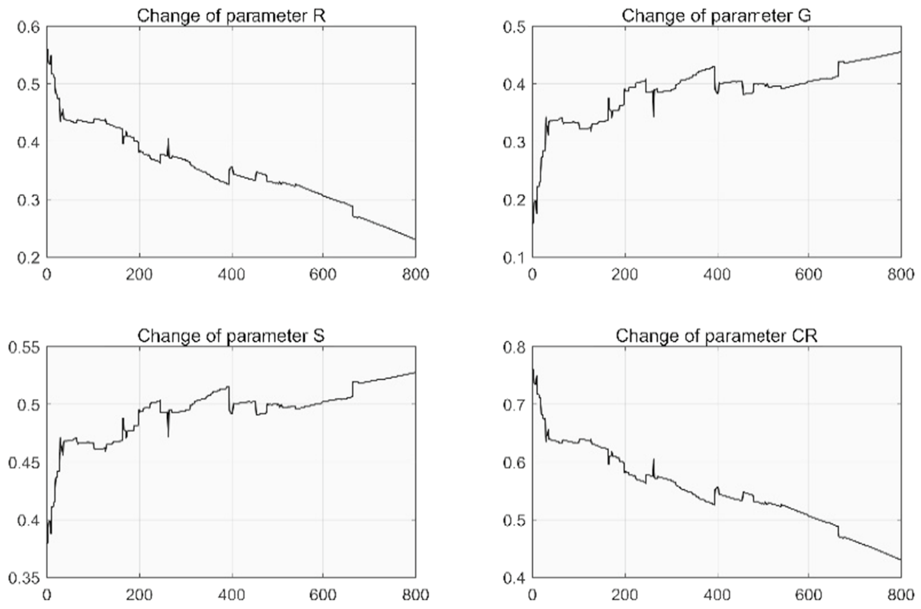


Fig. 3 Trend of four control parameters

are more inclined to determine the directional convergence of the individual, so it is more reasonable to set the influence ratio of the individual fitness values to twice the evolutionary numbers.

4.4 The basic process of SFSADE

Through the three strategies to improve DE introduced above, we come to the detailed framework of SFSADE as given in Algorithm 1. Generally speaking, the complexity of the optimization algorithm is divided into time complexity and space complexity (Guanghui et al. 2020). However, the time consumed by the direct execution of the algorithm is very dependent on the test environment and also affected by the scale of the data. In view of this, we analyze its time complexity from the code itself. It can be estimated by calculating the number of basic operations that require constant time. Its usual function form is $Time_n = f(n)$, where n is the dimension of the problem, $Time_n$ is the execution time of the n-dimensional algorithm, f$(*)$ is a function that returns the execution time (Mirsadeghi and Khodayifar 2021).

Algorithm 1 Pseudocode of SFSADE

```
Set the population size $NP$, the number of meme groups $N$, the solution dimension $D$ and the boundary
[$X_{min}, X_{max}$], the initial evolution algebra $t$, the total number of population iterations $T$, the number of
iterations within each group $K$, the control parameter boundary $R_{i,t} \in [R_{min}, R_{max}]$, $S_{i,t} \in [S_{min}, S_{max}]$,
$G_{i,t} \in [G_{min}, G_{max}]$, $CR_{i,t} \in [CR_{min}, CR_{max}]$;
According to Eq. (2), the initial population is randomly distributed in the solution space;
%Generational evolution
while $t \le T$
    for $i = 1:NP$
    Calculate the fitness value f($\boldsymbol{X_i}$) of each individual $\boldsymbol{X_i}$ according to Eq. (1);
    end
    Find the best individual $\boldsymbol{Xgbest}$ within the $t^{th}$ generation;
  Sort all individuals in the population according to the fitness value and divide the population into $N$
    groups equally through the grouping operator [see Eq. (13)];
    %Group evolution
  for $s = 1:N$
    Find the best individual $\boldsymbol{Xsbest}$ in the group $s$ of the $t^{th}$ generation;
        %Evolution within the group
    for $k = 1:K$
      Calculate the mean and standard deviation of the fitness of individuals in the $s^{th}$ group;
      for $i = 1:NP/N$
                %Perform mutation operation
        According to Eqs. (18-20), calculate the control parameters $R$, $S$, $G$ corresponding to the
                mutation of individual $\boldsymbol{X_i}$;
        Carry out classification mutation according to Eqs. (15-17);
        According to Eq. (8), detection operation is performed on variant individuals;
        %Perform Crossover operation
        Calculate the cross probability $CR$ according to Eq. (21);
        Perform binomial crossover according to Eq. (10);
        %Perform selection operation
        According to Eq. (12), the population adopts a greedy strategy to evolve;
      end
    end
  end
  Mix the groups to form a new population;
  $t = t + 1$;
end
```

There are many types of time complexity: constant time (does not depend on the size of the problem), linear time (increases in a linear manner as the problem size increases), polynomial time (increases as the problem dimension based on polynomial function), exponential time (exponentially increases with the increase of the problem dimension), factorial time (calculates according to the factorial as the problem size increases) and so on, respectively corresponding to$o(1)$,$o(n)$, $o(n^a)$,$o(b^n)$, $o(n!)$ and so on. Most optimization algorithms, such as differential evolution, particle swarm optimization, simulated annealing, etc., have a time complexity of$o(p(n + \mathrm{CF}))$, where p is the size of the population, n is the dimension of the problem, and CF is the Cost Function time complexity (Mirsadeghi and Khodayifar 2021). Therefore, as shown in the code in Algorithm 1, due to the additional nesting of loops outside, the time complexity of SFSADE is$o(p(n^2 + \mathrm{CF}))$.

5 Experimental study

To verify the effect of the above improvement measures and the performance of the improved SFSADE algorithm, we conducted a large number of simulation experiments. The experimental program is written and run in MATLAB R2016a, and the calculation is performed on a 64-bit computer with an Intel(R) Core(TM) i7-10875H @ 2.3 GHz CPU, 16 GB RAM, and a Windows 10 operating system. At the same time, the test function selected by our experiment, the setting of related parameters, and the improved DE for comparison are shown in the literature that proposed the SAMDE algorithm (Zhu et al. 2020).

5.1 Selection test function

To comprehensively evaluate the performance of our algorithm, we implemented and tested it on the 25 benchmark functions of CEC 2005 (Suganthan et al. 2005). As shown in Table 2 below, it includes unimodal functions $\mathrm{F1}-\mathrm{F5}$, basic multimodal functions $\mathrm{F6}-\mathrm{F12}$, extended multimodal functions $\mathrm{F13}-\mathrm{F14}$, and hybrid multimodal functions $\mathrm{F15}-\mathrm{F25}$. In fact, the number of local optimal solutions of most multimodal functions will increase exponentially with the increase of the dimension of the problem variables, which makes multimodal functions one of the most difficult optimization problems to solve (Bi and Xiao 2012).

5.2 Comparison with state-of-the-art DE algorithms

We compare SFSADE with 7 advanced DE algorithms in JADE (Zhang and Sanderson 2009), SaDE (Qin et al. 2009a), SOUPDE (Weber et al. 2011), EPSDE (Mallipeddi et al. 2011), ESADE (Guo et al. 2014), MPEDE (Wu et al. 2016), and SAMDE (Zhu et al. 2020). Among them, SOUPDE and MPEDE use multiple subpopulation coevolution strategies, such as SFSADE. SaDE and EPSDE have multiple mutation strategies, such as SFSADE. JADE and ESADE can adaptively adjust control parameters such as SFSADE. As the latest improved DE algorithm, SAMDE integrates all the above improvement strategies, such as SFSADE. Therefore, we choose these enhanced DE variants for comparison, which has strong significance and pertinence.

To carry out comparative experiments more fairly, we refer to the relevant parameter settings in (Zhu et al. 2020) and compare them with the relevant calculation results. The maximum number of function evaluations is set as $2000 \times D$, which is equal to the population evolution number T multiplied by the evolution number of meme groups N in our SFSADE algorithm. The size of the population is also set to 60, and the boundary range of the control parameters R, S, G and CR is also $[0, 1]$. The parameter settings of the other comparison algorithms are also the same. Moreover, the following settings are implemented: JADE with $NP = 100$; SaDE with $NP = 50$, $lp = 30$; SOUPDE with $NP = 30$,

Table 2 Benchmark functions

Function	Initialization range	Global optimum
F1: Shifted Sphere Function $f_1(X) = \sum_{i=1}^{D} x_i^2$	$-100 \le x_i \le 100$	$f_1(0) = 0$
F2: Shifted Schwefel's Problem 1.2 $f_2(X) = \sum_{i=1}^{D}\left(\sum_{j=1}^{i} x_j\right)^2$	$-100 \le x_i \le 100$	$f_2(0) = 0$
F3: Shifted Rotated High Conditioned Elliptic Function $f_3(X) = \sum_{i=1}^{D}\left(10^6\right)^{\frac{i-1}{D-1}} x_i^2,\ \ x = x_{\text{initial}} \times M,\ M\ :\ \ orthogonal\ matrix.$	$-100 \le x_i \le 100$	$f_3(0) = 0$
F4: Shifted Schwefel's Problem 1.2 with Noise in Fitness $f_4(X) = \sum_{i=1}^{D}\left(\sum_{j=1}^{i} x_j\right)^2 \times (1 + 0.4\lvert N(0,1)\rvert)$	$-100 \le x_i \le 100$	$f_4(0) = 0$
F5: Schwefel's Problem 2.6 with Global Optimum on Bounds $f_5(X) = \max\left\{\lvert A_i x - B_i\rvert\right\}$ A is $a\ D \times D\ matrix,\ a_{ij}$ are $integer\ random\ numbers\ in\ the\ range[-500, 500],$ $det(A) \ne 0,\ A_i\ is\ the\ i^{th}\ row\ of\ A,\ B_i = A_i \times o,\ o\ is\ a\ D \times 1\ vector,\ o_i\ are\ random$ $number\ in\ the\ range[-100, 100].$	$-100 \le x_i \le 100$	$f_5(0) = 0$
F6: Shifted Rosenbrock's Function $f_6(X) = \sum_{i=1}^{D-1}\left(100\left(x_i^2 - x_{i+1}\right)^2 + \left(x_i - 1\right)^2\right)$	$-100 \le x_i \le 100$	$f_6(1) = 0$
F7: Shifted Rotated Griewank's Function without Bounds $f_7(X) = \sum_{i=1}^{D}\frac{x_i^2}{4000} - \prod_{i=1}^{D}\frac{x_i}{\sqrt{i}} + 1$ $x = x_{\text{initial}} \times M,\ M = M'(1 + 0.3\lvert N(0,1)\rvert),$ $M'\ :\ \ linear\ transformation\ matrix,\ condition\ number = 3.$	$-600 \le x_i \le 600$	$f_7(0) = 0$
F8: Shifted Rotated Ackley's Function with Global Optimum on Bounds $f_8(X) = -20exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - exp\left(\frac{1}{D}\sum_{i=1}^{D} cos\left(2\pi x_i\right)\right) + 20 + e$ $x = x_{\text{initial}} \times M,\ M\ :\ \ linear\ transformation\ matrix,\ condition\ number = 100.$	$-32 \le x_i \le 32$	$f_8(0) = 0$

Table 2 (continued)

Function	Initialization range	Global optimum
F9: Shifted Rastrigin's Function $f_9(X) = \sum_{i=1}^{D} \left(x_i^2 - 10cos\left(2\pi x_i\right) + 10\right)$	$-5 \le x_i \le 5$	$f_9(0) = 0$
F10: Shifted Rotated Rastrigin's Function $f_{10}(X) = \sum_{i=1}^{D} \left(x_i^2 - 10cos\left(2\pi x_i\right) + 10\right)$		
$x = x_{\text{initial}} \times M,\ M:\ linear\ transformation\ matrix,\ condition\ number = 2.$	$-5 \le x_i \le 5$	$f_{10}(0) = 0$
F11: Shifted Rotated Weierstrass Function $f_{11}(X) = \sum_{i=1}^{D} \left(\sum_{k=0}^{k_{max}} [a^k cos(2\pi b^k (x_i + 0.5))] \right) - D \sum_{k=0}^{k_{max}} [a^k cos(2\pi b^k \times 0.5)]$ $a = 0.5, b = 3, kinitial_{max}$		
$M:$ *linear* transformation matrix, condition number=5.	-0.5 $\le x_i \le$ 0.5	$f_{11}(0) = 0$
F12: Schwefel's Problem 2.13 $f_{12}(X) = \sum_{i=1}^{D} \left(A_i - B_i(x)\right)^2$ $A_i = \sum_{j=1}^{D} \left(a_{ij} \sin\alpha_j + b_{ij}\cos\alpha_j\right),\ B_i(x) = \sum_{j=1}^{D} \left(a_{ij}\sin x_j + b_{ij}\cos x_j\right),\ for\ i = 1, 2, \dots, D$ $A,\ B$ are *two matrix*, $a_{ij},\ b_{ij}$ are int*eger random numbers*		
$\alpha = \left[\alpha_1, \alpha_2, \dots, \alpha_D\right],\ \alpha_j$ are int*eger random numbers in the range*$[-\pi, \pi]$.	$-\pi \le x_i \le \pi$	$f_{12}(\alpha) = 0$
F13: Shifted Extended Griewank's plus Rosenbrock's Function (F8F2) $f_{13}(X) = F8\left(F2\left(x_1, x_2\right)\right) + F8\left(F2\left(x_2, x_3\right)\right) + \dots + F8\left(F2\left(x_{D-1}, x_D\right)\right) + F8\left(F2\left(x_D, x_1\right)\right)$	$-3 \le x_i \le 1$	$f_{13}(0) = 0$
F14: Shifted Rotated Expanded Scaffer's F6 $f_{14}(X) = F\left(x_1, x_2\right) + F\left(x_2, x_3\right) + \dots F\left(x_{D-1}, x_D\right) + F\left(x_D, x_1\right)$ $F(x, y) = 0.5 + \frac{\left(\sin^2\left(\sqrt{x^2+y^2}\right) - 0.5\right)}{\left(1 + 0.001\left(x^2+y^2\right)\right)^2}$		
$x = x_{\text{initial}} \times M,\ M:\ linear\ transformation\ matrix,\ condition\ number = 3.$	$-100 \le x_i \le 100$	$f_{14}(0) = 0$
F15: Hybrid Composition Function $f_{1-2}(x) = Rastrigin's\ Function$ $f_{3-4}(x) = Weierstrass\ Function$ $f_{5-6}(x) = Griewank's\ Function$ $f_{7-8}(x) = Ackley's\ Function$		

Table 2 (continued)

Function	Initialization range	Global optimum
$f_{9-10}(x) = Sphere\ Function$		
$\sigma_i = 1\ for\ i = 1, 2, \ldots, D,$		
$\lambda = \left[1, 1, 10, 10, 5/60, 5/60, 5/32, 5/32, 5/100, 5/100\right],$		
M_i *are all identity matrices.*	$-5 \le x_i \le 5$	$f_{15}(0) = 0$
F16: Rotated Hybrid Composition Function		
Except M_i *are different linear transformation matrixes with*		
condition number of 2, *all other settings are the same as* $F_{15.}$	$-5 \le x_i \le 5$	$f_{16}(0) = 0$
F17: Rotated Hybrid Composition Function with Noise in Fitness		
$f_{17}(x) = F_{16} \times (1 + 0.2\lvert N(0, 1)\rvert)$		
All other settings are the same as F_{16}.	$-5 \le x_i \le 5$	$f_{17}(0) = 0$
F18: Rotated Hybrid Composition Function		
$f_{1-2}(x) = Ackley's\ Function$		
$f_{3-4}(x) = Rastrigin's\ Function$		
$f_{5-6}(x) = Sphere\ Function$		
$f_{7-8}(x) = Weierstrass\ Function$		
$f_{9-10}(x) = Griewank's\ Function$		
$\sigma = [1, 2, 1.5, 1.5, 1, 1, 1.5, 1.5, 2, 2];$		
$\lambda = \lambda = \left[2 * 5/32;5/32;2 * 1;1;2 * 5/100;5/100;2 * 10;10;2 * 5/60;5/60\right];$		
M_i *are all rotation matrices. Condition numbers are* [2323232030200300].	$-5 \le x_i \le 5$	$f_{18}(0) = 0$
F19: Rotated Hybrid Composition Function with a Narrow Basin for the Global Optimum		
All settings are the same as F_{18} *except* $\sigma = [0.1, 2, 1.5, 1.5, 1, 1, 1.5, 1.5, 2, 2],$		
$\lambda = \left[0.1 \times 5/32;5/32;2 \times 1;1;2 \times 5/100;5/100;2 \times 10;10;2 \times 5/60;5/60\right].$	$-5 \le x_i \le 5$	$f_{19}(0) = 0$
F20: Rotated Hybrid Composition Function with the Global Optimum on the Bounds		
All settings are the same as F_{18} *except after load the data file,*		
$set\ o_{1(2j)} = 5,\ for\ j = 1, 2, \ldots, D/2.$	$-5 \le x_i \le 5$	$f_{20}(0) = 0$
F21: Rotated Hybrid Composition Function		
$f_{1-2}(x) = Rotated\ Expanded\ Scaffer's\ F6\ Function$		
$f_{3-4}(x) = Rastrigin's\ Function$		

Table 2 (continued)

Function	Initialization range	Global optimum
$f_{5-6}(x) = F8F2\ \ Function$		
$f_{7-8}(x) = Weierstrass\ Function$		
$f_{9-10}(x) = Griewank's\ Function$		
$\sigma = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2];$		
$\lambda = \left[5 \times 5/100;5/100;5 \times 1;1;5 \times 1;1;5 \times 10;10;5 \times 5/200;5/200\right];$		
M_i *are all orthogonal matrix.*	$-5 \le x_i \le 5$	$f_{21}(0) = 0$
F22: Rotated Hybrid Composition Function with High Condition Number Matrix *All settings are the same as* F_{21} *except* M_i'*s condition numbers* *are* [10205010020010002000300040005000].	$-5 \le x_i \le 5$	$f_{22}(0) = 0$
F23: Non-Continuous Rotated Hybrid Composition Function *All settings are the same as* F_{21}. $Except\ x_j = \begin{cases} x_j & \left|x_j - o_{1j}\right| < 1/2 \\ round(2x_j)/2 & \left|x_j - o_{1j}\right| \ge 1/2 \end{cases}\ for\ j = 1, 2, \dots, D$ $round(x) = \begin{cases} a-1 & if\ x \le 0\&b \ge 0.5 \\ a & if\ b < 0.5 \\ a+1 & if\ x > 0\&b \ge 0.5 \end{cases}$ *where a is x′s intgral part and b is x′s decimal part.* All "round" operators in this document use the same schedule	$-5 \le x_i \le 5$	$f_{23}(0) = 0$
F24: Rotated Hybrid Composition Function		
$f_1(x) = Weierstrass\ Function$		
$f_2(x) = Rotated\ Expanded\ Scaffer's\ F6\ Function$		
$f_3(x) = F8F2\ \ Function$		
$f_4(x) = Ackley's\ Function$		
$f_5(x) = Rastrigin's\ Function$		
$f_6(x) = Griewank's\ Function$		
$f_7(x) = Non - Continuous\ Expanded\ Scaffer's\ F6\ Function$		

Table 2 (continued)

Function	Initialization range	Global optimum
$f_8(x) = Non - Continuous\ Rastrigin's\ Funtion$		
$f_9(x) = High\ Conditioned\ Elliptic\ Function$		
$f_{10}(x) = Sphere\ Funtion\ with\ Noise\ in\ Fitness$		
$\sigma = 2$, for $i = 1,2, \ldots ,$D;		
$\lambda = [10; 5/20; 1; 5/32; 1; 5/100; 5/50; 1; 5/100; 5/100]$;		
M_i *are all matrices, condition numbers are* [100 50 30 10 5 5 4 3 2 2].	$-5 \le x_i \le 5$	$f_{24}(0) = 0$
F25: Rotated Hybrid Composition Function without Bounds *All settings are the same as* F_{24} *except no exact*		
search range set for this test function.	$-2 \le x_i \le 5$	$f_{25}(0) = 0$

Table 3 Comparison of the average error (standard deviation) on 10 dimensions (Zhu et al. 2020)

	JADE	SaDE	SOUPDE	EPSDE	ESADE	MPEDE	SAMDE	SFSADE
F1	8.00E-10 (3.67E-10) –	3.45E-18 (1.73E-18) –	2.44E-06 (5.66E-06) –	1.86E-17 (9.26E-18) –	8.96E-15 (3.59E-15) –	1.24E+00 (2.71E-02) –	1.56E-22 (7.31E-23) –	**7.81E-73** **(2.97E-72)**
F2	4.86E-04 (2.54E-04) +	3.77E-03 (2.05E-03) +	3.39E+00 (2.16E+00) –	1.23E-03 (6.65E-04) +	2.94E-02 (2.57E-02) +	3.95E+01 (1.14E+00) –	**3.37E-09** **(1.82E-09)** +	5.75E-01 (8.51E-02)
F3	**7.00E+01** **(3.65E+01)** +	8.64E+05 (5.29E+05) –	3.51E+06 (2.49E+06) –	6.84E+04 (3.24E+04) +	2.74E+03 (5.94E+01) +	2.38E+05 (1.03E+04) –	6.23E+02 (1.33E+02) +	1.83E+05 (5.22E+04)
F4	7.97E-03 (4.15E-03) +	6.97E-02 (3.74E-02) +	1.24E+01 (8.69E+00) –	1.20E-02 (6.05E-03) +	5.82E-02 (4.74E-02) +	9.77E+01 (1.94E+00) –	**7.77E-08** **(3.71E-08)** +	1.67E+00 (3.49E+00)
F5	8.28E-01 (1.75E-01) +	3.48E-02 (6.54E-03) +	1.21E+02 (3.79E+01) +	3.06E-03 (7.80E-04) +	8.71E-04 (1.86E-04) +	1.95E+02 (2.28E+00) +	**1.46E-04** **(2.43E-05)** +	8.76E+02 (9.26E+02)
F6	2.06E+01 (6.38E+01) –	4.86E+00 (1.74E-01) +	1.73E+02 (1.22E+02) –	2.15E+00 (2.31E-01) +	7.82E+00 (1.57E+01) +	2.83E+03 (9.62E+01) –	**1.21E+00** **(6.58E-02)** +	1.07E+01 (3.78E+00)
F7	8.86E-01 (1.07E-01) –	7.53E-01 (1.49E-01) –	9.75E-01 (1.03E-01) –	7.98E-01 (1.28E-01) –	1.92E-01 (5.92E-03) –	1.33E+00 (7.26E-03) –	6.30E-01 (1.29E-01) –	**1.51E-01** **(2.87E-02)**
F8	2.11E+01 (1.61E-01) –	2.08E+01 (1.25E-01) –	2.09E+01 (1.32E-01) –	2.08E+01 (1.31E-01) –	2.16E+01 (2.09E-01) –	2.10E+01 (6.23E-03) –	2.08E+01 (1.31E-01) –	**2.02E+01** **(5.76E-02)**
F9	6.67E+00 (2.05E+00) –	9.44E+00 (2.71E+00) –	4.58E+00 (2.66E+00) –	9.98E+00 (3.10E+00) –	6.12E+01 (1.44E+01) –	4.46E+01 (6.16E-01) –	2.32E+01 (6.71E+00) –	**1.31E-13** **(1.75E-13)**
F10	7.11E+01 (1.17E+01) –	4.85E+01 (9.24E+00) –	5.67E+01 (1.13E+01) –	5.05E+01 (9.62E+00) –	1.11E+02 (2.11E+01) –	7.13E+01 (4.75E-01) –	5.15E+01 (8.19E+00) –	**1.30E+01** **(3.69E+00)**

Table 3 (continued)

	JADE	SaDE	SOUPDE	EPSDE	ESADE	MPEDE	SAMDE	SFSADE
F11	1.40E+01 (1.26E+00) –	1.12E+01 (1.22E+00) –	1.12E+01 (1.26E+00) –	1.12E+01 (1.20E+00) –	**6.63E-01** **(1.17E-01)** +	1.35E+01 (6.86E-02) –	1.14E+01 (1.29E+00) –	4.00E+00 (6.02E-01)
F12	2.09E+03 (2.80E+03) –	5.97E+02 (5.25E+02) –	3.04E+03 (1.51E+03) –	3.25E+02 (1.18E+02) –	3.54E+02 (1.26E+03) –	9.37E+03 (2.25E+02) –	1.69E+02 (1.20E-03) –	**3.03E-02** **(1.92E-02)**
F13	3.77E+00 (8.69E-01) –	3.09E+00 (6.98E-01) –	2.53E+00 (7.38E-01) –	2.86E+00 (7.41E-01) –	1.14E+01 (4.37E+00) –	5.38E+00 (5.53E-02) –	4.11E+00 (8.03E-01) –	**1.39E-01** **(2.53E-03)**
F14	4.58E+00 (1.63E-01) –	4.22E+00 (1.84E-01) –	4.29E+00 (1.99E-01) –	4.22E+00 (1.98E-01) –	4.92E+00 (1.31E-01) –	4.50E+00 (8.62E-03) –	4.18E+00 (1.86E-01) –	**2.84E+00** **(3.92E-02)**
F15	3.94E+02 (1.48E+02) –	2.20E+02 (8.52E+01) –	2.97E+02 (1.33E+02) –	2.20E+02 (5.93E+01 –	3.18E+02 (2.81E+01) –	5.86E+02 (2.50E+00) –	2.95E+02 (2.63E+01) –	**4.64E+00** **(4.87E+00)**
F16	2.60E+02 (3.94E+01) –	2.09E+02 (2.79E+01) –	2.34E+02 (3.15E+01) –	2.09E+02 (2.27E+01) –	3.03E+02 (4.42E+01) –	2.60E+02 (1.64E+00) –	2.04E+02 (2.24E+01) –	**9.27E+01** **(2.76E+01)**
F17	2.99E+02 (4.19E+01) –	2.33E+02 (2.93E+01) –	3.01E+02 (6.34E+01) –	2.39E+02 (2.60E+01) –	3.68E+02 (5.94E+01) –	2.96E+02 (1.72E+00) –	2.32E+02 (2.42E+01) –	**1.59E+02** **(8.22E+01)**
F18	7.75E+02 (5.78E+01) –	7.96E+02 (2.23E+01) –	6.59E+02 (4.36E+01) –	6.35E+02 (2.93E+00) –	8.32E+02 (1.19E-01) –	8.20E+02 (1.81E+00) –	6.85E+02 (6.83E-04) –	**3.15E+02** **(1.28E+02)**
F19	7.53E+02 (5.50E+01) –	7.15E+02 (2.12E+01) –	6.67E+02 (5.73E+01) –	7.78E+02 (5.79E+00) –	8.07E+02 (1.04E-01) –	8.14E+02 (1.56E+00) –	6.65E+02 (4.81E-04) –	**3.94E+02** **(7.07E+01)**
F20	7.84E+02 (2.65E+01) –	6.51E+02 (2.66E+01) –	6.70E+02 (4.15E+01) –	6.96E+02 (2.03E+00) –	8.23E+02 (1.24E-01) –	8.29E+02 (3.11E+00) –	7.47E+02 (2.90E-03) –	**3.82E+02** **(3.58E+01)**

Table 3 (continued)

	JADE	SaDE	SOUPDE	EPSDE	ESADE	MPEDE	SAMDE	SFSADE
F21	5.01E+02 (1.25E+02) –	7.22E+02 (2.51E+01) –	5.76E+02 (5.81E+00) –	4.85E+02 (7.63E-08) –	6.90E+02 (1.58E-03) –	8.17E+02 (1.27E+00) –	4.83E+02 (1.37E-06) –	**4.09E+02** **(1.10E+02)**
F22	8.03E+02 (1.07E+01) –	7.91E+02 (6.54E+00) –	7.93E+02 (1.94E+01) –	7.73E+02 (5.82E+00) –	8.05E+02 (1.20E+01) –	8.12E+02 (4.31E-01) –	7.70E+02 (5.39E+00) –	**4.52E+02** **(7.13E+01)**
F23	8.06E+02 (7.85E+01) –	7.73E+02 (2.93E+01) –	6.52E+02 (1.42E+01) –	7.54E+02 (4.19E+00) –	9.58E+02 (3.45E+01) –	8.43E+02 (2.21E+00) –	7.18E+02 (6.65E-08) –	**5.02E+02** **(2.64E+01)**
F24	2.15E+02 (2.81E+01) –	2.13E+02 (2.63E+00) –	2.01E+02 (2.74E+00) –	2.36E+02 (3.65E-10) –	2.48E+02 (1.27E-09) –	2.05E+02 (1.11E-01) –	2.12E+02 (2.94E-16) –	**2.00E+02** **(5.99E-02)**
F25	4.25E+02 (1.17E+01) –	3.99E+02 (5.01E+00) –	4.45E+02 (1.74E+01) –	4.17E+02 (4.60E+00) –	4.40E+02 (3.17E+01) –	4.11E+02 (4.44E-01) –	3.94E+02 (4.57E+00) –	**2.61E+02** **(1.21E+02)**
+	4	4	1	5	6	1	5	
≈	0	0	0	0	0	0	0	
-	21	21	24	20	19	24	20	

Table 4 Comparison of the average error (standard deviation) on 30 dimensions (Zhu et al. 2020)

	JADE	SaDE	SOUPDE	EPSDE	ESADE	MPEDE	SAMDE	SFSADE
F1	1.16E-23 (3.20E-24) –	1.57E-26 (4.24E-27) –	9.27E-06 (1.95E-05) –	4.92E-23 (1.36E-23) –	5.78E+00 (1.54E+00) –	4.27E-05 (7.22E-07) –	5.97E-24 (8.68E-25) –	**1.31E-92** **(2.41E-92)**
F2	**9.40E-03** **(7.21E-04)** +	2.02E+02 (1.96E+01) +	6.28E+02 (2.46E+02) +	1.36E+02 (2.21E+01) +	5.77E+00 (1.52E+00) +	1.51E+01 (1.30E-01) +	6.71E-01 (6.66E-02) +	1.90E+03 (5.41E+02)
F3	3.06E+05 (4.95E+03) –	9.77E+06 (1.91E+06) –	1.45E+08 v6.06E+07) –	6.50E+06 (1.20E+06) –	8.76E+05 (7.86E+03) +	**1.04E+05** **(4.54E+02)** +	1.93E+06 (1.30E+05) +	2.65E+06 (4.04E+05)
F4	**1.94E-01** **(1.86E-00)** +	3.32E+03 (4.23E+02) –	8.91E+03 (4.09E+03) –	1.44E+03 (2.54E+02) +	9.05E+02 (3.52E+01) +	2.41E+02 (2.32E+00) +	1.54E+02 (1.53E+01) +	2.73E+03 (4.11E+02)
F5	1.61E+03 (1.18E+01) +	2.71E+03 (4.16E+01) +	8.26E+03 (1.59E+03) +	1.88E+03 (6.84E+01) +	2.65E+03 (8.68E+00) +	1.58E+03 (3.54E+00) +	**1.52E+03** **(6.87E+00)** +	9.84E+03 (1.21E+03)
F6	1.16E+02 (2.11E+02) –	7.72E+01 (5.18E-01) +	3.50E+02 (7.02E+02) –	3.23E+01 (1.94E-01) +	6.76E+01 (3.54E+00) +	1.37E+02 (7.08E-01) –	**3.75E+01** **(2.27E-02)** +	8.87E+01 (1.32E+01)
F7	1.28E-02 (4.14E-06) –	2.49E-02 (3.02E-04) –	1.28E+00 (1.13E-01) –	1.39E-02 (2.65E-07) –	5.16E+00 (1.19E+00) –	5.04E-01 (3.95E-03) –	1.73E-02 (6.09E-06) –	**1.16E-02** **(1.00E-02)**
F8	2.14E+01 (7.97E-02) –	2.12E+01 (6.41E-02) –	2.12E+01 (6.26E-02) –	2.12E+01 (6.25E-02) –	2.16E+01 (1.20E-01) –	2.13E+01 (4.26E-03) –	2.12E+01 (6.54E-02) –	**2.08E+01** **(4.47E-02)**
F9	2.88E+01 (4.65E+00) –	3.47E+01 (5.16E+00) –	2.35E+01 (5.59E+00) –	1.01E+02 (1.38E+01) –	8.82E+01 (1.39E+01) –	1.09E+02 (1.12E+00) –	2.91E+01 (8.09E-01) –	**1.76E-10** **(3.52E-10)**
F10	2.94E+02 (2.40E+01) –	2.20E+02 (1.90E+01) –	2.64E+02 (3.43E+01) –	2.34E+02 (1.90E+01) –	2.14E+02 (2.51E+01) –	2.10E+02 (8.81E-01) –	2.25E+02 (1.64E+01) –	**2.01E+02** **(1.12E+01)**

Table 4 (continued)

	JADE	SaDE	SOUPDE	EPSDE	ESADE	MPEDE	SAMDE	SFSADE
F11	5.02E+01 (2.13E+00) –	4.35E+01 (2.04E+00) –	4.04E+01 (2.67E+00) –	4.26E+01 (2.20E+00) –	5.17E+01 (3.67E+00) –	4.44E+01 (1.21E-01) –	4.52E+01 (1.79E+00) –	**2.46E+01 (1.60E+00)**
F12	6.53E+04 (3.34E+04) –	1.53E+04 (1.13E+04) –	2.18E+05 (6.12E+04) –	6.57E+04 (1.55E+04) –	1.46E+04 (5.95E+03) –	9.96E+03 (3.44E+01) –	6.17E+03 (2.50E+00) –	**2.54E+00 (6.65E-01)**
F13	1.54E+01 (1.86E+00) –	1.50E+01 (1.46E+00) –	8.02E+00 (1.25E+00) –	1.40E+01 (1.76E+01) –	2.66E+01 (3.64E+00) –	1.28E+01 (1.34E-01) –	1.78E+01 (1.62E+00) –	**8.38E-01 (6.66E-02)**
F14	1.45E+01 (1.70E-01) –	1.40E+01 (1.86E-01) –	1.40E+01 (2.04E-01) –	1.40E+01 (1.86E-01) –	1.49E+01 (1.48E-01) –	1.43E+01 (1.12E-02) –	1.40E+01 (1.76E-01) –	**1.22E+01 (3.59E-01)**
F15	3.50E+02 (1.21E+01) –	3.59E+02 (2.08E+01) –	4.17E+02 (3.46E+01) –	3.88E+02 (3.23E-08) –	3.83E+02 (1.83E+00) –	3.97E+02 (5.47E-03) –	3.49E+02 (5.36E-13) –	**1.00E+00 (1.74E-01)**
F16	3.48E+02 (3.56E+01) –	2.86E+02 (2.92E+01) –	3.59E+02 (6.89E+01) –	2.94E+02 (1.89E+01) –	2.82E+02 (2.23E+01) –	2.50E+02 (1.63E+00) –	2.84E+02 (1.45E+01) –	**1.63E+02 (2.02E+01)**
F17	4.00E+02 (3.84E+01) –	3.02E+02 (3.98E+01) –	4.26E+02 (7.31E+01) –	3.38E+02 (1.88E+01) –	3.37E+02 (2.32E+01) –	2.87E+02 (1.15E+00) –	3.00E+02 (1.70E+01) –	**1.86E+02 (2.91E+01)**
F18	9.04E+02 (4.41E-01) –	9.04E+02 (5.30E-01) –	9.18E+02 (2.34E+00) –	9.04E+02 (2.76E-01) –	9.03E+02 (1.61E+00) –	9.07E+02 (1.32E-02) –	9.03E+02 (7.53E-04) –	**6.54E+02 (2.18E+02)**
F19	9.01E+02 (4.14E-01) –	9.03E+02 (2.45E+00) –	9.09E+02 (2.96E-01) –	9.19E+02 (2.30E+00) –	9.12E+02 (1.74E+00) –	9.07E+02 (1.17E-02) –	8.99E+02 (6.79E-04) –	**6.80E+02 (2.44E+02)**
F20	9.08E+02 (6.58E-01) –	9.07E+02 (5.35E-01) –	9.18E+02 (2.28E+00) –	9.10E+02 (3.34E-01) –	8.90E+02 (1.58E+00) –	9.07E+02 (1.17E-02) –	8.99E+02 (6.87E-04) –	**7.82E+02 (1.77E+02)**

Table 4 (continued)

	JADE	SaDE	SOUPDE	EPSDE	ESADE	MPEDE	SAMDE	SFSADE
F21	5.00E+02 (6.08E-13) –	5.00E+02 (3.07E-13) –	5.00E+02 (1.41E-05) –	5.16E+02 (2.67E-13) –	5.30E+02 (1.51E+00) –	5.00E+02 (2.81E-07) –	5.24E+02 (5.22E-13) –	**4.80E+02** **(3.42E+00)**
F22	9.47E+02 (8.35E+00) –	9.49E+02 (7.99E+00) –	1.11E+03 (5.68E+01) –	9.43E+02 (8.08E+00) –	9.14E+02 (7.56E+00) –	9.19E+02 (3.37E-01) –	9.29E+02 (5.49E+00) –	**5.01E+02** **(4.27E+00)**
F23	5.50E+02 (1.20E-03) –	5.50E+02 (1.89E+00) –	5.34E+02 (2.15E-03) –	5.66E+02 (1.28E-04) –	5.34E+02 (1.02E+00) –	5.38E+02 (3.14E-05) –	5.37E+02 (7.08E-13) –	**5.17E+02** **(1.13E+00)**
F24	2.00E+02 (7.61E-13) ≈	2.00E+02 (1.34E-13) ≈	2.00E+02 (1.22E-04) ≈	2.00E+02 (8.61E-13) ≈	2.05E+02 (1.40E+00) −	2.00E+02 (7.27E-07) ≈	2.00E+02 (2.21E-13) ≈	**2.00E+02** **(6.07E-02)**
F25	2.18E+02 (3.95E-01) –	2.16E+02 (3.93E-01) ≈	2.36E+02 (4.81E+00) –	2.13E+02 (2.43E-01) +	2.13E+02 (1.44E+00) +	2.13E+02 (6.04E-03) +	**2.12E+02** **(1.40E-01)** +	2.16E+02 (3.26E-01)
+	3	3	2	5	6	5	6	
≈	1	2	1	1	0	1	1	
−	21	20	22	19	19	19	18	

Table 5 Comparison of the average error (standard deviation) on 50 dimensions (Zhu et al. 2020)

	JADE	SaDE	SOUPDE	EPSDE	ESADE	MPEDE	SAMDE	SFSADE
F1	9.79E-28 (1.78E-28) –	3.35E-28 (7.07E-29) –	8.06E-06 (1.52E-05) –	5.89E-26 (1.09E-26) –	8.30E-02 (1.86E-02) –	5.27E-08 (5.79E-10) –	2.60E-18 (1.47E-19) –	**4.24E-126** **(7.76E-126)**
F2	**1.89E+00** **(7.39E-02)** +	2.34E+03 (8.22E+01) +	3.83E+03 (1.24E+03) +	2.47E+03 (2.03E+02) +	1.88E+02 (2.98E+00) +	5.09E+01 (1.63E-01) +	1.26E+02 (4.61E+00) +	7.31E+03 (3.03E+02)
F3	**8.28E+05** **(5.39E+03)** +	4.74E+06 (4.40E+04) +	3.20E+08 (1.04E+08) –	1.12E+07 (8.54E+05) –	2.57E+06 (7.23E+03) +	9.69E+05 (6.64E+02) +	2.80E+06 (5.25E+04) +	7.13E+06 (1.04E+06)
F4	6.95E+03 (2.75E+02) +	2.14E+04 (1.73E+03) –	9.34E+04 (3.49E+04) –	1.87E+04 (2.32E+03) –	1.38E+04 (3.14E+02) –	**5.89E+03** **(1.73E+01)** +	6.55E+03 (4.20E+02) +	9.35E+03 (1.25E+01)
F5	4.61E+03 (6.57E+00) +	6.08E+03 (2.81E+01) +	2.33E+04 (3.77E+03) –	4.45E+03 (4.76E+01) +	6.90E+03 (1.48E+00) +	**3.43E+03** **(1.44E+00)** +	4.35E+03 (6.45E+00) +	1.86E+04 (2.07E+03)
F6	8.45E+01 (4.13E-01) +	1.15E+02 (1.13E-01) +	3.80E+02 (7.28E+02) –	6.54E+01 (9.62E-02) +	1.27E+02 (3.14E+00) +	**1.07E+02** **(1.40E-01)** +	1.60E+02 (1.20E-01) +	1.99E+02 (1.17E+01)
F7	**6.06E-03 (5.83E-05)** +	2.99E-02 (4.42E-04) +	1.05E+00 (1.87E-02) –	7.00E-03 (1.12E-05) +	1.09E+00 (2.13E-02) –	5.63E-02 (2.72E-04) –	2.50E-02 (2.38E-04) +	5.57E-02 (3.63E-02)
F8	2.15E+01 (5.83E-02) –	2.13E+01 (4.45E-02) –	2.13E+01 (4.70E-02) –	2.13E+01 (4.38E-02) –	2.15E+01 (8.91E-02) –	2.14E+01 (1.85E-03) –	2.13E+01 (4.56E-02) –	**2.09E+01** **(4.86E-02)**
F9	1.03E+02 (1.11E+01) –	5.90E+01 (4.14E+00) –	4.36E+01 (7.42E+00) –	2.24E+02 (2.07E+01) –	2.08E+01 (3.60E+00) –	1.68E+02 (1.28E+00) –	5.14E+01 (1.80E-08) –	**1.93E-12** **(3.68E-12)**
F10	5.02E+02 (3.33E+01) –	4.08E+02 (2.72E+01) –	5.12E+02 (6.17E+01) –	4.35E+02 (2.63E+01) –	2.54E+02 (3.29E+01) –	3.09E+02 (2.09E+00) –	4.14E+02 (2.54E+01) –	**2.70E+02** **(4.61E+01)**

Table 5 (continued)

	JADE	SaDE	SOUPDE	EPSDE	ESADE	MPEDE	SAMDE	SFSADE
F11	8.71E+01 (2.77E+00) –	7.74E+01 (2.69E+00) –	7.17E+01 (3.85E+00) –	7.69E+01 (2.77E+00) –	7.29E+01 (3.96E+00) –	7.63E+01 (1.60E-01 –	8.02E+01 (2.34E+00) –	**5.27E+01 (3.14E-01)**
F12	2.36E+04 (9.63E+02) –	3.53E+04 (3.58E+01) –	6.44E+05 (1.44E+05) –	2.32E+04 (5.43E+01) –	4.50E+04 (3.97E+01) –	3.69E+04 (1.89E+01) –	3.48E+04 (7.99E+00) –	**1.09E+01 (3.76E+00)**
F13	2.65E+01 (2.52E+00) –	2.90E+01 (2.11E+00) –	1.34E+01 (1.65E+00) –	2.76E+01 (2.39E+00) –	1.26E+01 (1.59E+00) –	2.01E+01 (1.22E-01) –	2.00E+01 (1.45E+00) –	**1.44E+00 (6.03E-02)**
F14	2.44E+01 (1.91E-01) –	2.38E+01 (2.02E-01) –	2.38E+01 (2.47E-01) –	2.38E+01 (2.09E-01) –	2.41E+01 (2.26E-01) –	2.40E+01 (8.48E-03) –	2.38E+01 (1.83E-01) –	**2.19E+01 (2.39E-01)**
F15	3.29E+02 (2.62E+00) –	3.73E+02 (1.38E+01) –	3.55E+02 (6.83E+01) –	3.42E+02 (2.45E-13) –	3.42E+02 (9.09E-02) –	3.52E+02 (1.52E-04) –	3.21E+02 (1.14E-11) –	**1.84E+00 (6.14E-01)**
F16	3.62E+02 (2.19E+01) –	2.31E+02 (2.35E+01) –	4.45E+02 (4.95E+01) –	3.36E+02 (1.50E+01) –	2.76E+02 (1.49E+01) –	2.16E+02 (7.57E-01) –	2.89E+02 (1.59E+01) –	**1.96E+02 (3.17E+00)**
F17	4.21E+02 (2.83E+01) –	3.43E+02 (1.97E+01) –	5.71E+02 (8.49E+01) –	3.70E+02 (1.54E+01) –	3.62E+02 (2.19E+01) –	2.93E+02 (1.02E+00) –	3.67E+02 (1.12E+01) –	**2.72E+02 (1.92E+00)**
F18	9.32E+02 (2.24E+00) –	9.42E+02 (1.37E+00) –	9.76E+02 (1.67E+01) –	9.35E+02 (8.39E-01) –	7.87E+02 (2.15E-01) –	9.18E+02 (5.90E-03) –	9.30E+02 (3.12E-05) –	**7.70E+02 (4.62E-01)**
F19	9.36E+02 (1.51E+00) –	9.45E+02 (2.22E+00) –	9.62E+02 (6.94E+00) –	9.29E+02 (3.21E-01) –	7.44E+02 (2.24E-01) –	9.30E+02 (4.11E-03) –	9.27E+02 (8.08E-05) –	**7.36E+02 (2.12E-01)**
F20	9.28E+02 (1.12E+00) –	9.43E+02 (1.10E+00) –	9.77E+02 (1.72E+01) –	9.26E+02 (5.52E-01) –	8.09E+02 (2.51E-01) –	9.17E+02 (7.15E-03) –	9.32E+02 (1.89E-05) –	**8.09E+02 (3.64E-01)**

Table 5 (continued)

	JADE	SaDE	SOUPDE	EPSDE	ESADE	MPEDE	SAMDE	SFSADE
F21	5.40E+02 (1.47E-12) –	5.00E+02 (4.93E-13) –	5.00E+02 (5.35E-06) –	5.00E+02 (4.03E-13) –	5.40E+02 (1.89E-02) –	5.61E+02 (8.27E-04) –	5.39E+02 (1.07E-10) –	**4.99E+02 (3.72E-02)**
F22	1.00E+03 (7.80E+00) –	9.88E+02 (5.50E+00) –	1.09E+03 (3.00E+01) –	9.76E+02 (6.04E+00) –	9.49E+02 (6.64E+00) –	9.34E+02 (1.65E-01) –	9.64E+02 (4.29E+00) –	**7.11E+02 (2.57E+00)**
F23	5.95E+02 (2.28E-01) –	5.67E+02 (4.23E-02) –	5.39E+02 (1.75E-02) –	5.42E+02 (6.14E-08) –	5.67E+02 (1.21E-02) –	5.58E+02 (1.61E-04) –	5.83E+02 (2.02E-09) –	**5.35E+02 (2.13E+00)**
F24	8.80E+02 (8.63E+00) –	2.00E+02 (2.62E-11) ≈	2.00E+02 (9.40E-03) ≈	2.00E+02 (1.05E-13) ≈	2.00E+02 (2.03E-02) ≈	2.00E+02 (2.64E-09) ≈	2.00E+02 (2.86E-13) ≈	**2.00E+02 (6.01E-03)**
F25	4.92E+02 (3.82E+00) –	2.43E+02 (9.72E-01) –	3.31E+02 (1.33E+01) –	2.24E+02 (2.46E-01) –	2.23E+02 (1.35E-01) –	2.28E+02 (6.83E-03) –	2.22E+02 (1.46E-01) –	**2.07E+02 (7.14E-02)**
+	6	5	1	4	4	5	6	
≈	0	1	1	1	1	1	1	
–	19	19	23	20	20	19	18	

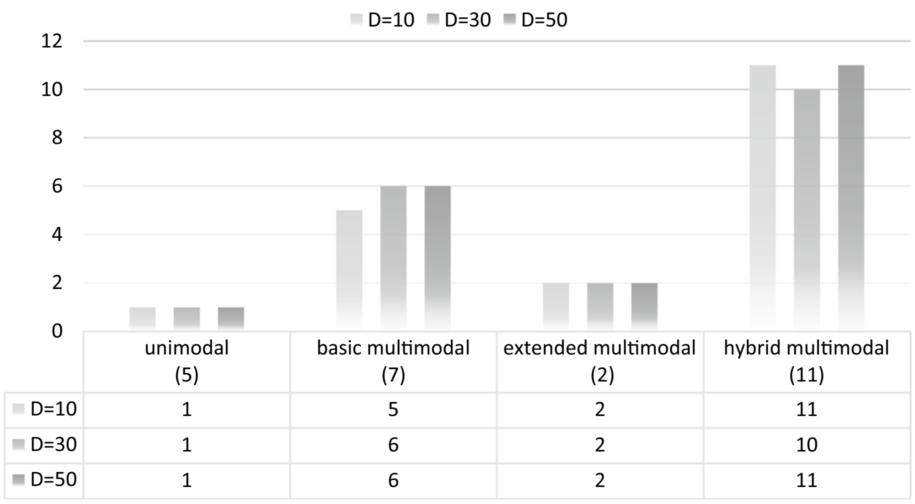
$ps = 0.5$, $pu = 0.5$, $CR = 0.9$; EPSDE with $NP = 50$; ESADE with $NP = 50$, $t_0 = 1000$; MPEDE with $NP = 250$, $ng = 20$, $\lambda_1 = \lambda_2 = \lambda_3 = 0.2$;

We have performed 10-, 30-, and 50-dimensional optimization problems on 25 benchmark functions, and each test function performs 20 independent experiments on the corresponding dimension, calculates its average error and corresponding standard deviation as the final result, and records them in Tables 3, 4 and 5. The symbols "+", "≈", and "−" in each table indicate that the performance of the corresponding algorithm is better, similar, or worse than SFSADE. For each test function, the optimal value found in all enhanced DE variants is highlighted in bold.

Tables 3, 4, 5 above respectively show the overall performance of SFSADE and the other 7 advanced DE variants on 25 test functions of 10, 30, and 50 dimensions. First of all, for unimodal functions F1 − F5, when solving the optimization problem F1, SFSADE performs optimally in three dimensions, and its performance is far superior to all other DE variants; on F2 − F5, the performance of SFSADE is at a general level, only better than some DE variants. Especially for F5, the optimization results are poor. Secondly, For the basic multimodal functions F6 − F12, SFSADE is only at a disadvantage in F6, and for other functions, it is almost all superior to other DE variants in three dimensions. Finally, whether it is the extended multimodal functions F13 − F14 or the hybrid multimodal functions F15 − F25, compared with other DE variants, the performance of SFSADE is still far ahead. Only for the two functions F24 and F25, SFSADE and other DE variants achieve similar results, or it is at an intermediate level.

In addition, for each dimension, we have calculated the number of times that SFSA outperforms JADE, SaDE, SOUPDE, EPSDE, ESADE, MPEDE, and SAMDE on 25 test functions. On 10-dimensional functions, it is 21, 21, 24, 20, 19, 24, 20, defeating other opponents 21.29 times on average. Suffice it to say that the overall performance of SFSADE is better than the other 7 advanced DE variants, and it is highly competitive. On 30 dimensions, the times are 21, 20, 22, 19, 19, 19, and 18 respectively. Although the average number of wins against other opponents has been reduced by 19.71, the overall performance of SFSADE is still better than other DE variants. On the 50th dimension, the times are 19, 19, 23, 20, 20, 19, and 18 respectively, and the average number of times of defeating other opponents is the same as on the 30th dimension, both are 19.71. It can also be seen that the overall performance of SFSADE is still far better than the other 7 advanced DE variants.

Through all the above optimization experiments, we can clearly see that SFSADE has competitive optimization results under different dimensions and types of test functions, and its results have great advantages over other algorithms. For further visual comparison, we quantitatively summarize these results in Fig. 4 according to four types and three dimensions. Among the 5 unimodal functions, one function has the best performance; among the 7 basic multimodal functions, 5, 6, and 6 optimal performances are achieved in the three dimensions respectively; on the 2 extended multimodal functions, SFSADE all reaches the best; for the 11 hybrid multimodal functions, it defeats almost all opponents in three dimensions. Obviously, our improved algorithm SFSADE is very successful and has strong comprehensive performance. Especially for high-dimensional hybrid multimodal functions that are extremely difficult to solve, SFSADE can still obtain the most accurate results when comparing with the other 7 algorithms, which further reflects its powerful optimization capabilities.



	unimodal (5)	basic multimodal (7)	extended multimodal (2)	hybrid multimodal (11)
D=10	1	5	2	11
D=30	1	6	2	10
D=50	1	6	2	11

Fig. 4 Situation where SFSADE is better than others under different dimensions and types of test functions

5.3 Statistical test

To further analyze the experimental results, we use the Wilcoxon signed ranks test and Friedman test, two nonparametric statistical tests, to evaluate the performance of the above algorithms (Mirsadeghi and Khodayifar 2021). Therefore, we consider to propose two different hypotheses and compare the proposed algorithm SFSADE with the other DE variants, namely JADE, SaDE, SOUPDE, EPSDE, ESADE, MPEDE, and SAMDE algorithms:

(a) H0: There is no significantly difference between the proposed algorithm and the other variant algorithms;
(b) H1: There is a significantly difference between the proposed algorithm and the other variant algorithms.

The Wilcoxon signed ranks test involves two samples and is used to detect the significant differences between the two samples (Derrac et al. 2011). Therefore, it is used for pairwise comparisons of the algorithms. Combining the optimization results of the average error in Tables 3, 4, 5, we perform Wilcoxon signed ranks test and show it in Table 6. In this table, R^+ represents the sum of ranks that our proposed algorithm is better than the second algorithm, and R^- represents the sum of ranks in the opposite case, and satisfies the equation $R^+ + R^- = n(n+1)/2$. From the p-value we calculate for each test function in each dimension, it can be seen that SFSADE shows a significant improvement over the other 7 DE variants with the significance levels $\alpha = 0.05$. Therefore, we reject the null hypothesis H0, which proves that there is a significant difference in the performance comparison between our proposed algorithm and the other 7 DE variants.

The Friedman test is a nonparametric simulation method of two-way analysis of variance. Through multiple comparisons, significant differences between different

Table 6 The Wilcoxon signed ranks test results

Dimension	Comparison	R^+	R^-	p-value
10	SAMDE versus JADE	270.0	55.0	0.003821
	SAMDE versus SaDE	285.0	40.0	0.000980
	SAMDE versus SOUPDE	302.0	23.0	0.000174
	SAMDE versus EPSDE	259.0	66.0	0.009417
	SAMDE versus ESADE	253.0	72.0	0.014889
	SAMDE versus MPEDE	303.0	22.0	0.000157
	SAMDE versus SAMDE	258.0	67.0	0.010181
30	SAMDE versus JADE	235.5	91.5	0.071861
	SAMDE versus SaDE	271.5	53.5	0.005139
	SAMDE versus SOUPDE	281.5	43.5	0.001843
	SAMDE versus EPSDE	240.5	84.5	0.042502
	SAMDE versus ESADE	222.0	103.0	0.109385
	SAMDE versus MPEDE	227.5	97.5	0.097490
	SAMDE versus SAMDE	213.5	111.5	0.198543
50	SAMDE versus JADE	222.0	103.0	0.109386
	SAMDE versus SaDE	238.5	86.5	0.048675
	SAMDE versus SOUPDE	303.5	21.5	0.000203
	SAMDE versus EPSDE	264.5	60.5	0.007235
	SAMDE versus ESADE	224.5	100.5	0.124528
	SAMDE versus MPEDE	219.5	105.5	0.153106
	SAMDE versus SAMDE	221.5	103.5	0.129953

algorithms are found (Derrac et al. 2011). We first sort the 7 algorithms to be compared for the results of the 25 test functions and then examine the differences across columns to compare the overall performance of each algorithm. The results of the Friedman test are shown in Table 7. We write the corresponding general rank according to the Friedman rank, which represents the general performance of all of the algorithms on these functions. It can be seen from the ranking results that the performance of all algorithms is ranked in any of the 10 dimensions, 30 dimensions, and 50 dimensions. The algorithm SFSADE we proposed ranks first, far surpassing other algorithms. This once again shows that SFSADE has a strong comprehensive performance.

5.4 Parameter analysis

Taken together, the performance of SFSADE is far superior to that of other advanced DE variants under the same experimental conditions. This strongly proves that the three key improvement strategies we proposed, namely, the grouping evolution strategy, the classification mutation strategy, and the self-adaptive control parameter adjustment strategy, are all very effective. However, the poor test results on the unimodal functions F2 – F5 are still due to inappropriate initial setting parameters (Zhu et al. 2020). In the above experimental process, the range of control parameters, the ratio of population evolution number and group evolution number, and the number of meme groups will all affect the final result. We tested the sensitivity of SFSADE to changes in these important parameters through

Table 7 The Friedman test results

Dimension	Algorithm	Friedman rank	General rank
10	JADE	5.34	6
	SaDE	3.98	4
	SOUPDE	5.08	5
	EPSDE	3.70	3
	ESADE	6.00	7
	MPEDE	6.94	8
	SAMDE	2.92	2
	SFSADE	2.04	1
30	JADE	4.84	5
	SaDE	4.60	4
	SOUPDE	5.84	8
	EPSDE	5.04	6
	ESADE	5.12	7
	MPEDE	4.48	3
	SAMDE	3.62	2
	SFSADE	2.46	1
50	JADE	5.24	7
	SaDE	5.06	6
	SOUPDE	6.20	8
	EPSDE	4.54	5
	ESADE	4.30	4
	MPEDE	4.08	2
	SAMDE	4.12	3
	SFSADE	2.46	1

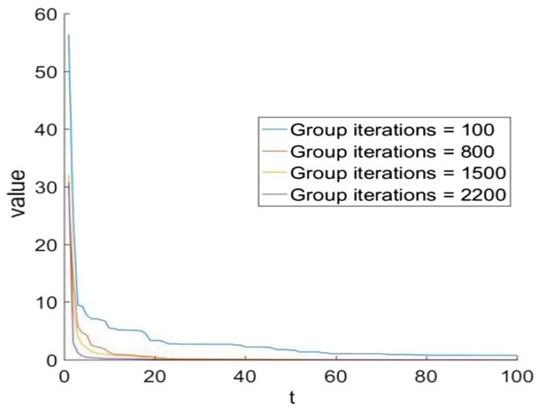


Fig. 5 The effect of the number of group iterations on the results

experiments. This further shows that SFSADE still has good performance for solving unimodal functions.

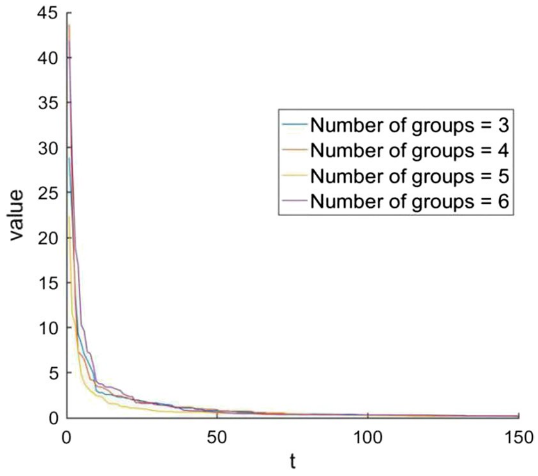


Fig. 6 The effect of the number of meme groups on the results

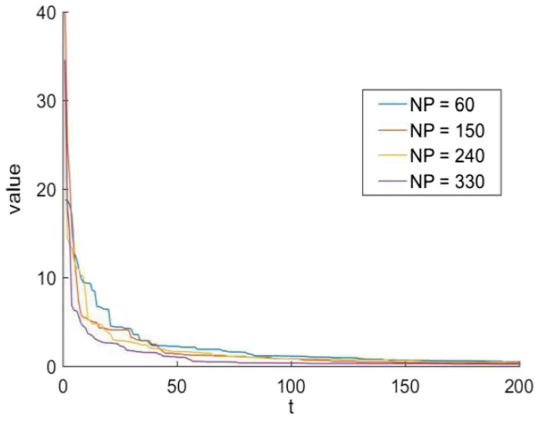


Fig. 7 The effect of the population size on the results

We take the experiment on the unimodal function F2 on 10 dimensions as an example, and the range of the self-adaptive control parameters of mutation and crossover operation is still $[0.1, 0.9]$.

First, we conducted an experiment on the sensitivity of the parameter and the number of function evaluations. It is $2000 \times D$ in the original experiment, which is equal to the population evolution number multiplied by the group evolution number. In this experiment, the number of meme groups is fixed at 4, the number of populations is fixed at 60, and the population evolution iterations are fixed at 200. The number of grouping iterations is set to 100, 800, 1500, and 2200 to increase the number of function evaluations. Figure 5 shows the experimental results. The global optimal solutions corresponding to the four different grouping iterations are 5.82E – 01, 2.31E-04, 3.58E-08, and 7.14E – 10. Therefore, within a certain range, as the number of grouping iterations increases, the convergence speed of the algorithm gradually increases, and the global optimal solution to be solved is increasingly better than the best result in Table 2.

Then, we experimented with the sensitivity of the parameter of the number of meme groups. In this experiment, the population number is fixed at 60, the population evolution iterations are fixed at 150, the number of grouping iterations is 300, and the number of meme groups is set to 3, 4, 5, and 6. Figure 6 shows the experimental results. The global optimal solutions corresponding to the four different numbers of meme groups are, 1.37E-01, 1.10E-04, 1.66E-08, and 1.53E-10. It can be seen that the number of meme groups is different, and the convergence speed of the algorithm and the global optimal solution of the solution are also different. Therefore, it is particularly important to select a more appropriate number of meme groups during the experiment.

Finally, we experimented on the sensitivity of the population size parameter. In this experiment, the number of meme groups is fixed at 3, and the number of function evaluations is fixed at $2000 \times D$, where the population evolution iterations is 200, the group iterations is 100, and the population numbers are set to 60, 120, 180, and 240. Figure 7 shows the experimental results. The global optimal solutions corresponding to the four different population sizes are 4.91E-01, 3.03E-03, 6.28E-07, and 1.63E − 09. Therefore, within a certain range, as the number of populations increases, the convergence speed of the algorithm gradually increases, and the global optimal solution to be solved is also better, which is also better than the best result in Table 2.

Through the above experiments, it can be seen that the setting of the experimental parameters has a great influence on the results. When optimizing the unimodal function, we can actively adjust the relevant experimental parameters to find a better global optimal solution. In particular, the two parameters that can be manually adjusted, the maximum evaluation number of the function and the population size, have significantly improved the solution effect of SFSADE. It also shows that our comprehensive improvement strategy for DE is very effective.

6 Conclusion

In order to improve the performance of DE algorithm in solving numerical optimization problems, this paper proposes an improved adaptive differential evolution algorithm SFSADE based on a shuffled frog-leaping strategy. In theory and experiment, we have verified the innovation and improvement in the algorithm from three different aspects: First, we have successfully integrated the shuffled frog-leaping algorithm, and divided the population into multiple sub-populations randomly based on the fitness value of the individual. Each sub-population does not affect independent evolution. After each generation is updated, these sub-populations reintegrate into a population, and carry out information transmission and exchange, so as to fully improve the diversity and convergence speed of the population. Second, in the process of evolution, we adopt a new classification mutation strategy designed, that is, individuals with different fitness values are affected by the dual effects of the global optimal individual and the grouped optimal individual to further enhance the adaptability and diversity of the population. Third, we introduce a new adaptive control parameter adjustment mechanism designed, that is, each generation will automatically adjust the control parameters related to mutation and crossover operations according to the evolutionary number and individual fitness values, which further improves the performance of the algorithm.

In addition, this paper uses the CEC2005 benchmark functions to carry out a large number of experiments, and conducts two non-parametric statistical tests, the Wilcoxon signed ranks test and Friedman test, which are compared with the other 7 most advanced

DE variants (JADE, SaDE, SOUPDE, EPSDE, ESADE, MPEDE, and SAMDE) to conduct a comprehensive comparison and analysis. Through the results, it is proved that SFSADE has a strong comprehensive performance, which is far superior to other DE variants. At the same time, we also analyzed the sensitivity of SFSADE to related experimental parameters, further showing that compared with other DE variants, the three innovative improvement strategies of SFSADE have obvious advantages and competitiveness.

The good performance of SFSADE can try to solve optimization problems in many fields. For example, for the optimization model in multithreshold image segmentation, the algorithm may improve the segmentation effect and increase the segmentation speed; in the UAV trajectory planning problem, the algorithm may be able to overcome complex constraints and improve the quality of trajectory planning. In the future, we will further improve SFSADE, try to address numerical optimization problems of higher dimensions or different benchmark functions under time constraints, and hope to further solve other specific optimization problems in the real world.

Acknowledgements The authors are very grateful to the reviewers for their beneficial suggestions. This work is supported by the National Natural Science Foundation Project of China (Grant No. 62073330), Young Talents Lifting Project of China (Grant No. 17JCJQQT048), Natural Science Foundation Project of Hunan Province of China (Grant No. 2019JJ20021) and NUDT Scientific Research Project of China (Grant No. ZK18-02-12).

Declarations

Conflict of interest The authors declare that they have no conflicts of interest about this paper.

References

Bi X, Xiao J (2012) Classification-based self-adaptive differential evolution and its application in multilateral multi-issue negotiation. Front Comp Sci 6:442–461

Brest J, Greiner S, Boskovic B et al (2006) Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. IEEE Trans Evol Comput 10:646–657. https://doi.org/10.1109/TEVC.2006.872133

Cai Y, Wang J (2015) Differential evolution with hybrid linkage crossover. Inf Sci 320:244–287. https://doi.org/10.1016/j.ins.2015.05.026

Cai X, Zhao H, Shang S et al (2021) An improved quantum-inspired cooperative co-evolution algorithm with muli-strategy and its application. Expert Syst Appl 171:114629. https://doi.org/10.1016/j.eswa.2021.114629

Chaudhary D, Tailor AK, Sharma VP, Chaturvedi S (2019) HyGADE: Hybrid of Genetic Algorithm and Differential Evolution Algorithm. In: 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT). IEEE, pp 1–4

Chebyshev PL (1867) Des Valeurs Moyennes Liouville's JMathPures Appl 12:177–184

Colorni A, Dorigo M, Maniezzo V (1991) Distributed Optimization by ant colonies. In: Proceedings of the First European Conference on Artificial Life. pp 134–142

Das S, Konar A (2009) Automatic image pixel clustering with an improved differential evolution. Appl Soft Comput 9:226–236. https://doi.org/10.1016/j.asoc.2007.12.008

Das S, Suganthan PN (2011) Differential Evolution: A Survey of the State-of-the-Art. IEEE Trans Evol Comput 15:4–31. https://doi.org/10.1109/TEVC.2010.2059031

Das S, Konar A, Chakraborty UK (2007) Annealed Differential Evolution. In: 2007 IEEE Congress on Evolutionary Computation. IEEE, pp 1926–1933

Deng W, Xu J, Gao X-Z, Zhao H (2020a) An Enhanced MSIQDE Algorithm With Novel Multiple Strategies for Global Optimization Problems. IEEE Trans Syst Man Cybern, Syst. https://doi.org/10.1109/TSMC.2020.3030792

Deng W, Xu J, Zhao H, Song Y (2020b) A Novel Gate Resource Allocation Method Using Improved PSO-Based QEA. IEEE Trans Intell Transport Syst. https://doi.org/10.1109/TITS.2020.3025796

Deng W, Shang S, Cai X et al (2021) Quantum differential evolution with cooperative coevolution framework and hybrid mutation strategy for large scale optimization. Knowl-Based Syst 224:107080. https://doi.org/10.1016/j.knosys.2021.107080

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 Evol Comput 1:3–18. https://doi.org/10.1016/j.swevo.2011.02.002

Elsayed SM, Sarker RA, Essam DL (2013) Self-adaptive differential evolution incorporating a heuristic mixing of operators. Comput Optim Appl 54:771–790. https://doi.org/10.1007/s10589-012-9493-8

Eusuff MM, Lansey KE (2003) Optimization of Water Distribution Network Design Using the Shuffled Frog Leaping Algorithm. J Water Resour Plan Manag 129:210–225. https://doi.org/10.1061/(ASCE)0733-9496(2003)129:3(210)

Fan HY, Lampinen J (2003) A trigonometric mutation operation to differential evolution. J Global Optim 27:105–129. https://doi.org/10.1023/A:1024653025686

Fan Q, Yan X (2015) Self-adaptive differential evolution algorithm with discrete mutation control parameters. Expert Syst Appl 42:1551–1572. https://doi.org/10.1016/j.eswa.2014.09.046

Fan Q, Wang W, Yan X (2019) Differential evolution algorithm with strategy adaptation and knowledge-based control parameters. Artif Intell Rev 51:219–253. https://doi.org/10.1007/s10462-017-9562-6

Gämperle R, Müller SD, Koumoutsakos P (2002) A parameter study for differential evolution. In: Proceedings of WSEAS international conference on advances in intelligent systems, fuzzy systems and e-computing. pp 293–298

Guanghui L, Zaiwen W, Ya-xiang Y, Qichao W (2020) Complexity analysis for optimization methods. Sci Sin-Math 50:1271. https://doi.org/10.1360/N012018-00251

Guo S, Shieh LS, Chen G, Coleman NP (2001) Observer-type Kalman innovation filter for uncertain linear systems. IEEE Trans Aerosp Electron Syst 37:1406–1418. https://doi.org/10.1109/7.976975

Guo H, Li Y, Li J et al (2014) Differential evolution improved with self-adaptive control parameters based on simulated annealing. Swarm Evol Comput 19:52–67. https://doi.org/10.1016/j.swevo.2014.07.001

Hsu H-P, Yang S-W (2020) Optimization of Component Sequencing and Feeder Assignment for a Chip Shooter Machine Using Shuffled Frog-Leaping Algorithm. IEEE Trans Automat Sci Eng 17:56–71. https://doi.org/10.1109/TASE.2019.2916925

Kennedy J, Eberhart R (1995) Particle swarm optimization. In: IEEE International Conference on Neural Networks - Conference Proceedings. p 1942—1948

Koloseni D, Lampinen J, Luukka P (2012) Optimized distance metrics for differential evolution based nearest prototype classifier. Expert Syst Appl 39:10564–10570. https://doi.org/10.1016/j.eswa.2012.02.144

Li Y, Wang S (2020) Differential evolution algorithm with elite archive and mutation strategies collaboration. Artif Intell Rev 53:4005–4050. https://doi.org/10.1007/s10462-019-09786-5

Li Y, Wang S, Yang B (2020a) An improved differential evolution algorithm with dual mutation strategies collaboration. Expert Syst Appl. https://doi.org/10.1016/j.eswa.2020.113451

Liu J, Lampinen J (2005) A Fuzzy Adaptive Differential Evolution Algorithm. Soft Comput 9:448–462. https://doi.org/10.1007/s00500-004-0363-x

Ma Y, Bai Y, JIANG Z, (2009) Fast Multi-objective Constrained Evolutionary Algorithm and Its Convergence. Syst Eng Theory Pract 29:149–157. https://doi.org/10.1016/S1874-8651(10)60050-6

Mallipeddi R, Suganthan PN, Pan QK, Tasgetiren MF (2011) Differential evolution algorithm with ensemble of parameters and mutation strategies. Appl Soft Comput 11:1679–1696. https://doi.org/10.1016/j.asoc.2010.04.024

Mao B, Xie Z, Wang Y et al (2017) A hybrid differential evolution and particle swarm optimization algorithm for numerical kinematics solution of remote maintenance manipulators. Fusion Eng Des 124:587–590

Maulik U, Saha I (2009) Modified differential evolution based fuzzy clustering for pixel classification in remote sensing imagery. Pattern Recogn 42:2135–2149. https://doi.org/10.1016/j.patcog.2009.01.011

Mirsadeghi E, Khodayifar S (2021) Hybridizing particle swarm optimization with simulated annealing and differential evolution. Cluster Comput 24:1135–1163. https://doi.org/10.1007/s10586-020-03179-y

Moscato P (1989) On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts - Towards Memetic Algorithms. Caltech Concurrent Computation Program

Pan Q-K, Suganthan PN, Wang L et al (2011) A differential evolution algorithm with self-adapting strategy and control parameters. Comput Oper Res 38:394–408. https://doi.org/10.1016/j.cor.2010.06.007

Pan Z, Liang H, Gao Z, Gao J (2018) Differential evolution with subpopulations for high-dimensional seismic inversion. Geophys Prospect 66:1060–1069. https://doi.org/10.1111/1365-2478.12620

Parouha RP, Verma P (2021) Design and applications of an advanced hybrid meta-heuristic algorithm for optimization problems. Artif Intell Rev. https://doi.org/10.1007/s10462-021-09962-6

Qin AK, Huang VL, Suganthan PN (2009a) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evol Comput 13:398–417. https://doi.org/10.1109/TEVC.2008.927706

Rahimi-Vahed A, Mirzaei AH (2007) A hybrid multi-objective shuffled frog-leaping algorithm for a mixed-model assembly line sequencing problem. Comput Ind Eng 53:642–666. https://doi.org/10.1016/j.cie.2007.06.007

Sakr WS, EL-Sehiemy RA, Azmy AM, (2017) Adaptive differential evolution algorithm for efficient reactive power management. Appl Soft Comput 53:336–351. https://doi.org/10.1016/j.asoc.2017.01.004

Sampson JR (1976) Adaptation in Natural and Artificial Systems (John H. Holland). SIAM Rev 18:529–530. https://doi.org/10.1137/1018105

Santander-Jimenez S, Vega-Rodriguez MA, Sousa L (2018) Multiobjective Frog-Leaping Optimization for the Study of Ancestral Relationships in Protein Data. IEEE Trans Evol Computat 22:879–893. https://doi.org/10.1109/TEVC.2017.2774599

Storn R, Price K (1997) Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. J Global Optim 11:341–359. https://doi.org/10.1023/A:1008202821328

Su Q, Huang Z, Hu Z, Wang X (2012) Binarization algorithm based on differential evolution algorithm for gray images. In: 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery. IEEE, pp 2611–2615

Suganthan PN, Hansen N, Liang JJ, et al (2005) Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Nanyang Technological University Technical Report

Sun J, Zhang Q, Tsang EPK (2005) DE/EDA: A new evolutionary algorithm for global optimization. Inf Sci 169:249–262

Sun J, Tang J, Lao S (2017) Collision Avoidance for Cooperative UAVs With Optimized Artificial Potential Field Algorithm. IEEE Access 5:18382–18390. https://doi.org/10.1109/ACCESS.2017.2746752

Tang J (2019) Conflict Detection and Resolution for Civil Aviation: A Literature Survey. IEEE Aerosp Electron Syst Mag 34:20–35. https://doi.org/10.1109/MAES.2019.2914986

Tang J, Piera MA, Guasch T (2016) Coloured Petri net-based traffic collision avoidance system encounter model for the analysis of potential induced collisions. Transportation Research Part C: Emerging Technologies 67:357–377. https://doi.org/10.1016/j.trc.2016.03.001

Wang Y, Cai Z, Zhang Q (2011) Differential evolution with composite trial vector generation strategies and control parameters. IEEE Trans Evol Comput 15:55–56. https://doi.org/10.1109/TEVC.2010.2087271

Wang H, Zhen X, Tu X (2019) SFDE: Shuffled frog-leaping differential evolution and its application on cognitive radio throughput. Wirel Commun Mob Comput. https://doi.org/10.1155/2019/2965061

Weber M, Neri F, Tirronen V (2011) Shuffle or update parallel differential evolution for large-scale optimization. Soft Comput 15:2089–2107. https://doi.org/10.1007/s00500-010-0640-9

Wu G, Mallipeddi R, Suganthan PN et al (2016) Differential evolution with multi-population based ensemble of mutation strategies. Inf Sci 329:329–345. https://doi.org/10.1016/j.ins.2015.09.009

Zhang X, Duan H (2015) An improved constrained differential evolution algorithm for unmanned aerial vehicle global route planning. Appl Soft Comput 26:270–284. https://doi.org/10.1016/j.asoc.2014.09.046

Zhang J, Sanderson AC (2009) JADE: Adaptive Differential Evolution With Optional External Archive. IEEE Trans Evol Comput 13:945–958. https://doi.org/10.1109/TEVC.2009.2014613

Zhao J, Xu Y, Luo F et al (2014) Power system fault diagnosis based on history driven differential evolution and stochastic time domain simulation. Inf Sci 275:13–29. https://doi.org/10.1016/j.ins.2014.02.039

Zhu L, Ma Y, Bai Y (2020) A self-adaptive multi-population differential evolution algorithm. Nat Comput 19:211–235. https://doi.org/10.1007/s11047-019-09757-3

Zou D, Wu J, Gao L, Li S (2013) A modified differential evolution algorithm for unconstrained optimization problems. Neurocomputing 120:469–481. https://doi.org/10.1016/j.neucom.2013.04.036