A fitness-based adaptive differential evolution algorithm

Xuewen Xia*[a], Ling Gui[a], Yinglong Zhang*[a], Xing Xu[a], Fei Yu[a], Hongrun Wu[a], Bo Wei[b], Guoliang He[c], Yuanxiang Li[c]

[a]*College of Physics and Information Engineering, Minnan Normal University, Zhangzhou, China*
[b]*School of Informatics Science and Technology, Zhejiang Sci-Tech University, Hangzhou, China*
[c]*School of Computer, Wuhan University, Wuhan, China*

Abstract

The performance of differential evolution algorithm (DE) mainly depends on its breeding offspring strategy (i.e., trial vector generation strategies and associated control parameters). To take full advantage of several effective breeding offspring strategies proposed in recent years, a fitness-based adaptive differential evolution algorithm (FADE) is proposed in this paper. In FADE, the entire population is split into multiple small-sized swarms, and three popular breeding strategies are saved in an archive. In each generation, different individuals in the same swarm adaptively select their own breeding strategy from the archive based on their fitness. With the adaptive breeding strategy, the individuals in the same swarm can exhibit distinct search behaviors. Moreover, the population size can be adaptively adjusted during the evolutionary process according to the performance of the best individual. Based on the adaptive population size, computational resources can be rationally assigned in different evolutionary stages, and then to satisfy diverse requirements of different fitness landscapes. The comprehensive performance of FADE is extensively evaluated by comparisons between it and other eight state-of-art DE variants based on CEC2013 and CEC2017 test suites as well as seven real applications. In addition, the effectiveness and efficiency of the newly introduced adaptive strategies are further

*Corresponding authors: Yinglong Zhang, Xuewen Xia, Email addresses: Zhang_yinglong@126.com, xwxia@whu.edu.cn

confirmed by a set of experiments.

Keywords: Differential evolution, Fitness-based, Multiple adaptive strategies, Breeding offspring strategy, Population sized

1. Introduction

DE [29] proposed by Storn and Price in 1997 is an efficient evolutionary algorithm (EA) for global optimization problems in the continuous domain, and the promising effectiveness and efficiency of DE have been testified by numerous scientific problems and technology applications [14, 24]. In DE, each individual in a population commonly denoted as a vector can be regarded as a candidate solution to a problem. In each generation, individuals in the population breed their offspring by three general evolutionary operators, i.e., the mutation, crossover, and selection operators. Thus, different DE variants have been proposed during the last decades by designing various promising operators. Extensive experiments have manifested that different operators display their own distinct characteristics and preferences [9, 20, 21, 45]. Nevertheless, it is extremely difficult, if not impossible, to predefine a set of operators and involved parameters in advance because many real applications are “black-box” problems. Moreover, it is usually a time-consuming process that applying a trial-and-error scheme to determine the most favorable operators and associated parameters. In fact, different fitness landscapes in a problem have distinct properties which require different search strategies. Hence, many researchers pour much attention into (self-)adaptive mechanisms in DE to satisfy various requirements in the different fitness landscapes [3, 17].

Due to that a promising adaptive mechanism only takes advantage of historical experiences of the population rather than a user’s prior knowledge, it can manifest very pleasurable and robust performance on different applications. Various adaptive-based studies have also verified that it is a promising method that adjusting the breeding offspring strategy and involved parameters based on its historical performance [37, 45]. In addition, it is also a favorable method to satisfy distinct requirements in different generations that adaptively selecting proper breeding strategies from existing outstanding strategies during the evolutionary process [19, 26, 38].

Furthermore, motivated by multi-swarms based techniques that have been successfully applied in different evolutionary algorithms, many scholars proposed different DE variants based on the techniques aiming to keep population diversity [4, 34, 41]. To make the best of various advantages of the multiple sub-swarms, proposing an efficient information sharing mechanism plays a paramount role. Similar to the selecting proper parameters, however, designing an optimal information sharing mechanism based on a trial-and-error scheme also requires high computational costs. Thus, finding an adaptive strategy to share information of the sub-swarms also is a feasible and promising research direction.

In this paper, inspired by the outstanding researches introduced above, we propose a fitness-based adaptive DE algorithm (FADE). In FADE, an entire population is randomly divided into multiple swarms. Furthermore, several popular breeding strategies with distinct favorable characteristics are saved in an archive. In each swarm, an individual adaptively selects its own breeding strategy from the archive in every generation according to its performance. In addition, the individual adaptively chooses a few individuals as candidate parents based on its fitness. After that, different individuals can perform their own distinct strategies to breed offspring. Furthermore, an adaptive adjustment of population size in FADE is used to rationally allocate computational resources, and then improve the convergence speed.

The rest of this paper is organized as follows. Section 2 describes the framework of the canonical DE and reviews some DE variants. Detail of FADE is described in Section 3. Extensive experiments between FADE and other eight state-of-art DEs on CEC2013 and CEC2017 test suites are introduced in Section 4. Moreover, to verify the performance of many newly introduced strategies in FADE, sensitivity analysis of the strategies is also detailed in this section. In Section 5, the performance of FADE is further testified by comparison results between it and the peer algorithms on seven real applications. Finally, conclusions are provided in Section 6.

2. Background and related work

In this section, some commonly used operators and related parameters of the canonical DE are briefly discussed. Moreover, various improvements in DE are also introduced in this section.

2.1. DE algorithm

In DE, a population $\{\mathbf{X}_i^t = (x_{i,1}^t, x_{i,2}^t, ..., x_{i,D}^t)|i = 1, 2, ..., N\}$ applies a breeding offspring strategy including mutation, crossover, and selection operators to generate offspring in each generation, where D is the number of variables in a problem and N is the population size. Details of the operators are described as follows.

2.1.1. Mutation

In each generation t, a mutant vector $\{\mathbf{V}_i^t = (v_{i,1}^t, v_{i,2}^t, ..., v_{i,D}^t)|i = 1, 2, ..., N\}$ is created for each individual $\mathbf{X}_i^t$ (named as a target vector) by a mutation operator. There are 6 frequently referred mutation strategies according to ways of generating $\mathbf{V}_i^t$, the details of them are listed as follows.

1) "DE/rand/1":

$$\mathbf{V}_i^t = \mathbf{X}_{r_1}^t + F \cdot (\mathbf{X}_{r_2}^t - \mathbf{X}_{r_3}^t) \tag{1}$$

2) "DE/rand/2":

$$\mathbf{V}_i^t = \mathbf{X}_{r_1}^t + F \cdot (\mathbf{X}_{r_2}^t - \mathbf{X}_{r_3}^t) + F \cdot (\mathbf{X}_{r_4}^t - \mathbf{X}_{r_5}^t) \tag{2}$$

3) "DE/best/1":

$$\mathbf{V}_i^t = \mathbf{X}_{best}^t + F \cdot (\mathbf{X}_{r_1}^t - \mathbf{X}_{r_2}^t) \tag{3}$$

4) "DE/best/2":

$$\mathbf{V}_i^t = \mathbf{X}_{best}^t + F \cdot (\mathbf{X}_{r_1}^t - \mathbf{X}_{r_2}^t) + F \cdot (\mathbf{X}_{r_3}^t - \mathbf{X}_{r_4}^t) \tag{4}$$

5) “DE/current-to-best/1”:

$$\mathbf{V}_i^t = \mathbf{X}_i^t + F \cdot (\mathbf{X}_{best}^t - \mathbf{X}_i^t) + F \cdot (\mathbf{X}_{r_1}^t - \mathbf{X}_{r_2}^t) \tag{5}$$

6) “DE/rand-to-best/1”:

$$\mathbf{V}_i^t = \mathbf{X}_{r_1}^t + F \cdot (\mathbf{X}_{best}^t - \mathbf{X}_{r_1}^t) + F \cdot (\mathbf{X}_{r_2}^t - \mathbf{X}_{r_3}^t) \tag{6}$$

The indices r_1, r_2, r_3, r_4, and r_5 in the above equations are mutually exclusive integers, which are randomly selected from the set $\{1, 2, ..., N\} \setminus \{i\}$. $\mathbf{X}_{best}^t$ denotes the current best individual of the population. The scaling factor F is a positive parameter used to scale the difference between two individuals.

2.1.2. Crossover

After generating the mutant vector $\mathbf{V}_i^t$, DE executes a crossover operator to generate a trial vector $\mathbf{U}_i^t = (u_{i,1}^t, u_{i,2}^t, ..., u_{i,D}^t)$ based on each pair of $\mathbf{X}_i^t$ and $\mathbf{V}_i^t$. Although there are many different crossover strategies, such as the arithmetic and the eigenvector-based crossover strategies [16], the binomial crossover is the easiest and widely accepted in many researches [26, 35, 45], which is defined as follows.

$$u_{i,j}^t = \begin{cases} v_{i,j}^t, & \text{if } rand_j(0,1) \le CR \text{ or } j = j_{rand} \\ x_{i,j}^t, & \text{otherwise} \end{cases} \tag{7}$$

where $rand_j$ is a random number uniformly distributed in the range [0, 1]; the crossover rate CR is applied to determine the probability that $u_{i,j}^t$ is copied from $v_{i,j}^t$ or $x_{i,j}^t$; and j_{rand} is a randomly selected integer in the range [1, D].

When $u_{i,j}^t$ exceeds out of a feasible region, it has to be reset within the feasible region by Eq. (8).

$$u_{i,j}^t = \begin{cases} min\{U_j, 2L_j - u_{i,j}^t\}, & \text{if } u_{i,j}^t < L_j \\ max\{L_j, 2U_j - u_{i,j}^t\}, & \text{if } u_{i,j}^t > U_j \end{cases} \tag{8}$$

where U_j and L_j are the upper and lower boundaries of the j^{th} variable, respectively.

2.1.3. Selection

When performing the selection operator, the better one between $\mathbf{X}_i^t$ and $\mathbf{U}_i^t$ is preserved to the next generation. Without loss of generality, minimization problems are considered in this paper. Hence, the selection operator in this study is described as follows.

$$\mathbf{X}_i^{t+1} = \begin{cases} \mathbf{X}_i^t, & \text{if } f(\mathbf{X}_i^t) < f(\mathbf{U}_i^t) \\ \mathbf{U}_i^t, & \text{otherwise} \end{cases} \tag{9}$$

where $f(x)$ is the function value of the vector x.

2.2. Previous studies

Over the past years, some researchers have proposed various DE variants by developing novel strategies to generate trial vector and/or choosing proper parameters.

2.2.1. Generation of trial vector

Besides the 6 popular mutation operators mentioned in Section 2.1.1, some novel generation strategies of the trial vector have been proposed to further enhance the performance of DE.

Using a mutation operator who has favorable historical performance to generate offspring in current generation is a popular and widely accepted strategy in many DE variants. For example, in MDE-DS [13], the arithmetic mean (centroid) value of the 50% best performing individuals is applied in mutation strategy. Similar as MDE-DS, a collective vector based on a linear combination of the top ranking m individuals and other randomly selected vectors is adopted to generate a new vector in CIPDE [47]. The main advantage of this type of improvements is that the exploitation can be enhanced. To balance the contradiction between the exploration and the exploitation, a new mutation strategy "DE/current-to-*lbest*/1" is proposed in APDDE [38] and ADE [43], where *lbest* is the best solution of an individual's local neighbors. Furthermore, considering that recently explored inferior solutions may provide further information about the promising progress direction, Zhang [45] proposed JADE, in which “DE/current-to-pbest/1” with optional archives is adopted. In the novel trial vector generating strategy, some recently generated inferior solutions as well as the best solution are utilized to guide the population. To loose variable correlations in non-separable problems, a covariance matrix learning method is introduced in CoBiDE [37] aiming to establish an Eigen coordinate system, and then some traditional crossover operators are performed on the modified coordinate system to generate offspring.

It has been observed that different vector generation strategies have their own distinct search behaviors [23, 21, 45]. Thus, it is a feasible and promising method that hybridizing different strategies by a composite mechanism. In TS-DE [19], for example, three mutation strategies beneficial for the exploration are applied to enhance the exploration ability in the initial evolutionary stage. On the contrary, two mutation strategies favorable for the exploitation are used to improve the convergence speed in the latter evolutionary stage. In CoDE [35], three trial vector generation strategies are used to constitute a strategy candidate pool. In each generation, each individual randomly selects its own strategy from the pool to breed offspring. Furthermore, to strengthen the adaptive ability of DE on different problems and different evolutionary stages, Qin [26] selected “DE/current-to-best/1” and “DE/rand/1” as two basic strategies in SaDE. During the optimization process, the population adapts probabilities of generating offspring by the two strategies based on their success ratios in the last few generations. The motivation of this type of improvements is adopting proper mutation strategies to meet different requirements of diverse search stages [6, 19, 35, 48].

2.2.2. Adjustment of parameters

Population size N, scaling factor F, and crossover rate CR are three indispensable parameters involved in DE. Many types of research have investigated various parameter settings in last decades [12, 25].

Although there are already various suggestions for parameter settings [12, 21], characteristics of many real applications are unknown as well as the search process of DE is dynamical. Thus, applying predefined and fixed parameters cannot satisfy various requirements of the dynamical optimization process. To deal with the problem, various dynamical adjustments for the parameters are proposed in recent years. For example, Zaharie [44] proposed an adaptive DE algorithm (ADE) with a varied population size. In NDE [33], the population size is also gradually reduced and the favorable individuals are retained as the number of generations increases. Therefore, it is helpful for enhancing the exploitation at the later evolutionary stage. Furthermore, in [23], linearly decrease methods are used to adjust F and N intending to satisfy distinct requirements in different evolutionary stages.

To take full advantage of the population's experience, various adaptive s-trategies are introduced to tune the parameters in DE [3, 17]. The main superiority of the adaptive strategies is that less prior knowledge or characteristics of a specific problem is required. In jDE [3], for instance, Brest *et al.* set F and CR as random numbers within certain ranges or as the values of the latest generation from predefined probability. The experimental results demonstrate that jDE is better than or at least the same as the traditional DE. Moreover, Qin *et al.* [26] developed a self-adaptive DE (SaDE), in which a more suitable setting of the parameters could be adaptively determined to match different phases of the search process. Based on this work, Pan *et al.* [24] proposed another self-adaptive algorithm named as SspDE, in which different F and CR settings are adjusted according to their historical performance. In JADE proposed by Zhang [45], information of recent successful F and CR is implemented to adjust the subsequent F and CR, respectively. In this condition, much knowledge of elites in the population can be used to guide other individuals to conduct more efficient search behaviors. Moreover, different fitness-based adaptation schemes for F and CR are also presented in recent researches [22, 31, 43, 48]. Extensive experimental results have verified that a well designed parameter adaptation is capable of enhancing the comprehensive performance of DE.

Another popular adjustment of parameters is designing proper integrate approaches to reasonably utilize pleasurable characteristics of different parameters

[19, 28, 32, 36]. In [36], for example, each individual randomly selects its own parameters, i.e., F and CR, from three predefined parameter settings. In [28], there are many candidate combinations of parameters saved in a pool. Along with the optimization process, those outstanding combinations of parameters measured by the success rate in the previous period are preserved for next generations while those unfavorable combinations of parameters are deleted from the pool. In TSDE [19], a pair of F and CR randomly selected from a pool is used to cooperate with a mutation operator to breed a trial vector. Furthermore, in [32], the parameters of each individual are tuned in accordance with the information on superior individuals as well as the individual difference information between two consecutive generations.

2.2.3. Topological strategy

Inspired by the population topology mechanism in particle swarm optimization (PSO), many DE variants based on a multi-swarm mechanism are emerged in recent years. To realize an adapted ensemble of different mutation strategies, Wu [40] proposed a novel DE variant, named as MPEDE, in which the entire population is dynamically partitioned into several swarms. Based on the historical performance of the swarms, different sizes are assigned for different swarms. In this way, a favorable swarm has a larger population size, and then to obtain more computational resources. In [31, 33, 46], individuals in the current population are classified into two categories, i.e., superiors and inferiors, in terms of their fitness values. In each generation, the individuals in different swarms conduct their own strategies to generate trial vector.

In the above DE variants, distinct characteristics of different individuals in a same topology are rarely considered when executing the mutation operator. To overcome the shortcoming, inspired by the phenomenon of social learning in animal societies, Cai proposed a SL-DE in 2018 [5], in which a social ranking and a social influence are assigned to each individual according to its fitness, and then a social network of the individual can be built. Based on the social network, each individual can select its own spouse to generate offspring. Similar to SL-

DE, multiple topologies with different degrees of connectivity are introduced by Sun [30]. Based on the multiple topologies, individuals in each generation select their own topologies according to the individuals' distinct types, and then to choose corresponding parents to generate offspring. In NaDE [4] and NDE [33], an individual dynamically adjusts its topology model according to its neighborhood evolutionary stage, which is characterized by its fitness value and diversity. Thus, the DE variants not only can make full use of the characteristics of individuals, but also can identify the neighborhood evolutionary for each individual.

3. FADE

3.1. Motivations

Similar to other EAs, DE also needs to establish a fair trade-off between the exploration and the exploitation. Extensive study in the last decades verify that different trial vector generation strategies and related control parameters display their distinct properties [9, 19, 26, 35, 45]. Nevertheless, selecting the most proper trial vector generation strategy and parameters for a "black-box" problem is cumbersome and unrealistic.

In recent work, it is a popular method that dividing the whole evolutionary process into many non-overlapping stages based on the number of fitness evaluations [19] or the dispersion level of the population [48]. Then, various generation strategies and parameters with distinct characteristics are assigned to the different evolutionary stages aiming to satisfy the diverse requirements, such as the exploration and the exploitation. However, due to the fact that there are various fitness landscapes in different problems even in different search areas of the same problem, it is very difficult to divide the entire evolutionary process into different stages to satisfy the distinct requirements in advance.

In fact, it is a common phenomenon in human society that allocating different tasks to different people according to their capabilities. Many studies also verify that assigning different search strategies for different individuals can yield very favorable performance [1, 11]. Furthermore, multi-swarm techniques have been successfully applied in EAs [15, 39, 40, 42] since they can yield very pleasurable performance in keeping population diversity.

Moreover, how to rationally allocate computational resources is very crucial for an efficient DE. A common method is that the population size continually reduce during the evolutionary process [23] or supplementing new individuals into the current population when stagnation is detected [49]. However, we regard that, for a specific problem, fitness landscapes in different search areas have distinctive characteristics. Thus, dynamic adjusting the population size, i.e., both deleting and adding individuals, based on properties of the current fitness landscape is a promising strategy to rationally allocate the computational resources.

Inspired by these observations, we propose a fitness-based adaptive DE (FADE) in this research. In FADE, three popular breeding strategies with distinct properties on the exploration and the exploitation are stored in an archive. Moreover, the entire population is split into multiple swarms. During the evolutionary process, individuals in each swarm ***a***daptively select their own ***b***reeding ***s***trategies and candidate parents (ABS) to generate offspring. Furthermore, an ***a***daptive strategy for ***p***opulation ***s***ize (APS) applied to rationally allocate computational resources is adopted to enhance the convergence speed.

3.2. Framework of FADE

Based on the above mentioned discussions, the flowchart of FADE is described in Figure 1. From the table we can see that FADE consists of three main procedures, i.e., initialization, ABS, and APS procedures. After the initialization, FADE uses ABS procedure to deal with various requirements in different fitness landscapes, and applies APS procedure to rationally assign computational resources.

The core ideas behind FADE are elucidated one by one in following sections. Note that, the definitions of *Condition* 1 and *Condition* 2 in APS procedure are detailed in Section 3.5.1 and Section 3.5.2, respectively.

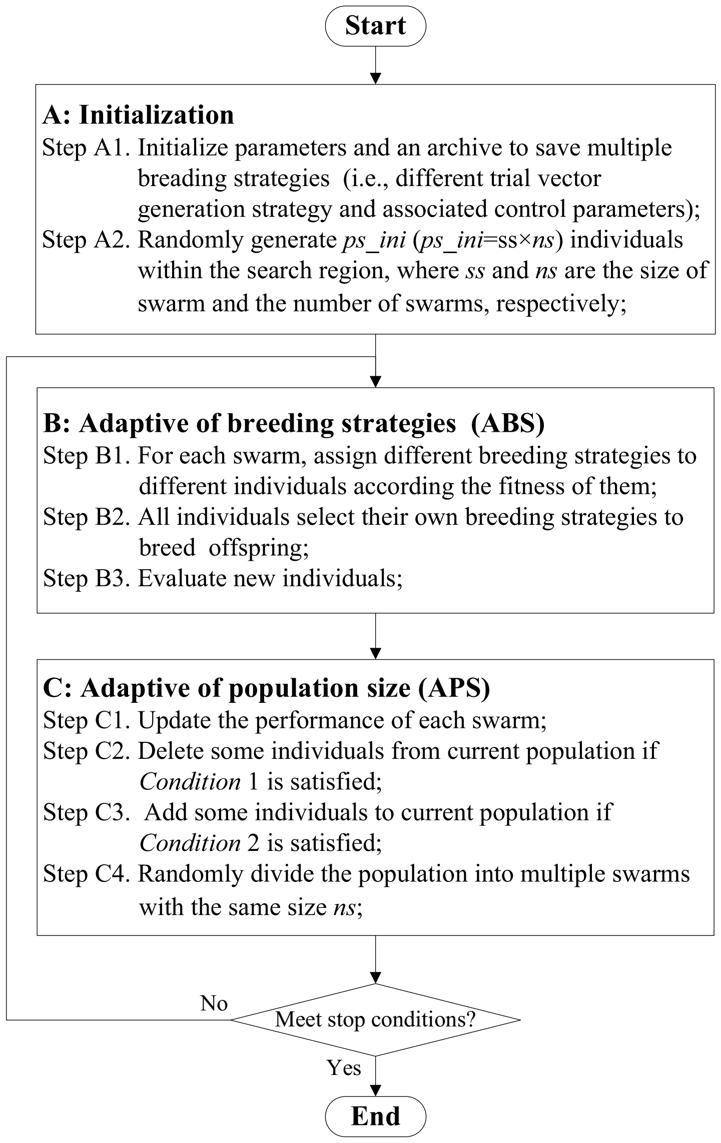

Start
A: Initialization
Step A1. Initialize parameters and an archive to save multiple breading strategies (i.e., different trial vector generation strategy and associated control parameters);
Step A2. Randomly generate ps_ini (ps_ini=ss×ns) individuals within the search region, where ss and ns are the size of swarm and the number of swarms, respectively;
B: Adaptive of breeding strategies (ABS)
Step B1. For each swarm, assign different breeding strategies to different individuals according the fitness of them;
Step B2. All individuals select their own breeding strategies to breed offspring;
Step B3. Evaluate new individuals;
C: Adaptive of population size (APS)
Step C1. Update the performance of each swarm;
Step C2. Delete some individuals from current population if Condition 1 is satisfied;
Step C3. Add some individuals to current population if Condition 2 is satisfied;
Step C4. Randomly divide the population into multiple swarms with the same size ns;
Meet stop conditions?
No
Yes
End

Figure 1: The flowchart of FADE.

3.3. Initialization

Before the evolutionary process, multiple breeding strategies with distinct characteristics need to be saved in an archive A_b. In FADE, three breeding strategies that beneficial for the exploration, the exploitation, and the trade-off abilities have to be designed. For simplicity, some existing popular strategies are chosen in this work.

In general, "DE/rand/1" and "DE/rand/2" are conducive to the exploration [45], while "DE/best/1" and "DE/current-to-best/2" are well-suited to the exploitation [20, 21]. In addition, various studies indicate that different F and CR can offer distinct characteristics [9, 19, 26, 35]. Concretely, a smaller F is beneficial for the exploitation, while a larger one is effective for the exploration. Furthermore, a smaller CR focuses on keeping the local exploitation since the target vector contributes more information to the trial vector. On the contrary, a larger CR is favorable for maintaining the population diversity due to the newly generated trial vector inheriting more information from the mutant vector.

In FADE, three breeding strategies with different properties are saved in A_b. The detail of the three settings of breeding strategies is presented as follows.

- *Breeding Strategy* 1: [DE/current-to-best/1, F=0.5, CR=0.1], which is applied to improve the exploitation ability;
- *Breeding Strategy* 2: [DE/current-to-lbest/1, F=0.8, CR=0.5], which can achieve a trade-off between the exploration and the exploitation; and
- *Breeding Strategy* 3: [DE/rand/2, F=1.0, CR=0.9], which is used to enhance the exploration ability.

Note that, the symbol "lbest" in *Breeding Strategy* 2 denotes the local best solution of a swarm that an individual belongs to. Moreover, the binomial crossover is applied in this research.

Furthermore, at the beginning of evolutionary process, a population with ps_ini individuals $\{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_{ps\_ini}\}$ should to be randomly generated within a feasible space $\Re^D$, where D is the number of variables in a problem. Then all the individuals are randomly divided into ns swarms, each of them has a same size ss. Many other indispensable parameters also need to be initialized before the evolution.

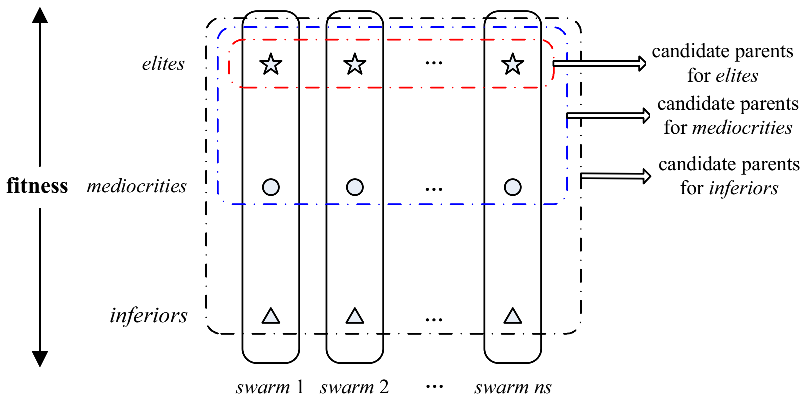
3.4. Adaptive of breeding strategies (ABS)

In FADE, the size of each swarm is set as ss=3. In each generation, 3 individuals in a swarm can be classified into 3 types, i.e., an elite, a mediocrity, and an inferior, based on their fitness. Concretely, the best individual in the swarm is regarded as an elite, while the worst individual in the swarm is deemed as an inferior. The individual being neither the best individual nor the worst individual in the swarm is named as a mediocrity. Based on the classifying operator, three different breeding strategies in A_b are assigned to the elite, mediocrity, and inferior individuals, respectively.

In this research, the elite dedicates to the exploitation aiming to enhance the solution accuracy. On the contrary, the inferior pours more attention to the exploration intending to find out more promising solutions in other regions. For the mediocrity, assigning a trade-off search behavior is a pleasurable choice. Hence, in each generation, the elite, mediocrity, and inferior in each swarm are committed to the exploitation ability, the trade-off ability, and the exploration ability, respectively.

Thus, individuals in FADE adaptively select their own breeding strategies to satisfy various search objectives of them. Concretely, the elite, mediocrity, and inferior individuals in a swarm adopt *Breeding Strategy* 1, *Breeding Strategy* 2, and *Breeding Strategy* 3 in A_b to breed offspring, respectively. Under this condition, an individual may select different breeding strategies in different evolutionary stages because the relative performance of it in the swarm may be changed during the evolutionary process. Thus, the individual can perform diverse search behaviors in the different stages.

After adopting a proper breeding strategy, an individual conducts the s-trategy to generate offspring based on a few individuals named as candidate parents. Unlike the classic DE algorithm, in which all individuals in the curren-t population are deemed as candidate parents, an individual in FADE has its own candidate parents according to the individual's type. Concretely, candidate parents of an elite consist of all elites in all swarms. In this case, cooperations among the elites are advantageous to the exploitation. On the contrary, candidate parents of an inferior are individuals in the entire population since the primary task of the inferior is keeping diversity. For a mediocrity, elites and mediocrities in all swarms are regarded as its candidate parents. Under the condition, the mediocrity can derive much positive knowledge and diverse information from the elites and mediocrities, respectively. The process of choosing candidate parents for different individuals can be demonstrated as Figure 2.



fitness
elites
mediocrities
inferiors
candidate parents for *elites*
candidate parents for *mediocrities*
candidate parents for *inferiors*
swarm 1
swarm 2
swarm ns

Figure 2: Candidate parents for different individuals in each generation.

Based on the above introduction, the ABS procedure in generation t is detailed in Algorithm 1.

3.5. Adaptive of population size (APS)

In FADE, the adaptive of population size includes two processes, i.e., deleting-individual process and adding-individual process. The core idea of the adaptation is that deleting some unfavorable individuals from the current population if the current fitness landscape is easy to be optimized. On the contrary, if the current population cannot find out more promising solutions, some pleasurable individuals are added to the current population. Since it is impracticable that without limitation of population size during the adaptation, a predefined maximum population size ps_{max} and minimum population size ps_{min} are used to clamp the population size within an interval $[ps_{min},\ ps_{max}]$. Relying on the

Algorithm 1. ABS ( )

```
Input: $t$, $Pop$, $A_b$;
01: $ps = |Pop|$; /* $ps$ is the current population size */
02: For $i$=1 to $ps$
03:   If $\mathbf{X}_i^t$ has the best fitness within the swarm it belongs to; /* elite */
04:     Select elites parents for $\mathbf{X}_i^t$ according to Figure 2;
05:     Select Breeding Strategy 1 from $A_b$ to generate offspring $\mathbf{X}_i^{t+1}$;
06:   ElseIf $\mathbf{X}_i^t$ has the worst fitness within the swarm it belongs to; /* inferior */
07:     Select inferiors parents for $\mathbf{X}_i^t$ according to Figure 2;
08:     Select Breeding Strategy 3 from $A_b$ to generate offspring $\mathbf{X}_i^{t+1}$;
09:   Else /* mediocrity */
10:     Select mediocrities parents for $\mathbf{X}_i^t$ according to Figure 2;
11:     Select Breeding Strategy 2 from $A_b$ to generate offspring $\mathbf{X}_i^{t+1}$;
12:   End if
13: End For
14: Randomly divide $Pop$ into multiple swarms.
Output: $Pop$.
```

adaptation of population, computational resources can be rationally allocated during the evolutionary process.

3.5.1. Deleting-individual process

Generally speaking, when successive generations of improvement (SG_{imp}) of the global best individual ($\mathbf{X}_{gbest}^t$) is greater than a predefined threshold $MaxG_{imp}$, we regard that the current fitness landscape is easy to be optimized, and deleting a few unfavorable individuals from the current population is propitious to save computational resources.

When deleting individuals from the current population, those individuals with the worst fitness are deleted from the population. Considering that all swarms in FADE have the same size, the number of deleted individuals (nd) is a multiple of ss. Note that only if the current population size is greater than ps_{min} can the deleting-individual process be performed. In other words, the *Condition* 1 appeared in Figure 1 includes not only the value of SG_{imp}, but also the current population size (ps). After deleting a few unfavorable individuals from the current population, individuals in the population are randomly divided into multiple swarms. The procedure of the deleting-individual is detailed in

Algorithm 2.

```
Algorithm 2. Deleting_Individual ( )
Input: $Pop$, $SG_{imp}$, $MaxG_{imp}$, $ps_{min}$;
01: $ps = |Pop|$;
02: If $SG_{imp} \geq MaxG_{imp}$ && $ps > ps_{min}$ // Condition 1 in Figure 1
03:     $SG_{imp}$=0;
04:     Sort $Pop$ satisfying $f(\mathbf{X}^t_{i_1}) \geq f(\mathbf{X}^t_{i_2}) \geq \ldots \geq f(\mathbf{X}^t_{i_{ps}})$;
05:     $Pop$=$Pop \setminus \{\mathbf{X}^t_{i_1}, \mathbf{X}^t_{i_1}, \ldots \mathbf{X}^t_{i_{nd}}\}$;
06: End If
Output: $Pop$, $SG_{imp}$.
```

3.5.2. Adding-individual process

On the contrary, the successive stagnation generations (SG_{stag}) of $\mathbf{X}_{gbest}$ are regarded as a criterion for the adding-individual process. Concretely, while SG_{stag} is larger than a predefined threshold $MaxG_{stag}$, we think that individuals in the current population cannot effectively address the problem. In this case, a few new individuals need to be added to the current population. In other words, the *Condition* 2 appeared in Figure 1 is that SG_{stag} is greater than $MaxG_{stag}$.

Another crux of the matter in the adding-individual process is how to obtain the new individuals. The easiest way to deal with the matter is random generating some individuals within the feasible space $\Re^D$. However, we regard that generating new promising individuals based on some elites of swarms is a favorable method for two advantages. One is that it is beneficial for reducing computational consumption. The other one is that much helpful knowledge can be drawn from the elites.

Thus, in this work, the elites of all swarms in each generation saved in A_E are used to generate a few promising individuals. At the beginning of the evolutionary process ($t = 1$), all elites, i.e., the local best solutions $\mathbf{X}^t_{lbest_i}$ ($1 \leq i \leq ns$) of all the swarms, are saved in A_E. Thereafter, the elites in generation t ($t > 1$) are applied to update A_E. Note that the maximum size of A_E is ns. Hence, when the size of A_E is greater than ns, those unfavorable individuals need to be removed from A_E. The update procedure of A_E at generation t is detailed in Algorithm 3. Note that $\mathbf{X}^t_{lbest_i}$ is the local best individual in the ith swarm in generation t.

```
Algorithm 3. Update_$A_E$ ( )
Input: $t$, $A_E$, $\mathbf{X}^t_{lbest_i}$ $(1 \leq i \leq ns)$;
01: If $A_E$==$\emptyset$ /*Initialize $A_E$ */
02:     $A_E$={$E_1, E_2, \ldots, E_{ns}$}={$\mathbf{X}^t_{lbest_1}, \mathbf{X}^t_{lbest_2}, \cdots, \mathbf{X}^t_{lbest_{ns}}$};
03: Else /*Update $A_E$ */
04:     For $i$=1 to $ns$
05:         If $\mathbf{X}^t_{lbest_i} \notin A_E$
06:             $A_E = A_E \cup \mathbf{X}^t_{lbest_i}$;
07:         End If
08:     End For
09: End If
10: If $|A_E| > ns$ /* $|A_E|$ is the size of $A_E$ */
11:     Sort $A_E$ satisfying $f(E_{i_1}) \leq f(E_{i_2}) \leq \ldots \leq f(E_{i_{|A_E|}})$;
12:     $A_E$=$A_E \setminus \{E_{i_{ns+1}}, E_{i_{ns+2}}, \ldots, E_{i_{|A_E|}}\}$;
12: End If
Output: $A_E$.
```

Based on A_E, new individuals that need to be added into the current population can be generated by GA operators, the details of which are presented in Algorithm 4. Note that σ_j defined as Eq.(10), which denotes the distribution of the elites in A_E, is used to confine the mutation limitation.

$$\sigma_j = \sqrt{\frac{\sum_{i=1}^{ns}(x^t_{lbest_i,j} - \mu_j)^2}{ns}} \tag{10}$$

where $x^t_{lbest_i,j}$ is the jth dimension of $\mathbf{X}^t_{lbest_i}$, and $\mu_j = \frac{\sum_{i=1}^{ns} x^t_{lbest_i,j}}{ns}$ is the mean value of the jth dimension on all elites in A_E.

Based on the aforementioned description, the procedure of the adding individuals is presented in Algorithm 5. Similar to the deleting process, the number of added individuals na is also a multiple of ss. After adding a few individuals to the current population, individuals in the population are randomly divided into multiple swarms.

Algorithm 4. Generate_Individual ( )

```
Input: $A_E$, $p_c$ and $p_m$;
01: Compute $\sigma_j$ for each $j$ ($1 \leq j \leq D$) according to (10);
02: For $i$=1 to $na$
03:    For $j$=1 to $D$
04:       Randomly select $E_{i1}$ and $E_{i2}$ from $A_E$;
05:       If $rand < p_c$ /* crossover */
06:          $\hat{x}_{i,j} = e_{i1,j}$; /*$e_{i1,j}$ is the $j^{th}$ dimension of $E_{i1}$ */
07:       Else
08:          $\hat{x}_{i,j} = e_{i2,j}$;
09:       End If
10:       If $rand < p_m$ /* Gauss mutation */
11:          $\hat{x}_{i,j} = Gauss(\hat{x}_{i,j}, \sigma_j)$;
12:       End If
13:    End For
14: End For
Output: $\hat{\mathbf{X}}_i$=[$\hat{x}_{i,1}, \hat{x}_{i,2}, ..., \hat{x}_{i,D}$] ($1 \leq i \leq na$).
```

Algorithm 5. Adding_Individual ( )

```
Input: $Pop$, $A_E$, $SG_{stag}$, $MaxG_{stag}$, $ps_{max}$;
01: If $SG_{stag} \geq MaxG_{stag}$ // Condition 2 in Figure 1
02:    $SG_{stag} = 0$; $ps = |Pop|$;
03:    Generate_Individual ( );
04:    If $ps < ps_{max}$
05:          $Pop = Pop \cup \hat{\mathbf{X}}_i (1 \leq i \leq na)$;
06:    Else
07:          Sort $Pop$ satisfying $f(\mathbf{X}^t_{i_1}) \geq f(\mathbf{X}^t_{i_2}) \geq \ldots \geq f(\mathbf{X}^t_{i_{ps}})$;
08:          Replace $\mathbf{X}^t_{i_j}$ by $\hat{\mathbf{X}}_i (1 \leq j \leq na)$;
09:    End If
10:    Randomly divide $Pop$ into multiple swarms;
11: End If
Output: $Pop$, $SG_{stag}$.
```

3.6. Pseudo-code implementation of FADE

Built on the aforementioned discussions, the pseudo code of FADE can be detailed in Algorithm 6. The initialization process is illustrated in line 01 - line 03. In the main loop (i.e., line 04 - line 11), two adaptive strategies are included. The adaptive breeding strategy is conducted in line 07, while the adaptive population size is performed in line 08 - line 09 based on A_E obtained in line 06.

Algorithm 6. FADE ( )

```
Input: $MaxG_{imp}$, $MaxG_{stag}$, $ps_{min}$, $ps_{max}$, $ns$, $ss$, $\Re^D$, $nd$, $na$, $ps\_ini$;
01: $t = 1$, $SG_{imp} = 0$, $SG_{stag} = 0$, $A_E = \emptyset$;
02: Randomly generate a population $\{\mathbf{X}_1^t, \mathbf{X}_2^t, \cdots \mathbf{X}_{ps\_ini}^t\}$ ($ps\_ini$=$ss$×$ns$);
03: Randomly divide the current population into $ns$ swarms with the same size $ss$;
04: While not meet stop conditions
05:     Evaluate individuals in each swarm;
06:     Update_$A_E$ ( );
07:     ABS( );
08:     Deleting_Individual ( );
09:     Adding_Individual ( );
10:     $t = t + 1$;
11: End While
Output: $\mathbf{X}_{gbest}^t$. /* $\mathbf{X}_{gbest}^t$ is the best solution of $Pop$ */
```

3.7. Complexity analysis of FADE

Compared with the original DE, the additional computations of FADE are due to the three novel operators, i.e., ABS, APS, and update of A_E. In ABS, the additional cost is incurred by sorting all the swarms and selecting proper parents for a specific individual to generate offspring (see line 03 - line 12 in Algorithm 1). The random selection of parents according to Figure 2 for an individual takes $O(1)$. Thus, the additional complexity caused by ABS is the sorting process, whose complexity is $O(ns \times \log ss)$, where ns and ss are the number of swarms and the size of each swarm, respectively.

In APS, there are two processes, i.e., the deleting-individual and the adding-individual. The complexity of the deleting-individual process is incurred by the sorting operator (see line 04 in Algorithm 2), the complexity of which is $O(\log ps)$, where ps is the current population size. In the adding-individual process, two process, i.e., the sorting operator (see line 07 in Algorithm 5) and the generation of individuals (see line 03 in Algorithm 5), are time-consuming. The complexity of the sorting operator is $O(\log ps)$. For the generating individuals process (see Algorithm 4), the complexity is $O(ns \times D + na \times D)$. Thus, the complexity of the adding-individual process is $O(\log ps + ns \times D + na \times D)$. S-ince the deleting-individual process and the adding-individual process cannot be executed at the same time, the complexity of APS is $max\{O(\log ps), O(\log ps +$

$ns \times D + na \times D)\}$. In APS, ns is greater than na. Thus, the complexity of APS is $O(\log ps + ns \times D)$.

Moreover, the additional complexity caused by the update of A_E is the sorting process (see line 11 in Algorithm 3), the complexity of which is $O(\log |A_E|)$, where $|A_E|$ is the size of the archive A_E.

Since the complexity of the original DE algorithm is $O(Gmax \times ps \times D)$, where G_{max} is the maximum number of generations, the total complexity of FADE is $O(Gmax \times (ps \times D + ns \times \log ss + \log ps + ns \times D + \log |A_E|))$. Considering that $ps = ns \times ss$, we can obtain a conclusion that FADE remains computationally efficacious in time when compared to the original DE algorithm.

4. Experimental results and discussions

4.1. Benchmark functions and peer algorithms

In this work, CEC2013 and CEC2017 test suites, consisting of unimodal, simple multimodal, hybrid, and composition functions, are selected to comprehensively test characteristics and performance of FADE. For demonstration purpose, 28 test functions in CEC2013 test suite are denoted as f_1-f_{28} while 30 functions in CEC2017 test suite are recorded as F_1-F_{30}. More detailed information of the test functions can refer to literatures [2, 18]. In experiments carried out in this section, the allowed maximum number of function evaluations (*MaxFEs*) for each run is set to 10 000D (i.e., 300 000 and 500 000 for D=30 and D=50, respectively) based on the guidelines provided in the special session of CEC2013 [18] and CEC2017 [2].

To testify the performance of FADE, eight state-of-art DE variants appeared from 2009 to 2019 are selected in this research. The first competitor is SaDE [26], in which four trial vector generation strategies (i.e., DE/rand/1/bin, DE/rand/2/bin, DE/rand-to-best/2/bin, and DE/current-to-rand/1) are maintained in a strategy candidate archive. During the evolutionary process, the more favorable performance one strategy displayed in previous generations, the more probability it will be chosen in the current generation to breed offspring. CoDE [35] is the second peer algorithm. In CoDE, three trial vector generation strategies (i.e., DE/rand/1/bin, DE/rand/2/bin, and DE/current-to-rand/1) and three control parameter settings (i.e., [F=1.0, CR=0.1], [F=1.0, CR=0.9], and [F=0.8, CR=0.8]) are randomly combined to generate the trial vectors. The third peer algorithm is CoBiDE [37], in which the covariance matrix learning and the bimodal distribution parameters setting are incorporated into the DE framework. The covariance matrix learning strategy is used to relax the dependence of DE on the coordinate system to a certain degree and the bimodal distribution for both F and CR composed of two Cauchy distributions can emphasize the exploration and exploitation abilities, respectively. In the fourth peer algorithm TSDE [19], the whole evolutionary process is divided into a former stage and a latter stage based on the number of fitness evaluations. TSDE focuses on the exploration in the former stage based on three trial vector generation strategies (i.e., DE/rand/1/bin, DE/rand/2/bin, and DE/current-to-rand/1). On the contrary, TSDE pays much attention to the exploitation in the latter stage by utilizing two trial vector generation strategies (DE/current-to-best/1 and DE/current-to-rand/1). EFADE [22] is the fifth competitor, in which a new triangular mutation operator is introduced. Moreover, two adaptation schemes are used to update the control parameters (i.e., F and CR). The sixth competitor is EDEV [41], in which a high-level ensemble strategy is proposed. In EDEV, three highly popular and efficient DE variants, i.e., JADE, CoDE, and EPSDE, are organically integrated based on a multi-population framework. DI-DE [32] is the seventh competitor, in which each individual possesses its own parameters and strategy, which are tuned in accordance with the information on superior individuals. The last peer algorithm is DPADE [7], in which an enhanced parameter adaptation technique is introduced. Note that all the mutation strategies and parameter settings in the eight peer DE variants are the same as those given in the original references.

To obtain statistical results, each algorithm is carried out 51 independent runs on each benchmark function.

Table 1: Basic Information of All the Peer Algorithms

Algorithm	Year	Characteristics
FADE	–	$MaxG_{imp}$=$MaxG_{stag}$=2, ps_{min}=ps_{ini}/3, ps_{min}=ps_{ini}, ns=25 or ns=40
SaDE [26]	2009	τ_1=τ_2=0.1, F_l=0.1, F_u=0.9, N=50
CoDE [35]	2011	Composite 3 trial vector generation strategies and 3 settings of F and CR, N=30
CoBiDE [37]	2014	A covariance matrix learning and a bimodal distribution setting for F and CR, N=60
TSDE [19]	2016	Composite 5 trial vector generation strategies and 3 settings of F and CR, N=50
EFADE [22]	2018	Composite 3 trial vector generation strategies, adaptive scheme for F and CR, N=50.
EDEV [41]	2018	λ_1=λ_2=λ_3=0.1, λ_4=0.7, ng=20, N=50 or N=100
DI-DE [32]	2019	Composite 3 trial vector generation strategies, G_0=0.15*G_{max}, ps_1=ps_2=0.35, p=0.1, σ=0.8∼0.3, N=180
DPADE [7]	2019	"DE/rand/1", c=0.1, P_{min}=0.1, ε=0.001, N=100

Settings of the newly introduced parameters in FADE and corresponding discussions are presented as follows:

1) [ss=3, ns=25 or 40]. ss=3 means that each swarm contains 3 individuals. Thus, in each generation, there are only one elite, one mediocrity, and one inferior in each swarm. ns=25 or 40 denotes the initial population contains 25 or 40 swarms when optimizing a problem with 30 or 50 variables, respectively.

2) [$MaxG_{imp}$=2, $MaxG_{stag}$=2], which means that deleting some worst individuals from the current population if the successive improvement generations of $\mathbf{X}_{best}^{t}$ is larger than 2, and adding new individuals into current population while the successive stagnation generations of $\mathbf{X}_{best}^{t}$ is greater than 2. Note that, in this study, 6 individuals (i.e., nd=6) are deleted from the population when performing the deleting-individual process. On the contrary, 3 individuals (i.e., na=3) are added into the population when conducting the increasing-individual process.

3) [ps_{min}=ps_ini/3, ps_{max}=ps_ini], where ps_ini is the population size at the initialization stage. ps_{min} and ps_{max} are the lower limit and upper limit of the population size when performing the change population size operator.

Note that the above mentioned values are determined by a trial-and-error process. Although we have a rudimentary understanding of how the different parameters affect the performance of FADE, the configurations of the parameters may not be the best settings for a particular problem because there may be dependencies between the different parameters, which may cause the combination of them no longer to be an optimal setting for FADE.

4.2. Comparison of solution accuracy

In this section, experiments are conducted to compare the performance of each algorithm, in terms of the mean values (Mean) and standard deviations (Std. Dev) of error values $f(\mathbf{X})$-$f(\mathbf{X}^*)$ over 51 independent runs, where $\mathbf{X}$ is the best solution found by the algorithm in each run, and $\mathbf{X}^*$ is the actual global optimum of the test function. The experimental results on CEC2013 test suite are presented in Table 2 and Table 3 while the results on CEC2017 test suite are provided in Table 4 and Table 5. The best results of the mean values are marked in bold. Moreover, for each test function, rank values of all peer algorithms measured by the mean values are also presented in these tables. Furthermore, t-tests are also conducted between FADE and other 8 peer algorithms, the results of which are also listed in Table 2 - Table 5. Note that, the t-test results appeared in the tables are "+", "-" and "=", which denote that FADE is significantly better than, significantly worse than and almost the same as the corresponding competitor algorithms, respectively. For instance, "8.02E-14(6+)" in the first row of Table 2 denotes that the mean value yielded by EFADE on f_1 is 8.02E-14, the rank value of which is 6. Moreover, the symbol '+" indicates that the performance of FADE on f_1 with 30 variables is significantly better than EFADE, contributing to the t-test result.

4.2.1. Experimental results on CEC2013 test suite

The comparison results listed in Table 2 indicate that FADE, SaDE, and EDEV yield the best performance, in terms of the number of archived best solutions ((#)*Best*), followed by CoBiDE and DI-DE. Concretely, EDEV yields the best performance on the unimodal functions (f_1-f_5) and the simple multimodal functions (f_6-f_{20}). On the contrary, FADE and SaDE offer the most favorable performance on the composition functions (f_{21}-f_{28}). The results verify that FADE dominates EDEV on the more complicated functions. The results presented in Table 3 show that FADE displays more comprehensive performance

Table 2: Comparison results of solution accuracy on CEC2013 test suite (D=30).

		FADE	SaDE	CoDE	CoBiDE	TSDE	EFADE	EDEV	DI-DE	DPADE
f_1	Mean(rank)	**0.00E+00**(1)	**0.00E+00**(1=)	1.43E-13(7+)	**0.00E+00**(1=)	**0.00E+00**(1=)	8.02E-14(6+)	**0.00E+00**(1=)	2.63E-13(9+)	2.05E-13(8+)
	Std.Dev	0.00E+00	0.00E+00	1.11E-13	0.00E+00	0.00E+00	1.10E-13	0.00E+00	1.05E-13	6.83E-14
f_2	Mean(rank)	2.66E+04(2)	3.77E+05(8+)	1.04E+05(6+)	5.46E+04(5+)	3.96E+04(3+)	4.17E+04(4+)	**2.22E+04**(1=)	2.83E+05(7+)	8.66E+06(9+)
	Std.Dev	1.69E+04	1.82E+05	6.24E+04	2.65E+04	2.21E+04	2.62E+04	1.81E+04	1.74E+05	2.18E+06
f_3	Mean(rank)	**1.14E+01**(1)	1.93E+07(8+)	4.86E+06(7+)	1.01E+02(2+)	1.08E+06(5+)	6.15E+05(3+)	8.79E+05(4+)	1.73E+06(6+)	4.44E+07(9+)
	Std.Dev	2.99E+01	3.72E+07	2.01E+07	3.72E+02	2.89E+06	1.40E+06	3.08E+06	2.93E+06	2.80E+07
f_4	Mean(rank)	5.50E+00(5)	2.61E+03(8+)	1.69E+00(4-)	**8.78E-05**(1-)	1.35E+00(3-)	7.63E+00(6=)	3.53E-01(2-)	2.48E+03(7+)	4.64E+04(9+)
	Std.Dev	5.92E+00	1.06E+03	4.17E+00	2.69E-04	4.00E+00	6.62E+00	3.82E-01	3.44E+03	6.90E+03
f_5	Mean(rank)	1.34E-14(4)	**0.00E+00**(1-)	8.47E-14(5+)	1.11E-13(7+)	1.11E-14(3=)	1.05E-13(6+)	**0.00E+00**(1-)	5.17E-13(9+)	1.14E-13(8+)
	Std.Dev	3.70E-14	0.00E+00	7.13E-14	1.59E-14	3.41E-14	3.09E-14	0.00E+00	8.01E-13	1.27E-29
f_6	Mean(rank)	8.84E+00(5)	3.08E+01(9+)	4.51E+00(3-)	9.66E+00(6=)	4.78E+00(4-)	4.12E+00(2-)	**5.47E-01**(1-)	1.41E+01(7+)	2.78E+01(8+)
	Std.Dev	5.89E+00	2.82E+01	7.39E+00	1.25E+01	1.01E+01	3.43E+00	1.39E+00	2.09E+00	1.80E+01
f_7	Mean(rank)	**2.53E+00**(1)	2.88E+01(8+)	1.10E+01(5+)	2.65E+00(2=)	1.05E+01(4+)	3.98E+00(3+)	1.19E+01(6+)	1.26E+01(7+)	7.98E+01(9+)
	Std.Dev	3.86E+00	1.37E+01	7.14E+00	2.12E+00	6.95E+00	3.00E+00	7.18E+00	6.60E+00	1.44E+01
f_8	Mean(rank)	2.09E+01(3)	2.10E+01(9+)	**2.08E+01**(1-)	2.09E+01(6=)	2.09E+01(4=)	2.09E+01(5=)	2.09E+01(2=)	2.09E+01(7=)	2.10E+01(8+)
	Std.Dev	4.90E-02	4.53E-02	2.21E-01	5.24E-02	8.20E-02	4.91E-02	7.23E-02	5.77E-02	3.94E-02
f_9	Mean(rank)	1.62E+01(5)	1.88E+01(7+)	1.42E+01(3-)	**1.14E+01**(1-)	1.41E+01(2-)	1.45E+01(4=)	1.81E+01(6+)	2.16E+01(8=)	2.95E+01(9+)
	Std.Dev	5.46E+00	2.56E+00	3.33E+00	3.42E+00	3.24E+00	3.43E+00	3.54E+00	5.57E+00	1.76E+00
f_{10}	Mean(rank)	**1.13E-02**(1)	2.80E-01(7+)	4.00E-02(5+)	2.88E-02(3+)	3.01E-02(4+)	2.55E-02(2+)	5.04E-02(6+)	9.46E-02(+)	3.77E-01(9+)
	Std.Dev	1.24E-02	1.78E-01	2.79E-02	1.95E-02	1.88E-02	1.29E-02	3.75E-02	6.11E-02	1.69E-01
f_{11}	Mean(rank)	2.63E+01(9)	7.80E-02(8-)	3.90E-02(7-)	8.92E-15(2-)	1.95E-02(6-)	1.67E-14(3-)	**0.00E+00**(1-)	1.19E-07(5-)	1.65E-13(4-)
	Std.Dev	1.06E+01	3.36E-01	1.95E-01	2.09E-14	1.39E-01	2.62E-14	0.00E+00	2.95E-07	8.06E-14
f_{12}	Mean(rank)	3.87E+01(4)	4.73E+01(7+)	3.82E+01(3=)	4.01E+01(6=)	4.75E+01(8+)	3.53E+01(2=)	3.93E+01(5=)	**2.85E+01**(1=)	1.34E+02(9+)
	Std.Dev	8.77E+00	1.22E+01	1.13E+01	1.04E+01	1.56E+01	1.06E+01	8.15E+00	1.08E+01	2.27E+01
f_{13}	Mean(rank)	7.23E+01(3)	9.90E+01(8+)	8.28E+01(7+)	7.93E+01(5=)	8.01E+01(6+)	7.44E+01(4=)	6.93E+01(2=)	**5.27E+01**(1=)	1.91E+02(9+)
	Std.Dev	1.74E+01	2.61E+01	2.43E+01	2.02E+01	2.62E+01	2.77E+01	2.16E+01	1.66E+01	2.57E+01
f_{14}	Mean(rank)	7.37E+02(8)	**7.57E-01**(1-)	9.32E+02(9=)	2.29E+02(7-)	5.26E+00(4-)	1.74E+00(2-)	2.82E+00(3-)	1.20E+02(6-)	1.54E+01(5+)
	Std.Dev	2.78E+02	1.07E+00	9.88E+02	5.02E+01	4.35E+00	1.24E+00	1.61E+01	3.26E+01	3.91E+00
f_{15}	Mean(rank)	4.28E+03(7)	4.84E+03(9+)	3.35E+03(3-)	**2.90E+03**(1-)	3.45E+03(4-)	4.78E+03(8=)	3.08E+03(2-)	3.54E+03(5-)	3.60E+03(6-)
	Std.Dev	1.49E+03	9.12E+02	5.06E+02	4.38E+02	5.50E+02	1.55E+03	5.57E+02	7.40E+02	3.19E+02
f_{16}	Mean(rank)	2.31E+00(8)	2.21E+00(7=)	3.56E-01(2-)	1.76E+00(6-)	**3.41E-01**(1-)	2.36E+00(9=)	6.64E-01(3-)	1.33E+00(5-)	1.10E+00(4-)
	Std.Dev	3.20E-01	2.30E-01	2.52E-01	1.07E+00	2.02E-01	2.56E-01	5.88E-01	2.31E-01	1.83E-01
f_{17}	Mean(rank)	5.80E+01(9)	**3.05E+01**(1-)	3.53E+01(7-)	3.74E+01(8-)	3.07E+01(4-)	3.25E+01(5-)	3.06E+01(2-)	3.31E+01(6-)	3.07E+01(3-)
	Std.Dev	9.86E+00	4.08E-02	1.02E+01	9.95E-01	1.85E-01	1.00E+00	5.18E-01	5.79E-01	6.39E-02
f_{18}	Mean(rank)	1.38E+02(6)	1.40E+02(7=)	6.59E+01(2-)	6.73E+01(3-)	6.74E+01(4-)	1.80E+02(8+)	**6.01E+01**(1-)	7.78E+01(5-)	1.94E+02(9+)
	Std.Dev	4.19E+01	3.61E+01	1.45E+01	9.01E+00	9.81E+00	3.37E+01	7.58E+00	2.89E+01	1.63E+01
f_{19}	Mean(rank)	2.74E+00(5)	3.82E+00(9+)	2.26E+00(3=)	2.83E+00(6=)	1.88E+00(2-)	3.58E+00(8+)	2.89E+00(7+)	2.47E+00(4=)	**1.32E+00**(1-)
	Std.Dev	1.41E+00	8.62E-01	2.13E+00	1.18E+00	4.22E-01	3.15E-01	5.82E-01	2.60E-01	2.22E-01
f_{20}	Mean(rank)	1.12E+01(4)	**1.08E+01**(1-)	1.17E+01(6+)	1.13E+01(5=)	1.11E+01(2=)	1.18E+01(8+)	1.18E+01(7+)	1.11E+01(3-)	1.19E+01(9+)
	Std.Dev	4.60E-01	5.91E-01	1.37E+00	5.75E-01	5.48E-01	3.84E-01	7.45E-01	3.50E-01	4.62E-01
f_{21}	Mean(rank)	2.77E+02(2)	**2.75E+02**(1=)	2.91E+02(4=)	3.05E+02(6=)	3.07E+02(7+)	3.15E+02(8+)	3.16E+02(9+)	2.81E+02(3=)	2.99E+02(5=)
	Std.Dev	8.19E+01	6.32E+01	8.84E+01	1.09E+02	1.05E+02	9.36E+01	7.68E+01	7.13E+01	7.54E+01
f_{22}	Mean(rank)	6.21E+02(8)	1.24E+02(2-)	7.47E+02(9=)	4.48E+02(7-)	**1.24E+02**(1-)	3.74E+02(6-)	1.25E+02(3-)	3.23E+02(5-)	1.32E+02(4-)
	Std.Dev	2.00E+02	5.68E+01	6.82E+02	1.67E+02	7.08E+00	1.45E+02	1.04E+02	7.80E+01	2.73E+01
f_{23}	Mean(rank)	4.20E+03(4)	4.77E+03(8+)	3.70E+03(2-)	3.72E+03(3-)	4.28E+03(6=)	5.15E+03(9+)	4.21E+03(5=)	**3.66E+03**(1-)	4.43E+03(7=)
	Std.Dev	1.05E+03	1.11E+03	8.14E+02	6.04E+02	6.13E+02	1.41E+03	5.46E+02	7.54E+02	4.79E+02
f_{24}	Mean(rank)	2.14E+02(2)	2.26E+02(7+)	2.22E+02(5+)	2.15E+02(3=)	2.22E+02(6+)	**2.13E+02**(1=)	2.72E+02(9+)	2.17E+02(4+)	2.67E+02(8+)
	Std.Dev	8.19E+00	6.18E+00	9.06E+00	7.11E+00	7.86E+00	8.57E+00	3.70E+00	5.02E+00	8.25E+00
f_{25}	Mean(rank)	**2.48E+02**(1)	2.66E+02(7+)	2.54E+02(3+)	2.48E+02(2=)	2.54E+02(4+)	2.59E+02(6+)	2.75E+02(8+)	2.58E+02(5+)	3.04E+02(9+)
	Std.Dev	7.05E+00	1.21E+01	6.01E+00	6.53E+00	7.54E+00	8.14E+00	7.28E+00	1.40E+01	4.63E+00
f_{26}	Mean(rank)	**2.00E+02**(1)	2.03E+02(8=)	2.05E+02(9=)	2.03E+02(7=)	2.02E+02(6=)	2.00E+02(2=)	2.00E+02(3=)	2.00E+02(4=)	2.00E+02(5+)
	Std.Dev	1.54E-03	1.85E+01	2.30E+01	1.79E+01	1.56E+01	1.56E-03	1.27E-02	9.04E-03	9.72E-02
f_{27}	Mean(rank)	**5.34E+02**(1)	6.24E+02(6+)	5.92E+02(3+)	5.52E+02(2=)	6.17E+02(4+)	6.45E+02(7+)	9.92E+02(8+)	6.23E+02(5+)	1.06E+03(9+)
	Std.Dev	1.17E+02	7.88E+01	1.20E+02	1.04E+02	1.09E+02	1.21E+02	6.33E+01	1.82E+02	5.98E+01
f_{28}	Mean(rank)	3.00E+02(2)	**2.96E+02**(1=)	3.00E+02(2=)	3.00E+02(2=)	3.00E+02(2=)	3.00E+02(2=)	3.00E+02(2=)	3.00E+02(8=)	3.27E+02(9=)
	Std.Dev	0.00E+00	2.80E+01	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	3.93E-11	1.92E+02
(#)	*Best*	7	7	1	4	3	1	6	3	1
(#)	+	-	16	11	4	10	13	10	11	19
(#)	=	-	6	7	14	7	10	8	8	3
(#)	-	-	6	10	10	11	5	10	9	6

on the higher dimensionality case since it attains 8 out of the 28 test functions, followed by EDEV and CoBiDE, who obtain the best mean values on 5 and 4 functions, respectively. Although SaDE offers the same performance as FADE measured by the number of the obtained best results, FADE yields more favorable performance when D=50. Furthermore, comparing Table 2 and Table 3 we can find out that FADE has the best performance on the composition functions with more variables.

Table 3: Comparison results of solution accuracy on CEC2013 test suite (D=50).

		FADE	SaDE	CoDE	CoBiDE	TSDE	EFADE	EDEV	DI-DE	DPADE
f_1	Mean(rank)	**0.00E+00**(1)	**0.00E+00**(1=)	1.43E-13(6+)	**0.00E+00**(1=)	**0.00E+00**(1=)	1.52E-13(7+)	**0.00E+00**(1=)	1.12E-12(9+)	2.27E-13(8+)
	Std.Dev	0.00E+00	0.00E+00	1.11E-13	0.00E+00	0.00E+00	1.08E-13	0.00E+00	8.60E-13	2.55E-29
f_2	Mean(rank)	2.73E+05(3)	8.18E+05(8+)	5.94E+05(6+)	3.41E+05(5+)	2.98E+05(4+)	2.01E+05(2-)	**3.33E+04**(1-)	6.16E+05(7+)	1.15E+07(9+)
	Std.Dev	1.16E+05	2.46E+05	2.21E+05	1.27E+05	1.76E+05	8.81E+04	2.03E+04	2.38E+05	2.32E+06
f_3	Mean(rank)	**7.07E+04**(1)	6.72E+07(7+)	7.42E+07(8+)	2.60E+06(2+)	1.26E+07(4+)	5.12E+06(3+)	1.31E+07(5+)	1.88E+07(6+)	1.18E+08(9+)
	Std.Dev	1.46E+05	7.84E+07	1.91E+08	3.56E+06	1.31E+07	6.34E+06	2.29E+07	1.78E+07	6.63E+07
f_4	Mean(rank)	2.33E+01(5)	3.95E+03(7+)	9.72E+01(6+)	6.74E-01(2-)	**4.94E-02**(1-)	1.54E+01(4-)	7.94E-01(3-)	7.98E+03(8+)	6.93E+04(9+)
	Std.Dev	9.93E+00	1.79E+03	8.38E+01	4.15E+00	5.30E-02	1.10E+01	1.05E+00	8.33E+03	7.67E+03
f_5	Mean(rank)	1.14E-13(4)	**0.00E+00**(1-)	1.72E-13(7=)	1.14E-13(4=)	8.69E-14(3=)	1.14E-13(4=)	**0.00E+00**(1-)	1.19E-10(9+)	1.92E-13(8=)
	Std.Dev	0.00E+00	0.00E+00	9.48E-14	0.00E+00	4.87E-14	0.00E+00	0.00E+00	4.82E-10	5.33E-14
f_6	Mean(rank)	4.22E+01(2)	5.50E+01(9+)	4.39E+01(6=)	4.34E+01(4=)	4.37E+01(5=)	4.34E+01(3=)	**1.77E+01**(1-)	4.60E+01(7+)	4.68E+01(8+)
	Std.Dev	6.61E+00	1.96E+01	1.54E+00	1.56E-08	1.11E+00	7.66E-13	3.99E+00	6.54E-01	1.24E+00
f_7	Mean(rank)	**8.44E+00**(1)	4.84E+01(8+)	3.61E+01(4+)	1.93E+01(3+)	3.89E+01(5+)	1.68E+01(2+)	4.42E+01(6+)	4.48E+01(7+)	1.09E+02(9+)
	Std.Dev	4.76E+00	9.53E+00	1.05E+01	8.67E+00	1.05E+01	6.45E+00	1.26E+01	1.00E+01	9.08E+00
f_8	Mean(rank)	2.11E+01(7)	2.11E+01(6=)	2.14E+01(9+)	2.11E+01(3=)	**2.10E+01**(1=)	2.11E+01(8=)	2.11E+01(2=)	2.11E+01(4=)	2.11E+01(5=)
	Std.Dev	3.32E-02	3.92E-02	3.08E-01	7.50E-02	7.86E-02	3.58E-02	7.69E-02	3.98E-02	4.09E-02
f_9	Mean(rank)	3.16E+01(3)	3.97E+01(7+)	3.12E+01(2=)	**2.81E+01**(1-)	3.30E+01(5+)	3.28E+01(4=)	3.73E+01(6+)	4.77E+01(8=)	5.75E+01(9+)
	Std.Dev	8.76E+00	4.31E+00	5.16E+00	4.83E+00	5.42E+00	4.43E+00	7.53E+00	8.19E+00	2.48E+00
f_{10}	Mean(rank)	**1.57E-02**(1)	2.93E-01(8+)	5.95E-02(5+)	4.75E-02(4+)	4.64E-02(3+)	6.34E-02(6+)	4.34E-02(2+)	1.53E-01(7+)	2.39E+00(9+)
	Std.Dev	1.06E-02	1.67E-01	2.76E-02	2.73E-02	2.60E-02	3.09E-02	2.83E-02	1.41E-01	7.50E-01
f_{11}	Mean(rank)	4.26E+01(9)	2.32E+00(8-)	1.06E+00(7-)	9.67E-09(4-)	6.24E-01(6-)	2.45E-14(2-)	**0.00E+00**(1-)	2.33E-04(5-)	1.93E-12(3-)
	Std.Dev	9.94E+00	2.15E+00	2.51E+00	3.23E-08	7.70E-01	2.84E-14	0.00E+00	3.75E-04	1.76E-12
f_{12}	Mean(rank)	7.09E+01(2)	1.20E+02(8+)	9.26E+01(7+)	8.50E+01(4+)	8.94E+01(5+)	7.24E+01(3=)	9.25E+01(6+)	**5.10E+01**(1=)	3.31E+02(9+)
	Std.Dev	1.16E+01	2.41E+01	1.77E+01	2.22E+01	2.00E+01	1.95E+01	1.78E+01	1.85E+01	3.75E+01
f_{13}	Mean(rank)	**1.39E+02**(1)	2.58E+02(8+)	1.95E+02(6+)	1.64E+02(4+)	1.90E+02(5+)	1.42E+02(2=)	2.02E+02(7+)	1.55E+02(3+)	4.25E+02(9+)
	Std.Dev	2.99E+01	3.74E+01	3.92E+01	4.17E+01	3.77E+01	4.03E+01	3.28E+01	3.01E+01	3.68E+01
f_{14}	Mean(rank)	8.13E+02(6)	6.40E+00(2-)	3.29E+03(9+)	8.68E+02(8=)	2.75E+01(3-)	8.17E+02(7=)	**4.63E-01**(1-)	2.52E+02(5-)	4.01E+01(4-)
	Std.Dev	2.71E+02	4.48E+00	2.40E+03	1.81E+02	1.16E+01	1.45E+02	1.76E+00	4.11E+01	8.00E+00
f_{15}	Mean(rank)	6.95E+03(4)	9.23E+03(8+)	7.01E+03(5=)	**6.42E+03**(1-)	6.76E+03(3=)	1.10E+04(9+)	6.68E+03(2-)	7.37E+03(6+)	7.41E+03(7+)
	Std.Dev	4.85E+02	2.12E+03	7.28E+02	7.29E+02	7.88E+02	3.19E+03	6.81E+02	1.44E+03	4.71E+02
f_{16}	Mean(rank)	3.26E+00(7)	3.31E+00(8=)	2.78E+00(6=)	**3.28E-01**(1-)	8.53E-01(2-)	3.35E+00(9=)	1.05E+00(3-)	1.69E+00(5=)	1.46E+00(4-)
	Std.Dev	3.12E-01	2.99E-01	4.20E+00	5.37E-01	4.31E-01	2.70E-01	5.71E-01	4.80E-01	2.33E-01
f_{17}	Mean(rank)	9.59E+01(8)	5.12E+01(2-)	1.27E+02(9+)	8.47E+01(6-)	5.34E+01(4-)	9.59E+01(7=)	**5.08E+01**(1-)	5.65E+01(5-)	5.17E+01(3-)
	Std.Dev	1.07E+01	3.87E-01	3.91E+01	3.85E+00	8.72E-01	3.20E+00	3.08E-04	9.48E-01	1.48E-01
f_{18}	Mean(rank)	3.56E+02(8)	1.63E+02(6-)	1.25E+02(4-)	**1.14E+02**(1-)	1.19E+02(2-)	3.10E+02(7-)	1.19E+02(3-)	1.36E+02(5-)	4.11E+02(9+)
	Std.Dev	2.33E+01	6.90E+01	1.93E+01	1.82E+01	1.68E+01	1.16E+02	1.48E+01	5.70E+01	3.29E+01
f_{19}	Mean(rank)	4.49E+00(4)	1.16E+01(8+)	1.62E+01(9+)	4.44E+00(3=)	3.34E+00(2-)	1.06E+01(7+)	7.47E+00(6+)	4.60E+00(5-)	**2.59E+00**(1-)
	Std.Dev	2.09E+00	2.73E+00	1.19E+01	1.54E+00	5.91E-01	7.28E-01	5.14E+00	6.81E-01	2.95E-01
f_{20}	Mean(rank)	2.03E+01(4)	2.03E+01(5=)	**1.99E+01**(1-)	2.02E+01(2=)	2.03E+01(3=)	2.16E+01(8+)	2.04E+01(6=)	2.08E+01(7=)	2.17E+01(8+)
	Std.Dev	3.61E-01	7.01E-01	7.18E-01	7.43E-01	9.09E-01	5.78E-01	7.67E-01	3.72E-01	5.35E-01
f_{21}	Mean(rank)	5.48E+02(3)	8.73E+02(9+)	6.31E+02(5=)	5.78E+02(4=)	6.72E+02(6=)	**2.80E+02**(1-)	7.46E+02(7+)	3.27E+02(2-)	8.12E+02(8+)
	Std.Dev	4.18E+02	3.46E+02	4.14E+02	3.20E+02	4.56E+02	2.26E+02	4.20E+02	3.21E+02	3.71E+02
f_{22}	Mean(rank)	1.04E+03(6)	**2.44E+01**(1-)	2.32E+03(9+)	1.17E+03(7+)	6.38E+01(2-)	1.65E+03(8+)	2.09E+02(4-)	4.58E+02(5-)	9.55E+01(3-)
	Std.Dev	2.89E+02	1.16E+01	1.20E+03	3.10E+02	5.05E+01	5.73E+02	7.60E+02	2.06E+02	2.55E+01
f_{23}	Mean(rank)	9.29E+03(7)	8.51E+03(6-)	7.33E+03(2-)	**6.46E+03**(1-)	7.51E+03(4-)	1.09E+04(9+)	7.63E+03(5-)	7.34E+03(3-)	9.53E+03(8=)
	Std.Dev	2.91E+03	2.11E+03	7.80E+02	8.93E+02	9.29E+02	2.87E+03	8.04E+02	1.24E+03	5.97E+02
f_{24}	Mean(rank)	**2.30E+02**(1)	2.77E+02(7+)	2.62E+02(4+)	2.33E+02(2=)	2.65E+02(6+)	2.43E+02(3+)	3.45E+02(9+)	2.63E+02(5+)	3.45E+02(8+)
	Std.Dev	1.55E+01	1.09E+01	1.19E+01	1.43E+01	1.02E+01	1.20E+01	7.07E+00	9.63E+00	9.25E+00
f_{25}	Mean(rank)	**2.85E+02**(1)	3.46E+02(7+)	3.13E+02(3+)	2.99E+02(2+)	3.15E+02(4+)	3.21E+02(5+)	3.48E+02(8+)	3.25E+02(6+)	4.07E+02(9+)
	Std.Dev	1.07E+01	1.07E+01	1.12E+01	1.14E+01	1.40E+01	1.43E+01	8.48E+00	1.71E+01	7.39E+00
f_{26}	Mean(rank)	2.04E+02(2)	2.72E+02(5+)	2.91E+02(8+)	2.52E+02(4+)	2.74E+02(6+)	2.13E+02(3=)	2.75E+02(7+)	3.33E+02(9+)	**2.01E+02**(=)
	Std.Dev	2.75E+01	9.03E+01	9.06E+01	7.44E+01	8.56E+01	4.65E+01	1.06E+02	9.27E+01	3.94E-01
f_{27}	Mean(rank)	**8.59E+02**(1)	1.24E+03(6+)	1.12E+03(5+)	9.65E+02(2+)	1.07E+03(4+)	1.05E+03(3+)	1.72E+03(8+)	1.26E+03(7+)	1.85E+03(9+)
	Std.Dev	1.54E+02	9.71E+01	1.34E+02	1.12E+02	1.03E+02	1.61E+02	6.56E+01	2.06E+02	5.87E+01
f_{28}	Mean(rank)	**4.00E+02**(1)	5.95E+02(9=)	4.59E+02(7=)	4.00E+02(4=)	**4.00E+02**(1=)	4.00E+02(3=)	5.82E+02(8+)	4.00E+02(6=)	4.00E+02(5=)
	Std.Dev	0.00E+00	7.86E+02	4.25E+02	3.57E-13	0.00E+00	2.43E-13	7.36E+02	4.28E-08	9.28E-11
(#)	*Best*	8	3	1	5	4	1	7	1	2
(#)	+	-	16	17	10	11	12	13	14	17
(#)	=	-	5	7	10	8	11	3	6	5
(#)	-	-	7	4	8	9	5	12	8	6

Although CoDE applies three candidate breeding strategies (i.e., mutation strategies and control parameters) for each individual, which is similar to FADE, the fitness-based adaptive selection strategy in FADE yields more pleasurable performance than the random selection strategy in CoDE.

Table 4: Comparison results of solution accuracy on CEC2017 test suite (D=30).

		FADE	SaDE	CoDE	CoBiDE	TSDE	EFADE	EDEV	DI-DE	DPADE
F_1	Mean	1.34E-14(3)	1.62E+03(8+)	1.45E-14(4=)	7.88E-13(7=)	9.62E-15(2=)	1.50E-14(5=)	**8.36E-16**(1=)	2.26E-14(6+)	2.23E+03(9+)
	Std.Dev	1.59E-14	1.49E+03	1.12E-14	4.65E-12	8.36E-15	8.24E-15	3.38E-15	1.03E-14	3.03E+03
F_2	Mean	**0.00E+00**(1)	**0.00E+00**(1=)	1.02E+00(6+)	**0.00E+00**(1=)	**0.00E+00**(1=)	**0.00E+00**(1=)	2.73E+00(7+)	6.05E+08(8+)	2.35E+11(9+)
	Std.Dev	0.00E+00	0.00E+00	1.19E+00	0.00E+00	0.00E+00	0.00E+00	1.95E+01	4.32E+09	9.43E+11
F_3	Mean	1.06E-10(6)	7.52E-03(7+)	3.15E-13(4-)	**0.00E+00**(1-)	**0.00E+00**(1-)	2.22E-13(3-)	4.24E-11(5=)	4.05E+02(8+)	8.93E+02(9+)
	Std.Dev	2.01E-10	2.00E-02	8.99E-13	0.00E+00	0.00E+00	2.63E-13	3.03E-10	1.31E+03	6.34E+03
F_4	Mean	3.21E+01(4)	3.66E+01(5=)	3.11E+01(3=)	2.22E+01(2=)	5.04E+01(7+)	3.74E+01(6=)	**1.02E+00**(1-)	6.53E+01(8+)	9.18E+01(9+)
	Std.Dev	2.98E+01	3.51E+01	2.96E+01	2.67E+01	2.25E+01	2.91E+01	1.75E+00	1.34E+01	1.14E+01
F_5	Mean	**2.68E+01**(1)	3.71E+01(4+)	3.92E+01(7+)	4.73E+01(9+)	3.91E+01(6+)	4.20E+01(8+)	3.05E+01(2+)	3.24E+01(3+)	3.82E+01(5+)
	Std.Dev	5.37E+00	9.20E+00	9.19E+00	1.41E+01	1.14E+01	1.32E+01	7.40E+00	1.17E+01	1.41E+01
F_6	Mean	1.24E-01(9)	3.38E-05(7-)	1.67E-13(5-)	2.82E-07(6-)	1.11E-13(2-)	1.25E-13(4-)	**1.09E-13**(1-)	6.39E-05(8-)	1.14E-13(3-)
	Std.Dev	1.68E-01	2.42E-04	5.73E-14	2.01E-06	1.59E-14	3.41E-14	2.23E-14	9.80E-05	1.27E-29
F_7	Mean	6.50E+01(3)	**6.40E+01**(1=)	6.74E+01(5=)	8.04E+01(8+)	6.88E+01(6+)	9.51E+01(9+)	6.42E+01(2=)	6.69E+01(4+)	7.73E+01(7+)
	Std.Dev	8.62E+00	9.98E+00	1.02E+01	1.43E+01	1.03E+01	2.54E+01	5.25E+00	1.16E+01	1.96E+01
F_8	Mean	2.91E+01(2)	3.86E+01(4+)	3.94E+01(5+)	4.85E+01(9+)	3.95E+01(6+)	4.54E+01(8+)	**2.76E+01**(1=)	3.34E+01(3=)	4.21E+01(7+)
	Std.Dev	7.54E+00	9.32E+00	1.20E+01	1.19E+01	1.27E+01	1.36E+01	7.96E+00	9.82E+00	1.45E+01
F_9	Mean	5.46E+00(9)	2.12E+00(8-)	9.80E-02(4-)	1.21E-01(5-)	4.45E-02(3-)	2.23E-14(2-)	1.69E-01(6-)	2.10E-01(7-)	**0.00E+00**(1-)
	Std.Dev	9.74E+00	3.90E+00	3.50E-01	4.44E-01	1.36E-01	4.56E-14	3.49E-01	2.58E-01	0.00E+00
F_{10}	Mean	3.17E+03(6)	3.43E+03(8=)	1.96E+03(2-)	2.41E+03(4-)	**1.94E+03**(1-)	3.57E+03(9+)	2.16E+03(3-)	2.66E+03(5-)	3.19E+03(7=)
	Std.Dev	8.00E+02	6.78E+02	5.30E+02	5.42E+02	4.35E+02	7.36E+02	4.88E+02	6.31E+02	8.98E+02
F_{11}	Mean	1.21E+01(2)	7.52E+01(9+)	2.23E+01(6+)	2.51E+01(7+)	1.70E+01(4=)	1.30E+01(3=)	1.88E+01(5+)	**9.46E+00**(1=)	2.78E+01(8+)
	Std.Dev	1.05E+01	3.19E+01	2.06E+01	1.68E+01	1.66E+01	4.99E+00	7.54E+00	1.12E+01	2.43E+01
F_{12}	Mean	**1.62E+03**(1)	1.67E+04(9+)	8.30E+03(5+)	1.05E+04(6+)	8.27E+03(4+)	7.33E+03(3+)	5.72E+03(2+)	1.34E+04(7+)	1.39E+04(8+)
	Std.Dev	1.47E+03	7.46E+03	5.87E+03	9.91E+03	9.72E+03	7.21E+03	5.78E+03	1.33E+04	9.84E+03
F_{13}	Mean	1.52E+03(8)	7.35E+03(9+)	5.66E+01(5=)	1.65E+02(7=)	**2.88E+01**(1=)	3.29E+01(2=)	4.93E+01(4=)	4.73E+01(3=)	1.16E+02(6=)
	Std.Dev	5.52E+03	6.17E+03	9.96E+01	2.84E+02	1.28E+01	2.70E+01	8.06E+01	2.65E+01	1.84E+02
F_{14}	Mean	2.42E+01(7)	7.36E+01(9+)	1.32E+01(3-)	2.51E+01(8=)	1.23E+01(2-)	**1.11E+01**(1-)	1.49E+01(4-)	2.39E+01(6-)	1.83E+01(5-)
	Std.Dev	1.14E+01	2.73E+01	8.85E+00	1.19E+01	7.91E+00	4.26E+00	1.08E+01	5.96E+00	9.99E+00
F_{15}	Mean	1.49E+01(6)	1.22E+02(9+)	1.13E+01(3=)	1.75E+01(8=)	8.58E+00(2-)	**7.91E+00**(1-)	1.17E+01(4=)	1.23E+01(5=)	1.70E+01(7=)
	Std.Dev	2.14E+01	1.05E+02	5.09E+00	1.05E+01	3.81E+00	3.22E+00	1.02E+01	3.03E+00	1.39E+01
F_{16}	Mean	5.13E+02(8)	4.35E+02(3-)	4.84E+02(5=)	5.12E+02(7=)	5.48E+02(9=)	4.47E+02(4=)	4.99E+02(6=)	**2.09E+02**(1-)	3.03E+02(2-)
	Std.Dev	1.99E+02	1.38E+02	2.26E+02	1.84E+02	1.96E+02	1.76E+02	1.70E+02	1.37E+02	1.45E+02
F_{17}	Mean	1.30E+02(9)	4.65E+01(2-)	5.19E+01(5-)	7.96E+01(8-)	7.18E+01(7-)	6.17E+01(6-)	4.69E+01(3-)	4.99E+01(4-)	**3.95E+01**(1-)
	Std.Dev	1.10E+02	2.60E+01	5.14E+01	7.92E+01	7.60E+01	3.89E+01	2.93E+01	1.10E+01	2.72E+01
F_{18}	Mean	**3.61E+01**(1)	5.79E+03(9+)	1.38E+02(5+)	4.31E+02(8+)	8.45E+01(3+)	9.96E+01(4+)	5.25E+01(2=)	1.41E+02(6+)	2.04E+02(7+)
	Std.Dev	9.28E+00	5.69E+03	1.51E+02	5.91E+02	6.06E+01	1.11E+02	1.34E+02	2.35E+02	2.77E+02
F_{19}	Mean	1.05E+01(7)	4.51E+01(9+)	**5.54E+00**(1-)	8.40E+00(5=)	5.92E+00(2-)	6.35E+00(3-)	7.71E+00(4-)	1.12E+01(8=)	8.85E+00(6=)
	Std.Dev	1.09E+01	3.21E+01	1.73E+00	3.29E+00	1.83E+00	2.27E+00	4.99E+00	2.08E+00	9.17E+00
F_{20}	Mean	1.69E+02(9)	6.64E+01(4-)	1.14E+02(7-)	1.11E+02(6-)	1.10E+02(5-)	1.16E+02(8-)	3.31E+01(2-)	**2.76E+01**(1-)	4.71E+01(3-)
	Std.Dev	1.09E+02	6.14E+01	9.15E+01	8.86E+01	8.99E+01	6.08E+01	4.91E+01	1.11E+01	5.84E+01
F_{21}	Mean	**2.31E+02**(1)	2.36E+02(3+)	2.38E+02(5+)	2.53E+02(9+)	2.40E+02(6+)	2.42E+02(7+)	2.36E+02(4+)	2.33E+02(2+)	2.43E+02(8+)
	Std.Dev	6.61E+00	8.34E+00	9.27E+00	1.47E+01	8.80E+00	1.10E+01	7.86E+00	1.04E+01	1.34E+01
F_{22}	Mean	1.01E+02(7)	1.01E+02(6-)	1.00E+02(3-)	1.68E+02(8+)	1.00E+02(5=)	1.00E+02(2-)	2.79E+02(9+)	1.00E+02(4-)	**1.00E+02**(1-)
	Std.Dev	1.98E+00	1.10E+00	5.01E-13	4.85E+02	0.00E+00	4.49E-13	6.20E+02	1.50E-12	1.39E-13
F_{23}	Mean	**3.78E+02**(1)	3.83E+02(4+)	3.85E+02(5+)	3.97E+02(9+)	3.86E+02(6+)	3.92E+02(8+)	3.83E+02(3+)	3.79E+02(2=)	3.87E+02(7+)
	Std.Dev	8.87E+00	1.17E+01	1.08E+01	1.18E+01	9.14E+00	1.21E+01	8.17E+00	9.39E+00	1.22E+01
F_{24}	Mean	4.56E+02(4)	4.51E+02(2-)	4.56E+02(5=)	4.69E+02(8+)	4.60E+02(6=)	4.70E+02(9+)	4.53E+02(3=)	**4.47E+02**(1=)	4.62E+02(7+)
	Std.Dev	1.02E+01	1.14E+01	9.81E+00	1.13E+01	1.10E+01	1.35E+01	7.43E+00	7.27E+00	1.35E+01
F_{25}	Mean	3.87E+02(2)	3.92E+02(9+)	3.87E+02(5=)	3.87E+02(6=)	3.87E+02(4=)	3.87E+02(3=)	**3.83E+02**(1=)	3.87E+02(7+)	3.87E+02(8+)
	Std.Dev	1.23E+00	1.15E+01	1.03E-01	1.73E-01	5.13E-01	3.79E-02	6.63E+00	1.91E-01	3.69E-01
F_{26}	Mean	1.24E+03(3)	1.29E+03(4=)	1.37E+03(7+)	1.51E+03(9+)	1.39E+03(8+)	**1.07E+03**(1-)	1.31E+03(5+)	1.23E+03(2-)	1.32E+03(6=)
	Std.Dev	2.79E+02	4.41E+02	2.17E+02	1.43E+02	1.26E+02	5.37E+02	9.85E+01	9.32E+01	9.88E+01
F_{27}	Mean	5.02E+02(7)	5.17E+02(9+)	5.01E+02(5=)	5.07E+02(8+)	4.96E+02(3-)	4.98E+02(2-)	5.00E+02(4=)	**4.98E+02**(1=)	5.02E+02(6=)
	Std.Dev	7.90E+00	8.14E+00	7.45E+00	7.65E+00	7.98E+00	7.27E+00	1.38E-04	6.58E+00	6.22E+00
F_{28}	Mean	3.24E+02(2)	3.24E+02(3=)	3.30E+02(4=)	3.34E+02(6=)	3.31E+02(5=)	**3.14E+02**(1=)	4.44E+02(9+)	3.42E+02(7+)	3.74E+02(8+)
	Std.Dev	4.58E+01	4.54E+01	5.04E+01	5.33E+01	5.14E+01	3.59E+01	7.95E+01	5.17E+01	5.42E+01
F_{29}	Mean	4.30E+02(3)	4.49E+02(7+)	4.32E+02(4=)	5.08E+02(9+)	4.61E+02(8+)	4.40E+02(6=)	**4.09E+02**(1-)	4.38E+02(5=)	4.18E+02(2=)
	Std.Dev	3.24E+01	2.67E+01	5.28E+01	8.99E+01	8.11E+01	5.17E+01	6.59E+01	3.78E+01	4.12E+01
F_{30}	Mean	2.00E+03(2)	2.96E+03(9+)	2.09E+03(3+)	2.15E+03(5+)	2.09E+03(4+)	2.27E+03(7+)	**2.23E+02**(1-)	2.18E+03(6+)	2.29E+03(8+)
	Std.Dev	5.46E+01	7.69E+02	1.01E+02	1.34E+02	9.79E+01	1.94E+02	2.19E+01	1.66E+02	2.99E+02
(#)	*Best*	6	2	1	2	4	5	7	5	3
(#)	+	-	17	10	14	11	10	9	12	14
(#)	=	-	6	11	10	9	9	11	9	9
(#)	-	-	7	9	6	10	11	10	9	7

4.2.2. Experimental results on CEC2017 test suite

The comparison results presented in Table 4 verify that EDEV and FADE display more promising performance on CEC2017 test suite with 30 variables, measured by the number of archived best solutions ((#)*Best*), followed by E-FADE and DI-DE. While the number of variables increasing from 30 to 50, FADE and DI-DE yield more promising scalability than other 7 peer algorithm-s. Although FADE does not display outstanding performance on the unimodal

Table 5: Comparison results of solution accuracy on CEC2017 test suite (D=50).

		FADE	SaDE	CoDE	CoBiDE	TSDE	EFADE	EDEV	DI-DE	DPADE
F_1	Mean	1.40E+01(5)	1.17E+03(6+)	2.99E+03(7+)	6.76E+00(4=)	1.65E-04(2-)	8.41E-01(3-)	**9.34E-12**(1-)	4.96E+03(9+)	3.00E+03(8+)
	Std.Dev	2.51E+01	1.46E+03	2.92E+03	4.38E+01	6.83E-04	1.68E+00	3.23E-11	5.08E+03	3.37E+03
F_2	Mean	1.34E+06(2)	8.07E+11(6+)	1.23E+33(8+)	3.05E+10(5+)	1.75E+06(3=)	**0.00E+00**(1-)	3.35E+09(4=)	8.28E+29(7=)	1.31E+38(9=)
	Std.Dev	8.67E+06	1.95E+12	6.67E+33	1.35E+11	6.77E+06	0.00E+00	1.57E+10	5.88E+30	7.77E+38
F_3	Mean	4.35E-01(5)	1.94E+02(6+)	2.21E+02(7+)	2.04E-07(2-)	**2.53E-08**(1-)	7.75E-05(4-)	1.58E-05(3-)	6.77E+03(8+)	1.01E+04(9+)
	Std.Dev	5.49E-01	1.73E+02	1.86E+02	1.94E-07	3.11E-08	2.28E-04	5.43E-05	8.85E+03	3.36E+04
F_4	Mean	5.85E+01(3)	1.03E+02(7+)	9.05E+01(6+)	6.84E+01(4=)	6.94E+01(5+)	4.20E+01(2-)	**1.05E+00**(1-)	1.29E+02(9+)	1.07E+02(8+)
	Std.Dev	4.32E+01	4.25E+01	5.45E+01	4.44E+01	3.79E+01	3.03E+01	1.96E+00	3.84E+01	5.14E+01
F_5	Mean	9.05E+01(6)	1.99E+02(8+)	4.65E+02(9+)	8.77E+01(5=)	**7.53E+01**(1-)	9.56E+01(7=)	8.50E+01(3=)	7.82E+01(2-)	8.55E+01(4=)
	Std.Dev	5.07E+01	1.31E+01	1.10E+02	2.03E+01	1.70E+01	2.61E+01	1.55E+01	2.46E+01	2.68E+01
F_6	Mean	4.59E-03(7)	**1.11E-13**(1-)	7.92E-03(9+)	1.77E-08(4-)	2.02E-08(5-)	1.83E-13(3-)	7.22E-04(6-)	5.46E-03(8=)	1.14E-13(2-)
	Std.Dev	9.08E-03	1.59E-14	3.39E-03	3.88E-08	2.63E-08	5.61E-14	5.15E-03	5.37E-03	1.27E-29
F_7	Mean	2.07E+02(7)	2.70E+02(8+)	5.58E+02(9+)	1.37E+02(5-)	1.22E+02(2-)	1.77E+02(6-)	1.28E+02(3-)	**1.19E+02**(1-)	1.35E+02(4-)
	Std.Dev	5.23E+01	1.32E+01	7.94E+01	2.14E+01	1.60E+01	7.00E+01	1.40E+01	2.34E+01	3.28E+01
F_8	Mean	1.15E+02(7)	2.00E+02(8+)	4.61E+02(9+)	8.96E+01(5-)	7.33E+01(2-)	9.09E+01(6-)	7.41E+01(3-)	**6.95E+01**(1-)	8.66E+01(4-)
	Std.Dev	5.28E+01	1.28E+01	1.08E+02	1.99E+01	1.62E+01	1.94E+01	1.36E+01	2.12E+01	3.08E+01
F_9	Mean	1.35E+00(6)	2.26E+00(7+)	8.35E-02(4-)	1.77E-02(2-)	7.82E-02(3-)	**5.27E-03**(1-)	6.95E+00(9+)	4.41E+00(8+)	3.75E-01(5-)
	Std.Dev	2.47E+00	1.72E+00	2.67E-01	7.02E-02	1.93E-01	2.13E-02	8.32E+00	3.29E+00	6.50E-01
F_{10}	Mean	**4.19E+03**(1)	8.74E+03(8+)	1.71E+04(9+)	4.24E+03(3=)	4.41E+03(4+)	8.02E+03(7+)	4.23E+03(2=)	5.27E+03(5+)	6.57E+03(6+)
	Std.Dev	9.19E+02	3.02E+02	1.77E+03	7.23E+02	5.72E+02	1.61E+03	6.87E+02	1.04E+03	1.48E+03
F_{11}	Mean	4.16E+01(2)	1.12E+02(9+)	6.20E+01(6+)	5.14E+01(5+)	4.50E+01(3+)	4.66E+01(4+)	7.65E+01(8+)	**3.62E+01**(1-)	6.90E+01(7+)
	Std.Dev	6.77E+00	2.49E+01	5.24E+01	1.02E+01	7.58E+00	7.34E+00	3.14E+01	3.89E+00	1.94E+01
F_{12}	Mean	**1.01E+04**(1)	6.29E+04(4+)	2.53E+05(9+)	7.53E+04(6+)	4.57E+04(2+)	4.83E+04(3+)	6.42E+04(5+)	1.02E+05(7+)	1.29E+05(8+)
	Std.Dev	1.29E+04	3.85E+04	1.72E+05	4.27E+04	3.09E+04	2.97E+04	7.69E+04	6.55E+04	1.23E+05
F_{13}	Mean	**2.96E+02**(1)	5.42E+02(2+)	6.09E+02(3+)	1.04E+03(6+)	6.68E+02(4+)	2.71E+03(9+)	1.37E+03(7+)	8.47E+02(5=)	2.25E+03(8+)
	Std.Dev	2.91E+02	6.52E+02	3.77E+02	1.79E+03	8.28E+02	2.51E+03	2.83E+03	2.44E+03	2.38E+03
F_{14}	Mean	**4.22E+01**(1)	1.17E+02(8+)	1.05E+02(7+)	4.55E+01(4=)	4.44E+01(3=)	4.42E+01(2=)	1.76E+03(9+)	7.51E+01(5+)	8.21E+01(6+)
	Std.Dev	5.11E+00	3.00E+01	8.83E+01	1.27E+01	6.45E+00	1.36E+01	7.52E+03	2.03E+01	3.95E+01
F_{15}	Mean	2.07E+03(9)	1.30E+02(6-)	1.09E+02(4-)	5.68E+01(3-)	**4.41E+01**(1-)	1.20E+02(5-)	1.31E+02(7-)	5.59E+01(2-)	1.07E+03(8=)
	Std.Dev	3.48E+03	5.99E+01	7.69E+01	2.47E+01	2.00E+01	2.36E+02	1.90E+02	1.93E+01	1.69E+03
F_{16}	Mean	8.73E+02(3)	1.03E+03(8+)	1.62E+03(9+)	9.48E+02(6=)	1.00E+03(7+)	8.99E+02(4=)	9.40E+02(5=)	**7.75E+02**(1=)	8.45E+02(2=)
	Std.Dev	2.39E+02	1.40E+02	1.52E+03	2.76E+02	3.20E+02	2.06E+02	2.81E+02	2.73E+02	2.07E+02
F_{17}	Mean	4.87E+02(3)	7.87E+02(8+)	2.01E+03(9+)	5.83E+02(5+)	5.86E+02(6+)	6.02E+02(7+)	5.60E+02(4+)	**3.58E+02**(1-)	4.58E+02(2=)
	Std.Dev	1.80E+02	1.10E+02	1.45E+03	2.12E+02	1.92E+02	1.38E+02	1.89E+02	1.65E+02	1.83E+02
F_{18}	Mean	**6.18E+01**(1)	4.60E+02(3+)	6.92E+02(5+)	9.83E+02(6+)	3.52E+02(2+)	3.02E+03(7+)	6.79E+02(4+)	1.62E+04(9+)	6.37E+03(8+)
	Std.Dev	1.95E+01	4.58E+02	4.01E+02	7.08E+02	3.03E+02	3.19E+03	1.35E+03	1.06E+04	6.53E+03
F_{19}	Mean	2.46E+01(4)	3.06E+01(6+)	8.72E+01(8+)	1.73E+01(2-)	**1.37E+01**(1-)	2.17E+01(3=)	4.56E+01(7+)	2.72E+01(5+)	1.04E+03(9+)
	Std.Dev	6.04E+00	1.27E+01	1.05E+02	5.12E+00	3.88E+00	2.47E+01	3.20E+01	6.85E+00	2.63E+03
F_{20}	Mean	3.70E+02(3)	5.51E+02(8+)	5.93E+02(9+)	4.06E+02(6=)	4.15E+02(7+)	3.95E+02(4=)	4.01E+02(5=)	**3.12E+02**(1=)	3.62E+02(2=)
	Std.Dev	1.48E+02	1.21E+02	4.95E+02	1.77E+02	1.56E+02	1.67E+02	1.64E+02	1.48E+02	1.92E+02
F_{21}	Mean	**2.67E+02**(1)	3.90E+02(8+)	6.59E+02(9+)	2.86E+02(5+)	2.70E+02(3=)	2.91E+02(6+)	2.73E+02(4=)	2.70E+02(2=)	2.97E+02(7+)
	Std.Dev	1.36E+01	1.27E+01	1.16E+02	1.78E+01	1.68E+01	2.10E+01	1.42E+01	2.19E+01	3.48E+01
F_{22}	Mean	4.16E+03(4)	**1.47E+03**(1-)	1.43E+04(9+)	3.77E+03(2=)	4.28E+03(6=)	6.44E+03(8+)	4.24E+03(5=)	5.42E+03(7+)	3.81E+03(3=)
	Std.Dev	2.57E+03	3.23E+03	6.29E+03	1.96E+03	1.45E+03	3.78E+03	1.24E+03	1.29E+03	3.15E+03
F_{23}	Mean	**4.89E+02**(1)	6.18E+02(8+)	8.89E+02(9+)	5.15E+02(5+)	4.97E+02(4+)	5.18E+02(6+)	4.92E+02(3=)	4.91E+02(2=)	5.21E+02(7+)
	Std.Dev	2.14E+01	3.19E+01	1.12E+02	2.11E+01	1.87E+01	2.67E+01	2.42E+01	2.03E+01	3.40E+01
F_{24}	Mean	5.71E+02(5)	**5.48E+02**(1-)	5.59E+02(3-)	5.82E+02(7+)	5.59E+02(4-)	5.84E+02(8+)	5.71E+02(6=)	5.53E+02(2-)	5.95E+02(9+)
	Std.Dev	2.53E+01	1.45E+01	2.28E+01	1.93E+01	1.52E+01	1.78E+01	1.45E+01	1.54E+01	4.12E+01
F_{25}	Mean	5.17E+02(7)	5.57E+02(9=)	5.09E+02(6=)	5.05E+02(4=)	5.07E+02(5=)	4.89E+02(3-)	**4.46E+02**(1-)	4.82E+02(2-)	5.25E+02(8=)
	Std.Dev	3.70E+01	3.17E+01	3.41E+01	3.13E+01	3.31E+01	1.92E+01	2.01E+01	5.15E+00	3.53E+01
F_{26}	Mean	1.80E+03(2)	2.56E+03(8+)	4.60E+03(9+)	2.02E+03(6+)	1.84E+03(3=)	2.08E+03(7+)	2.00E+03(5+)	**1.72E+03**(1-)	1.98E+03(4+)
	Std.Dev	1.91E+02	4.76E+02	1.49E+03	2.31E+02	2.14E+02	2.03E+02	1.69E+02	1.59E+02	2.34E+02
F_{27}	Mean	5.10E+02(3)	5.63E+02(9+)	5.08E+02(2=)	5.31E+02(6+)	5.12E+02(4=)	5.29E+02(5+)	**5.00E+02**(1-)	5.32E+02(7+)	5.45E+02(8+)
	Std.Dev	1.24E+01	2.80E+01	8.18E+00	1.88E+01	9.54E+00	1.07E+01	2.28E-04	1.18E+01	1.42E+01
F_{28}	Mean	4.77E+02(6)	5.19E+02(9+)	4.63E+02(4-)	4.74E+02(5=)	4.63E+02(3-)	**4.59E+02**(1-)	4.93E+02(7+)	4.60E+02(2-)	4.95E+02(8+)
	Std.Dev	2.24E+01	2.81E+01	1.35E+01	2.16E+01	1.33E+01	5.31E-13	1.47E+01	6.84E+00	2.03E+01
F_{29}	Mean	4.49E+02(4)	6.51E+02(8+)	2.70E+03(9+)	5.28E+02(7+)	4.39E+02(3=)	4.84E+02(5=)	5.02E+02(6+)	4.37E+02(2=)	**4.12E+02**(1=)
	Std.Dev	1.02E+02	7.66E+01	2.16E+03	1.59E+02	1.13E+02	1.13E+02	9.76E+01	8.61E+01	1.04E+02
F_{30}	Mean	5.87E+05(2)	6.92E+05(9+)	6.16E+05(6+)	5.97E+05(4+)	5.93E+05(3+)	6.11E+05(5+)	**4.99E+02**(1-)	6.21E+05(7+)	6.47E+05(8+)
	Std.Dev	1.20E+04	6.45E+04	2.88E+04	2.59E+04	2.61E+04	5.22E+04	3.42E+02	4.46E+04	6.45E+04
(#)	*Best*	7	3	0	0	4	3	5	7	1
(#)	+	-	25	24	13	11	13	11	12	17
(#)	=	-	1	2	10	8	6	9	8	9
(#)	-	-	3	4	7	11	11	10	10	4

functions with 50 variables, it offers the most promising performance on the multimodal functions, especially on the complicated functions (i.e., hybrid functions and composition functions). Concretely, FADE attains 6 out of the 20 complicated functions, followed by DI-DE and EDEV.

From the comparison results of solution accuracy listed in Table 2, Table 3, Table 4, and Table 5, we can see that FADE not only offers the most promising performance, but also yields very reliable scalability.

4.3. Results of statistic tests

It is a widely accepted method that using statistic tests to analyze experimental results in the field of computational intelligence [10]. In this section, three popular statistic tests, i.e., the two-tailed t-test, the Wilcoxon test, and the Friedman-test with a significance level of α=0.5, are used to testify the performance of all the algorithms.

4.3.1. t-test results

The t-test results between FADE and other 8 peer algorithms are presented in Table 6 aiming to provide a comprehensive performance (*CP*) of all the algorithms. Note that the value of *CP* is equal to "(#)+" minus "(#)-" on the two test suites. From the results listed in Table 6 we can see that FADE has the best performance measured by the values of *CP*. Concretely, FADE is slightly better than EDEV and TSDE on the two test suites. Although FADE and EFADE archive almost the same performance on CEC2017 test suite, FADE significantly dominates EFADE on CEC2013 test suite. On the contrary, FADE significantly surpasses CoBiDE on CEC2017 though CoBiDE displays more favorable performance on CEC2013. Furthermore, we can find out that FADE significantly dominates SaDE, CoDE, and DPADE on both CEC2013 and CEC2017 test suites. From the values of *CP* we can see that FADE displays the most oustanding performance while SaDE, DPADE, and CoDE yield unfavorable performance.

Table 6: Statistical results of t-test between FADE and other 8 competitors on CEC2013 and CEC2017 test suites.

	(#)+		(#)-		
FADE vs.	CEC2013	CEC2017	CEC2013	CEC2017	*CP*
SaDE	32	42	13	10	51
CoDE	28	34	14	13	35
CoBiDE	14	27	18	13	10
TSDE	21	22	20	21	2
EFADE	25	23	10	22	16
EDEV	23	20	22	20	1
DI-DE	25	24	17	19	13
DPADE	36	31	12	11	44

4.3.2. Wilcoxon-test results

In this part, the Wilcoxon signed ranks test at a 0.05 significance level is used to check the difference between FADE and other competitors for the two test suites, the results of which are presented in Table 7 and Table 8, respectively. Note that there is a significant difference between two algorithms only when the value of p is less than 0.05.

From Table 7 we can see that FADE is significantly better than SaDE and DPADE when D=30. Although FADE does not significantly better than other 6 peer algorithms in terms of the p value, it offers more promising performance than CoDE, CoBiDE, EFADE, and DI-DE measured by the number of R^+ and R^-. When D=50, FADE offers more favorable performance. Concretely, it significantly dominates 3 peer algorithms (i.e., SaDE, CoDE, and DPADE) measured by p. Furthermore, FADE overcomes all the competitors, in terms of the number of R^+ and R^-. From the results demonstrated in Table 8 we also find out that FADE displays more promising performance on the higher dimension case then the lower dimension cases of CEC2017 test suite. Thus, a preliminary conclusion can be obtained from Table 7 and Table 8 that FADE can achieve more favorable characteristics on higher dimension functions.

Table 7: Wilcoxon-test results between FADE and the other 8 competitors on CEC2013 test suite with the two dimension cases.

	D=30			D=50		
FADE vs.	R^+	R^-	p	R^+	R^-	p
SaDE	18	9	**0.049**	18	7	**0.050**
CoDE	15	12	0.494	22	6	**0.005**
CoBiDE	14	10	0.689	14	10	0.331
TSDE	12	13	0.778	13	11	0.804
EFADE	17	8	0.192	19	5	0.067
EDEV	12	12	0.648	14	12	0.990
DI-ED	13	12	0.798	19	9	0.439
DPADE	20	7	**0.044**	19	7	**0.015**

4.3.3. Friedman-test results

In this section, Friedman-test, which is a widely used nonparametric test in EA community, is carried out to validate the performance of all the algorithms,

Table 8: Wilcoxon-test results between FADE and the other 8 competitors on CEC2017 test suite with the two dimension cases.

	D=30			D=50		
FADE vs.	R^+	R^-	p	R^+	R^-	p
SaDE	20	7	**0.006**	26	4	**0.000**
CoDE	14	14	0.716	24	6	**0.000**
CoBiDE	18	10	0.084	18	12	**0.022**
TSDE	15	13	0.322	17	13	0.066
EFADE	14	14	0.657	18	12	0.063
EDEV	10	20	0.299	18	11	0.086
DI-ED	15	14	0.795	17	13	0.262
DPADE	18	10	**0.042**	20	10	**0.004**

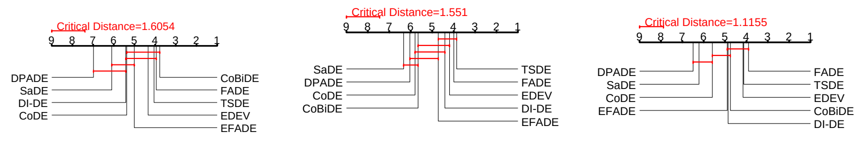
in terms of the mean value of the independent 51 runs. The results of Friedman-test on the two test suites are listed in Table 9, in which each algorithm and its rankings are listed in ascending order (the lower the better).

From Table 9 we can see that TSDE and CoBiDE attain the most outstanding performance, measured by ranking values, on the two dimension cases of CEC2013 test suite, respectively. Moreover, EDEV and TSDE show the most favorable performance on the two dimension cases of CEC2017 test suite, respectively. Although FADE cannot display the most outstanding characteristics on CEC2013 test suite or CEC2017 test suite, it yields the best overall performance, followed by TSDE and EDEV.

Furthermore, a graphical approach is used to show critical difference (CD) with Friedman rankings, the result of which is demonstrated in Figure 3. From Figure 3(a) we can see that FADE is significantly better than DPADE, SaDE, DI-DE and CoDE on CEC2013 test suite though there is no significant difference between FADE and other 4 DE variants on the test suite. In addition, Figure 3(b) shows that FADE is significantly better than SaDE, DPADE, CoDE, and CoBiDE on CEC2017 test suite. The overall performance on the two test suites demonstrated in Figure 3(c) manifests that FADE has the best result measured by the ranking value though it is not significantly better than TSDE, EDEV, CoBiDE, and DI-DE.

Table 9: Friedman-test results on CEC2013 and CEC2017 test suites.

	CEC2013(D=30)		CEC2013(D=50)		CEC2017(D=30)		CEC2017(D=50)		Overall	
	Algorithm	Ranking	Algorithm	Ranking	Algorithm	Ranking	Algorithm	Ranking	Algorithm	Ranking
1	TSDE	4.11	CoBiDE	3.32	EDEV	3.50	TSDE	3.52	FADE	4.01
2	CoBiDE	4.23	FADE	3.79	TSDE	4.30	FADE	3.72	TSDE	4.16
3	FADE	4.23	TSDE	3.99	FADE	4.53	DI-DE	4.18	EDEV	4.21
4	EDEV	4.32	EDEV	4.45	DI-DE	4.57	EDEV	4.48	CoBiDE	4.75
5	CoDE	4.86	EFADE	4.95	CoDE	4.63	CoBiDE	4.63	DI-DE	4.83
6	EFADE	5.07	DI-DE	5.73	EFADE	4.68	EFADE	4.73	EFADE	4.84
7	DI-DE	5.11	CoDE	5.89	DPADE	5.98	DPADE	6.10	CoDE	5.58
8	SaDE	5.93	SaDE	6.27	SaDE	6.10	SaDE	6.57	SaDE	6.19
9	DPADE	7.14	DPADE	6.82	CoBiDE	6.68	CoDE	7.07	DPADE	6.46



(a) CEC2013 test suite (b) CEC2017 test suite (c) Overall performance on the two test suites

Figure 3: Comparison results of the critical difference on CEC2013 and CEC2017 test suites.

4.4. *Performance of newly introduced strategies*

In this section, a series of experiments is conducted to testify the performance of new introduced strategies (i.e., ABS and APS) in FADE.

4.4.1. *Sensitivity of ABS*

In ABS, each individual in a swarm selects its own breeding strategy from the archive A_b according to the individual's fitness, and then chooses a few individuals as candidate parents to generate offspring. In this part, two experiments are conducted to analyze the sensitivity of the two adaptive selections in ABS, i.e., selections of breeding strategy and candidate parents.

In the first experiment, the fitness-based selection strategy of breeding s-trategy in FADE is replaced by a **r**andom selection for **b**reeding strategy. The comparison results between FADE and the FADE variants, named as FADE/rb, are listed in Table 10. Due to the space limitation, only CEC2017 test suite (D=30) is selected in this section.

Table 10: Comparison results between FADE and FADE/rb on CEC2017 test suite.

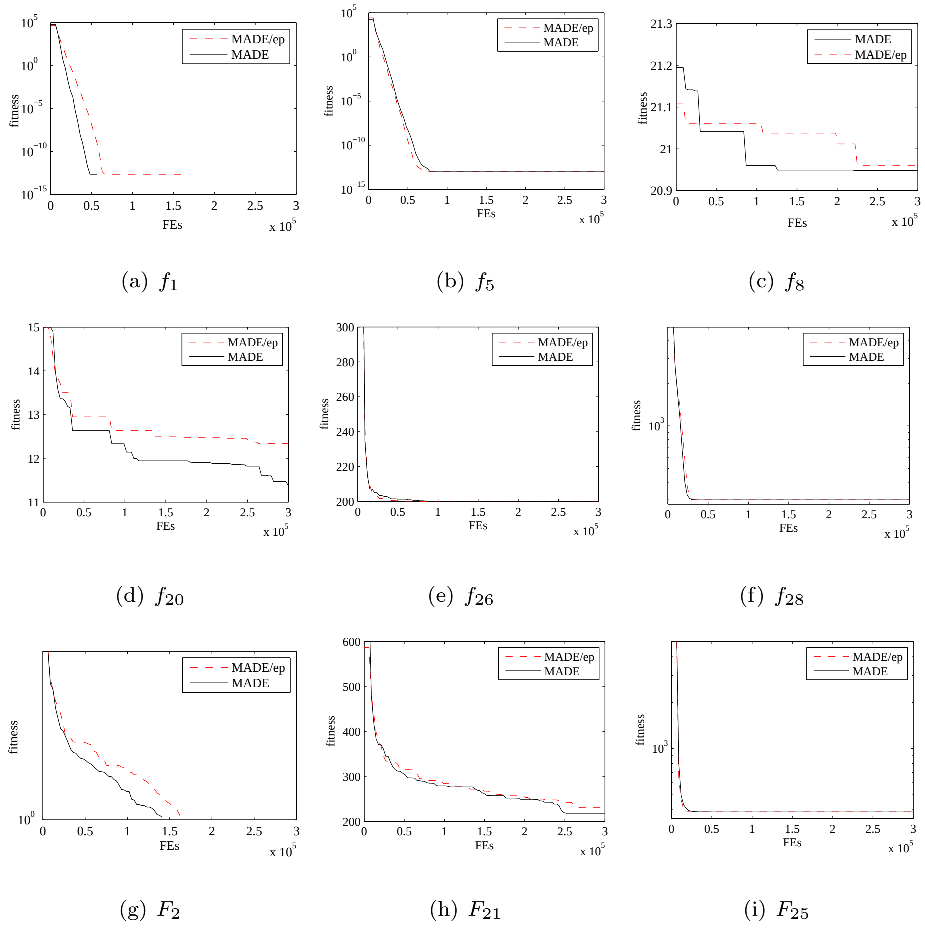
	FADE	FADE/rb
F_1	1.34E-14±1.59E-14	7.59E-12±1.33E-11(+)
F_2	0.00E+00±0.00E+00	0.00E+00±0.00E+00(=)
F_3	1.06E-10±2.01E-10	5.36E-09±6.65E-09(+)
F_4	3.21E+01±2.98E+01	4.23E+01±2.82E+01(=)
F_5	2.68E+01±5.37E+00	5.45E+01±2.01E+01(+)
F_6	1.24E-01±1.68E-01	2.45E-02±6.02E-02(-)
F_7	6.50E+01±8.62E+00	1.04E+02±1.57E+01(+)
F_8	2.91E+01±7.54E+00	5.18E+01±1.89E+01(+)
F_9	5.46E+00±9.74E+00	1.13E+00±2.00E+00(-)
F_{10}	3.17E+03±8.00E+02	4.38E+03±4.16E+02(+)
F_{11}	1.21E+01±1.05E+01	1.51E+01±1.83E+00(=)
F_{12}	1.62E+03±1.47E+03	1.61E+03±1.54E+03(=)
F_{13}	1.52E+03±5.52E+03	4.24E+01±2.85E+01(=)
F_{14}	2.42E+01±1.14E+01	3.47E+01±9.37E+00(+)
F_{15}	1.49E+01±2.14E+01	1.41E+00±5.67E+00(=)
F_{16}	5.13E+02±1.60E+02	4.60E+02±2.42E+02(=)
F_{17}	1.30E+02±1.10E+01	7.42E+01±5.99E+01(-)
F_{18}	3.61E+01±9.28E+00	3.70E+01±8.28E+00(=)
F_{19}	1.05E+01±1.09E+01	1.54E+01±3.79E+00(+)
F_{20}	1.69E+02±1.09E+02	1.11E+01±8.41E+01(-)
F_{21}	2.31E+02±6.61E+00	2.43E+02±1.66E+01(+)
F_{22}	1.01E+02±1.98E+00	1.00E+02±5.83E-01(-)
F_{23}	3.78E+02±8.87E+00	3.93E+02±1.76E+01(+)
F_{24}	4.56E+02±1.02E+01	4.52E+02±1.33E+01(=)
F_{25}	3.87E+02±1.23E+00	3.87E+02±8.03E-02(=)
F_{26}	1.24E+03±2.79E+02	1.23E+03±2.45E+02(=)
F_{27}	5.02E+02±7.90E+00	4.98E+02±8.84E+00(-)
F_{28}	3.24E+02±4.58E+01	3.30E+02±5.22E+01(=)
F_{29}	4.30E+02±3.24E+01	4.57E+02±5.16E+01(+)
F_{30}	2.00E+03±5.46E+01	2.01E+03±3.91E+01(=)
(#)+		11
(#)=		13
(#) -		6

The comparison results between FADE and FADE/rb show that FADE significantly dominates FADE/rb on 11 out of the 30 test functions, in terms of the *t*-test results. On the three unimodal functions, i.e., F_1 - F_3, FADE is significantly better than FADE/rb on F_1 and F_3, while both of them attain almost the same performance on F_2. Hence, we think selecting elites as candidate parents and choosing the breeding strategy that favorable for the exploitation are beneficial for unimodal functions. Furthermore, FADE also outperforms

FADE/rb on the majority of the multimodal functions. The experimental results manifest that the adaptation of breeding strategies in FADE can efficiently deal with the multimodal functions. However, we also observe that there is no significant difference between FADE and FADE/rb on the composition functions, contributing to the t-test results. Thus, it can be inferred that ABS plays a positive effect on unimodal functions and basic multimoal functions, though the performance of ABS on those complicated multimodal functions, i.e., the composition functions, needs to be further improved.

In the second experiment, comparison results between FADE and a FADE variant are used to analyze the performance of the adaptive selection strategy of candidate parents in ABS. In the FADE variant, named as FADE/ep in this work, the adaptive selection strategy of candidate parents in FADE is replaced by a common strategy applied in the canonical DE algorithm, in which all individuals in the entire population are regarded as candidate parents. In the experiment, we use figures to demonstrate the comparison results of the convergence process between FADE and FADE/ep, and then analyze the performance of the adaptive selection strategy of candidate parents in ABS. Due to the space limitation, only 9 functions, i.e., 3 unimodal functions (f_1, f_5 , and F_2), 2 basic multimodal functions (f_8 and f_{20}), and 4 composition functions (f_{26}, f_{28}, F_{21}, and F_{25}) in CEC2013 and CEC2017 test suites are selected in this experiment. The comparison results of convergence process between FADE and FADE/ep are illustrated in Figure 4.

The experimental results on the 3 unimodal functions (f_1, f_5, and F_2) show that FADE yields higher convergence speeds on f_1 and F_2 than FADE/ep though both of them obtain the global optimal solutions on the two functions. On f_5, FADE and FADE/ep offer almost the same performance, in terms of solution accuracy. Furthermore, the difference in convergence speed between the two DE variants is also negligible. On the two basic multimodal functions, i.e., f_8 and f_{20}, FADE dominates FADE/ep, contributing to solution accuracy. In addition, FADE offers a higher convergence speed than FADE/ep on the two multimodal functions. For the four composition functions, FADE shows slight better performance than FADE/ep on F_{21}, in terms of the convergence speed. On the contrary, the two algorithms display almost the same property on the two composition functions. The advantage of FADE indicates that the adaptive selection of candidate parents is propitious for speeding up the convergence on different functions to some extent.



(a) f_1 (b) f_5 (c) f_8

(d) f_{20} (e) f_{26} (f) f_{28}

(g) F_2 (h) F_{21} (i) F_{25}

Figure 4: Comparison results between FADE and FADE/ep.

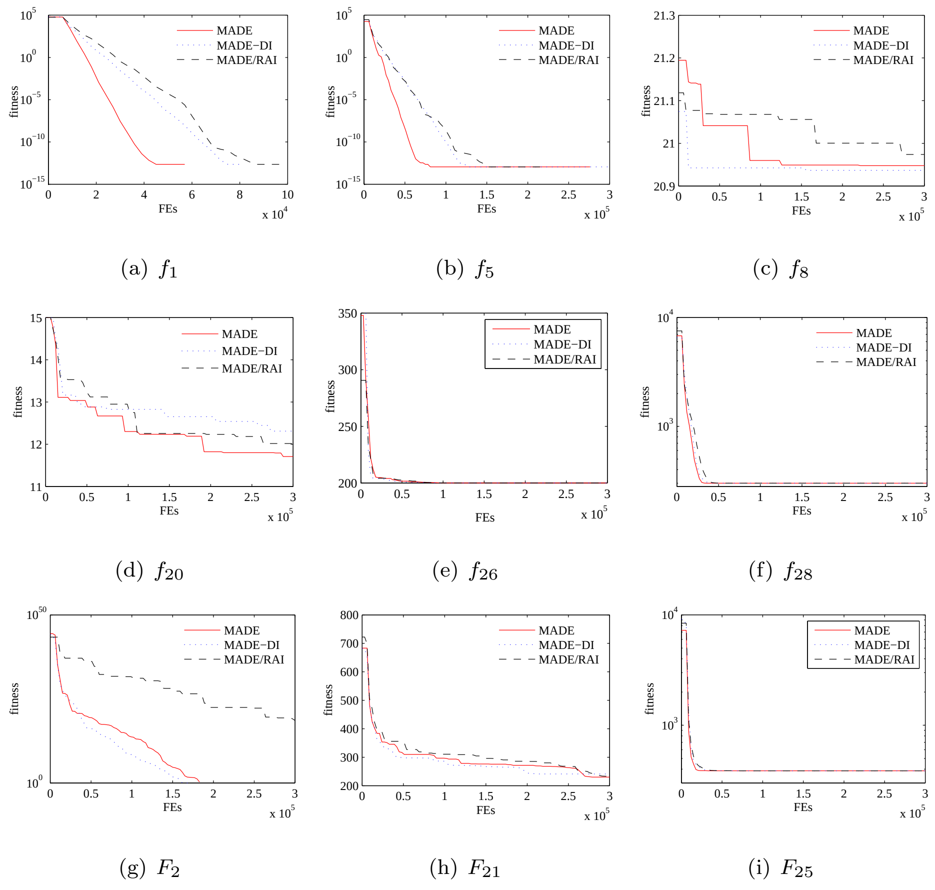
4.4.2. Sensitivity of APS

In APS, there are two processes, i.e., the deleting-individual and adding-individual processes. In this part, sensitivities of the two processes are discussed based on comparison results between FADE and other two FADE variants, respectively. In the first one FADE variant, named as FADE-DI, the **d**eleting **i**ndividuals process is removed. FADE/RAI is another FADE variant, in which the adding-individual process is replaced by a **r**andom **a**dding **i**ndividuals process. Note that the test functions selected in the previous experiment are also applied in this experiment. The comparison results of convergence process among FADE, FADE-DI and FADE/RAI are illustrated in Figure 5.

From Figure 5 we can observe that FADE dominates FADE-DI on two unimodal functions (i.e., f_1 and f_5), contributing to convergence speed. On the other unimodal function (i.e., F_4), FADE yields slightly worse performance than FADE-DI. Although FADE offers a slower convergence speed than FADE-DI on f_8, both of the two algorithms obtain almost the same solution accuracy on the multimodal function. Furthermore, FADE displays more comprehensive performance on f_{20} than FADE-DI, in terms of convergence speed and solution accuracy. On the 4 composition functions (i.e., f_{26}, f_{28}, F_{21}, and F_{25}), which are difficult to be optimized, the difference between FADE and FADE-DI is very little and can be neglected. The comparison results between FADE and FADE-DI manifest that the deleting-individual process plays positive performance on different functions, especially on unimodal functions.

The comparison results between FADE and FADE/RAI demonstrate that FADE significantly outperforms FADE/RAI on all the 3 unimodal functions as well as the 2 multimodal functions. On the 4 complicated functions, i.e., composition functions, FADE is also slightly better than FADE/RAI on f_{28}, F_{21}, and F_{25}, in terms of convergence speed. Thus, we regard that the adding-individual process is beneficial for different functions to different extents.

Moreover, an additional experiment in this part is used to investigate the changing trend of population size on different functions, the results of which are



(a) f_1 (b) f_5 (c) f_8

(d) f_{20} (e) f_{26} (f) f_{28}

(g) F_2 (h) F_{21} (i) F_{25}

Figure 5: Performance of deleting individuals and adding individuals processes.

illustrated in Figure 6. Note that the red line in each sub-figure denotes the change of fitness, while the blue line represents the fluctuation of population size during the entire evolutionary process.

The results on f_1, f_5, f_{28}, and F_{25} show that the population size sinks rapidly while the fitness on the two functions dramatically declines during the initial evolutionary process. Combining the experimental results demonstrated in Figure 5, we find out that the decreasing-individual process speeds up the convergence on the 4 functions, and then improves the accuracy of the solutions. On f_8, the population size keeps unchanged in the entire evolutionary process. Furthermore, there is only slight fluctuation of population size on f_{20} and F_2. Thus, we regard that the adding-individual process allows FADE to attain competitive results on these functions. The conclusion is also verified by the results demonstrated in Figure 5. On f_{26}, the population size changes in the later evolutionary stage, while it dramatically decreases in the later stage. Combining the experimental results demonstrated in Figure 5, we observe that the performance of APS on the two functions needs to be further improved.

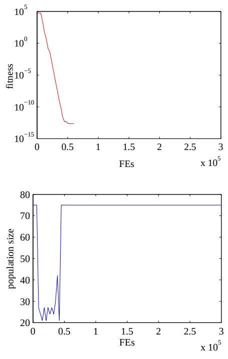
Although two threshold values, Max_{imp} and Max_{stag}, are applied in all the 9 test functions, the experimental results demonstrated in Figure 5 show that the deleting-individual and adding-individual processes are executed at different time in the different functions. In other words, when to perform the processes ultimately depends on the fitness landscape in a test function. Hence, we regard the APS procedure is adaptively conducted during the evolutionary process.

5. Performance on real applications

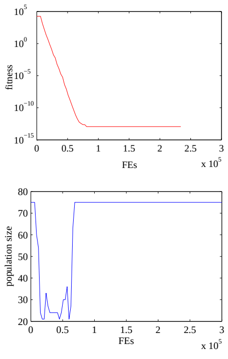
To testify the performance of FADE on real applications, 7 engineering problems are adopted in this section, including:

- P_1: Parameter estimation for frequency modulated sound waves;
- P_2: Lennard-Jones potential problem;
- P_3: Bifunctional catalyst blend optimal control problem;
- P_4: Optimal control of a non-linear stirred tank reactor;
- P_5: Tersoff potential function minimization problem;
- P_6: Spread spectrum radar polly phase code design; and
- P_7: Transmission network expansion planning problem.

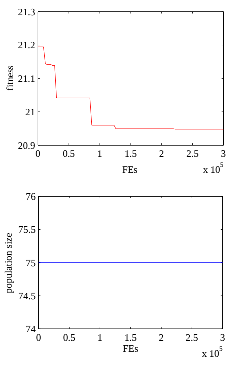
Due to the space limitation, details of the 7 real applications can be observed from the literature [8].



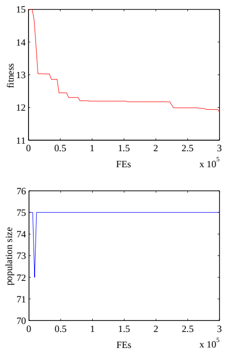
(a) f_1



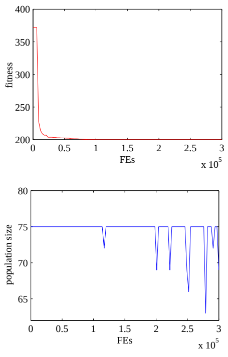
(b) f_5



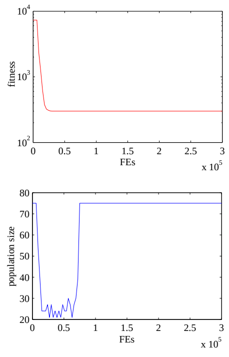
(c) f_8



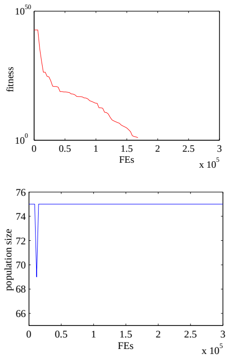
(d) f_{20}



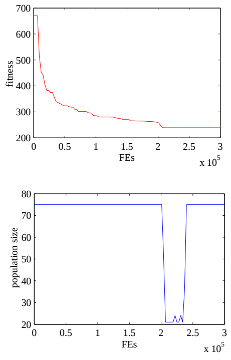
(e) f_{26}



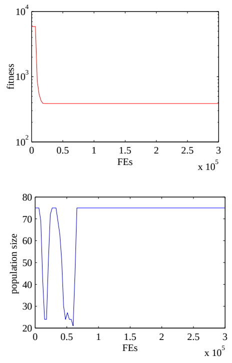
(f) f_{28}



(g) F_2



(h) F_{21}



(i) F_{25}

Figure 6: Fluctuation of population size on different functions.

Parameters of each involved algorithm and experimental setup are the same as that in the above experiments except *MaxFEs* is set to 150 000 for each run. The experimental results measured by the mean value of 30 independent runs are listed in Table 11. Moreover, results of the t-test between FADE and other competitors also presented in the table.

Table 11: Comparison results on 7 real applications.

		FADE	SaDE	CoDE	CoBiDE	TSDE	EFADE	EDEV	DI-DE	DPADE
P_1	*Mean*	6.08E+00	4.10E-01(-)	7.29E-01(-)	2.13E+00(-)	2.73E+00(-)	**2.74E-01**(-)	6.79E+00(=)	2.84E+00(-)	5.97E+00(=)
P_2	*Mean*	-2.47E+01	-2.12E+01(+)	-2.67E+01(=)	-2.43E+01(=)	-1.99E+01(+)	-2.43E+01(=)	-2.43E+01(=)	**-3.18E+00**(-)	-2.50E+01(=)
P_3	*Mean*	1.15E-05	1.15E-05(=)	1.15E-05(=)	1.15E-05(=)	1.15E-05(=)	1.15E-05(=)	**2.35E-06**(-)	1.15E-05(=)	1.15E-05(=)
P_4	*Mean*	**1.53E+01**	1.76E+01(+)	1.90E+01(+)	1.99E+01(+)	1.74E+01(+)	2.05E+01(+)	1.71E+01(+)	1.57E+01(=)	1.78E+01(+)
P_5	*Mean*	**-3.28E+01**	-3.05E+01(=)	5.05E+01(+)	-3.11E+01(=)	-3.09E+01(=)	-2.76E+01(+)	-3.15E+01(=)	1.36E+00(+)	1.43E+00(+)
P_6	*Mean*	-2.35E+01	-1.63E+01(+)	1.03E+01(+)	-2.08E+01(+)	-1.74E+01(+)	-1.49E+01(+)	**-2.40E+01**(=)	1.36E+00(+)	1.26E+00(+)
P_7	*Mean*	1.15E+00	1.12E+00(=)	6.36E-01(-)	6.42E-01(-)	**6.08E-01**(-)	1.00E+00(=)	8.84E-01(=)	1.88E+00(+)	2.20E+02(+)
(#)	+	-	3	3	2	3	3	1	3	4
(#)	=	-	3	2	3	2	3	5	2	3
(#)	-	-	1	2	2	2	1	1	2	0

From the comparison results listed in Table 11 we can observe that FADE and EDEV attain the best performance since they both yield the best mean results on 2 out of the 7 real applications, followed by TSDE, EFADE, and DI-DE. Although CoBiDE displays very promising performance on CEC2013 test suite, it cannot yield very reliable performance on the 7 applications since it cannot offer the best mean value on any real application. On the contrary, FADE attains more comprehensive and favorable characteristics since it exhibits outstanding performance both on the 2 test suites and the 7 real applications. Furthermore, the t-test results indicate that FADE surpasses SaDE, CoDE, TSDE, EFADE, DI-DE, and DPADE because FADE dominates the 6 competitors on the majority of the applications. Furthermore, FADE displays the same performance as CoBiDE measured by the t-test results though CoBiDE does not yield the best mean value on any real application. The comparison results between FADE and EDEV show that there is no significant difference between the two DE variants, except FADE and EDEV attain more significantly better performance on P_4 and P_3, respectively.

6. Conclusions

In this paper, we propose a fitness-based adaptive differential evolution algorithm (FADE), in which the entire population is split into many small-sized swarms, and then three adaptive strategies are conducted during the entire evolutionary process. Firstly, individuals in a swarm adaptively select distinct breeding strategies saved in an archive in each generation based on the relative fitness of the individuals. Furthermore, every individual in a swarm also uses the fitness-based method to adaptively choose some individuals from the current population as its candidate parents. After that, each individual breeds offspring based on the adaptively selected breeding strategy (i.e., trial vector generating strategy and control parameters) and the candidate parents. Since an individual may play different roles in different generations due to its different relative fitness in a swarm, the individual can display distinct search behaviors in different generations based on the different breeding strategies. Lastly, an adaptive technique is chosen to adjust the population size of FADE. Relying on the adaptability of population size, the computational resources can be rationally distributed in different evolutionary stages.

To evaluate the performance of FADE, extensive experiments have been carried out on CEC2013 and CEC2017 test suites with two dimensionality cases, as well as 7 real applications. The comparison results between FADE and other 8 state-of-art DE variants manifest that FADE offers more favorable overall performance both at the test suites and real applications. Moreover, the sensitivity of the novel adaptive strategies involved in FADE also has been discussed by a series of experiments. The experimental results illustrate a few merits of FADE. The one is that adaptive selecting of breeding strategies and candidate parents for different individuals is conducive to speeding up convergence and enhance population's exploration ability. Moreover, the adaption of population size is propitious for speed up convergence, especially for unimodal and some multimodal functions.

Although FADE offers a competitive performance testified by the extensive experiments, there are a few problems need to be further studied in our future work. The one is the efficiency and effectiveness of different trial vector generation strategies and control parameters applied in FADE need to be in-depth analyzed, though some preliminary discussions have been proposed in this paper. The other one is when and how to adjust the population size to satisfy various fitness landscapes.

Acknowledgment

This study was funded by the National Natural Science Foundation of China (Grant nos.: 61663009, 61602174, 61762036, 61806204, 61876136, 61702239) and the Science and Technology Plan Projects of Zhangzhou (Grant nos: ZZ2020J06, ZZ2020J24).

Reference

[1] M.Z. Ali, N.H. Awad, P.N. Suganthan *et al.*, An adaptive multipopulation differential evolution with dynamic population reduction, IEEE Trans. Cybern. 47(9)(2017) 2768-2779.

[2] N.H. Awad, M.Z. Ali, J.J. Liang *et al.*, Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective real-parameter numerical optimization, Nanyang Technological Univ., Singapore, Tech. Rep., 2016.

[3] J. Brest, S. Greiner, B. Bosković *et al.*, Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems, IEEE Trans. Evol. Comput. 10(6)(2006) 646-657.

[4] Y.Q. Cai, G. Sun, T. Wang *et al.*, Neighborhood-adaptive differential evolution for global numerical optimization, Appl. Soft Comput. 59(2017) 659-706.

[5] Y.Q Cai, J.L. Liao, T. Wang *et al.*, Social learning differential evolution, Inf. Sci. 433-434(2018) 464-509.

[6] L.Z. Cui, G.H. Li, Z.X. Zhu *et al.*, Adaptive multiple-elites-guided composite differential evolution algorithm with a shift mechanism, Inf. Sci. 422(2018) 122-143.

[7] L.Z. Cui, G.H. Li, Z.X. Zhu *et al.*, Differential evolution algorithm with dichotomy-based parameter space compression, Soft Comput. (23)(2019): 3643-3660.

[8] S. Das, P.N. Suganthan, Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems, Jadavpur University, Kolkata, Technical Report, 2010.

[9] S. Das, S.S. Mullick, P.N. Suganthan, Recent advances in differential evolution-an updated survey, Swarm Evol. Comput. 27(2016) 1-30.

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

[11] W. Du, S.Y. Leung, Y. Tang *et al.*, Differential evolution with event-triggered impulsive control, IEEE Trans. Cybern. 47(1)(2017) 244-257.

[12] R. Gämperle, S. D. Muller, P. Koumoutsakos, A parameter study for differential evolution, in: Proc. of Advances Intelligent Systerms Fuzzy Systerms, 2002, pp. 293-298.

[13] A. Ghosh, S. Das, R. Mallipeddi *et al.*, A modified differential evolution with distance-based selection for continuous optimization in presence of noise, IEEE Access, (2017) 26944-26964.

[14] P. Ghosh, S. Das, H. Zafar, Adaptive-differential-evolution-based design of two-channel quadrature mirror filter banks for sub-band coding and data transmission, IEEE Trans. Syst., Man, Cybern. C, Appl. Rev. 42(6)(2012) 1613-1623.

[15] L. Gui, X.W. Xia, F. Yu *et al.*, A multi-role based differential evolution, Swarm Evol. Comput. (50)(2019)100508.

[16] S.M. Guo, C.C. Yang, Enhancing differential evolution utilizing eigenvector-based crossover operator, IEEE Trans. Evol. Comput., 19(1)(2015) 31-49.

[17] V.L. Huang, A.K. Qin, P.N. Suganthan, Self-adaptive differential evolution algorithm for constrained real-parameter optimization, in: Proc. of IEEE Congress on Evolutionary Computation, CEC'06, Vancouver, BC, Canada, 2006, pp. 17-24.

[18] J.J. Liang, B.Y. Qu, P.N. Suganthan, Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization, Nanyang Technological Univ., Singapore, Tech. Rep., 2013.

[19] Z.Z. Liu, Y. Wang, S.X. Yang *et al.*, Differential evolution with a two-stage optimization mechanism for numerical optimization, in: Proc. of IEEE Congress on Evolutionary Computation, CEC'16, Vancouver, BC, Canada, 2016, pp. 3170-3177.

[20] R. Mendes, I. Rocha, E. C. Ferreira *et al.*, A comparison of algorithms for the optimization of fermentation processes, in: Proc. of IEEE Congress on Evolutionary Computation, CEC'06, Vancouver, BC, Canada, 2006, pp. 2018-2025.

[21] E. Mezura-Montes, J. Velázquez-Reyes, C.A.C. Coello, A comparative s-tudy of differential evolution variants for global optimization, in: Proc. of the 2006 annual conference on Genetic and Evolutionary Computation, GECCO'06, Seattle, Washington, USA, 2006, pp. 485-492.

[22] A.W. Mohamed, P.N. Suganthan, Real-parameter unconstrained optimization based on enhanced fitness-adaptive differential evolution algorithm with novel mutation, Soft Comput. 22(10)(2018) 3215-3235.

[23] A.W. Mohamed, A.A. Hadi, K.M. Jambi, Novel mutation strategy for enhancing SHADE and LSHADE algorithms for global numerical optimization, Swarm Evol. Comput. (50)(2019)100455.

[24] Q.-K. Pan, L. Wang, L. Gao *et al.*, An effective hybrid discrete differential evolution algorithm for the flow shop scheduling with intermediate buffers, Inf. Sci. 181(3)(2011) 668-685.

[25] K. Price, R. Storn, Differential evolution: a practical approach to global optimization, Springer-Verlag New York, 2005.

[26] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput. 13(2)(2009) 398-417.

[27] X. Qiu, K.C. Tan, J.X. Xu, Multiple exponential recombination for differential evolution, IEEE Trans. Cybern. 47(4)(2017) 995-1006.

[28] R.A. Sarker, S.M. Elsayed, T. Ray, Differential evolution with dynamic parameters selection for optimization problems, IEEE Trans. Evol. Comput. 18(5)(2014) 689-707.

[29] R. Storn, K. Price, Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11(4)(1997) 341-359.

[30] G. Sun, Y.q. Cai, T. Wang *et al.*, Differential evolution with individual-dependent topology adaptation, Inf. Sci. 450(2018) 1-38.

[31] L.X. Tang, Y. Dong, J.Y. Liu, Differential evolution with an individual-dependent mechanism, IEEE Trans. Evol. Comput. 19(4)(2015) 560-574.

[32] L. Tian, Z.C. Li, X.F. Yan, Differential evolution algorithm directed by individual difference information between generations and current individual information, Appl. Intel. (49)(2019): 628-649.

[33] M.N Tian, X.B Gao, Differential evolution with neighborhood-based adaptive evolution mechanism for numerical optimization, Inf. Sci. 478(2019) 422-448.

[34] M.N. Tian, X.B. Gao, An improved differential evolution with information intercrossing and sharing mechanism for numerical optimization, Swarm Evol. Comput. (50)(2019) 100341.

[35] Y. Wang, Z.X. Cai, Q.F. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, IEEE Trans. Evol. Comput. 15(1)(2011) 55-67.

[36] Y. Wang, Z.X. Cai, Q.F. Zhang, Enhancing the search ability of differential evolution through orthogonal crossover, Inf. Sci. 185(1)(2012) 153-177.

[37] Y. Wang, H.X. Li, T. Huang *et al.*, Differential evolution based on covariance matrix learning and bimodal distribution parameter setting, Appl. Soft Comput. 18(1)(2014) 232-247.

[38] H.B. Wang, X.N. Ren, G.Q. Li, *et al.*, APDDE: self-adaptive parameter dynamics differential evolution algorithm, Soft Comput. 22(2018) 1313-1333.

[39] B. Wei, X.W. Xia, F. Yu *et al.*, Multiple adaptive strategies based particle swarm optimization algorithm, Swarm Evol. Comput. 57(2020)100731.

[40] G.H. Wu, R. Mallipeddi, P.N. Suganthan *et al.*, Differential evolution with multi population based ensemble of mutation strategies, Inf. Sci. 329(C)(2016) 329-345.

[41] G.H. Wu, X. Shen, H.F. Li *et al.*, Ensemble of differential evolution variants, Inf. Sci. 423(2018) 172-186.

[42] X.W. Xia, L. Gui, Z.H. Zhan, A multi-swarm particle swarm optimization algorithm based on dynamical topology and purposeful detecting, Appl. Soft Comput. 67(2018) 126-140.

[43] W.J. Yu, M. Shen, W.N. Chen *et al.*, Differential evolution with two-level parameter adaptation, IEEE Trans. Cybern. 44(7)(2014) 1080-1099.

[44] D. Zaharie, Control of population diversity and adaption in differential evolution algorithms, in: Proc. of Mendel 2003, Ninth International Conference on Soft Computing, 2003, pp. 41-46.

[45] J. Zhang, A. Sanderson, JADE: adaptive differential evolution with optional external archive, IEEE Trans. Evol. Comput. 13(5)(2009) 945-958.

[46] L.M. Zheng, S.X. Zhang, S.Y. Zheng *et al.*, Differential evolution algorithm with two-step subpopulation strategy and its application in microwave circuit designs, IEEE Trans. Ind. Inf. 12(3)(2016) 911-923.

[47] L.M. Zheng, S.X. Zhang, K.S. Tang *et al.*, Differential evolution powered by collective information, Inf. Sci. 399(2017) 13-29.

[48] X.G. Zhou, G.J. Zhang, Abstract convex underestimation assisted multistage differential evolution, IEEE Trans. Cybern. 47(9)(2017) 2730-2741.

[49] W. Zhu, Y. Tang, J. Fang *et al.*, Adaptive population tuning scheme for differential evolution, Inf. Sci. 223(2013) 164-191.