Enhanced Differential Evolution with Adaptive Strategies for Numerical Optimization

Wenyin Gong, Zhihua Cai, Charles X. Ling, *Senior Member, IEEE*, and Hui Li

***Abstract*— Differential evolution (DE) is a simple yet efficient evolutionary algorithm for global numerical optimization, which has been widely used in many areas. However, the choice of the best mutation strategy is difficult for a specific problem. To alleviate this drawback and enhance the performance of DE, in this paper, we present a family of improved DE that attempts to adaptively choose a more suitable strategy for a problem at hand. In addition, in our proposed strategy adaptation mechanism (SaM) different parameter adaptation methods of DE can be used for different strategies. In order to test the efficiency of our approach, we combine our proposed SaM with JADE, which is a recently proposed DE variant, for numerical optimization. Twenty widely used scalable benchmark problems are chosen from the literature as the test suit. Experimental results verify our expectation that SaM is able to adaptively determine a more suitable strategy for a specific problem. Compared with other state-of-the-art DE variants, our approach performs better, or at least comparably, in terms of the quality of the final solutions and the convergence rate. Finally, we validate the powerful capability of our approach by solving two real-world optimization problems.**

***Index Terms*— Differential evolution; strategy adaptation; parameter adaptation; numerical optimization; real-world problems.**

I. Introduction

OVER THE LAST few decades, evolutionary algorithms (EAs) have received much attention regarding their potential as global optimization techniques [1]. Inspired by the mechanisms of natural evolution and survival of the fittest, EAs utilize a collective learning process of a population of individuals. Offspring are generated using randomized operations such as mutation and recombination. According to a fitness measure, the selection process favors better individuals to reproduce more often than those that are relatively worse.

Differential evolution (DE), proposed by Storn and Price [2], is a simple yet powerful EA with the generate-and-test feature for global optimization. In DE, the mutation operator is based on the distribution of solutions in the current population. New candidates are created by combining the parent solution and the mutant. A candidate replaces the parent only if it has better fitness value. In [2], Price and Storn gave the working principle of DE with single mutation strategy. Later on, they suggested ten different mutation strategies in [3], [4]. Among DE's advantages are its simple structure, ease of use, speed, and robustness. These advantages give it many real-world applications, such as data mining [5], [6], pattern recognition, digital filter design, neural network training, etc. [3], [7], [8]. Most recently, DE has also been used for the global permutation-based combinatorial problems [9].

Although DE has been widely used in many fields, it has been shown to have certain weaknesses. One of these weaknesses is that choosing the best among different mutation strategies available for DE is not easy for a specific problem [10], [11]. To the best of our knowledge, there is a little work to improve DE with the adaptive strategy method. Xie and Zhang [12] presented a swarm algorithm framework (SWAF), where a neural network is used to adaptively update the weights on some strategies of DE based on their previous success rates. In [13], Zamuda *et al.* set fixed selection probability for each strategy. Then, a uniform randomly generated parameter r_s is used to determine which mutation strategy will be selected. Qin *et al.* [10], [11] proposed a variant of DE, namely SaDE. In SaDE, it implements different strategies and updates their weights in the search based on their previous success rate[1].

In order to select a more suitable strategy adaptively for a specific problem and further enhance the performance of DE, in this paper, we describe a family of DE variants, in which a simple strategy adaptation mechanism (SaM) is implemented for each variant. Additionally, in our proposed SaM different parameter adaptation methods proposed in the DE literature can be used for different strategies. In order to evaluate the performance of our approach, SaM is combined with JADE [15], [16], which is a recent DE variant and obtains good results in numerical optimization. Experimental results indicate that our proposed SaM is able to adaptively determine a more suitable strategy at different stages of evolution process for a specific problem.

The rest of this paper is organized as follows. Section II briefly describes the DE algorithms and some of its variants. Our proposed approach is presented in detail in Section III. In Section IV, we verify our approach through 20 benchmark functions and two real-world problems. Moreover, the experimental results are compared with several other DE approaches in this section. Section V is devoted to conclusions.

This work was partly supported by the Fund for Outstanding Doctoral Dissertation of CUG, the National High Technology Research and Development Program of China under Grant No. 2009AA12Z117, and the Research Fund for the Doctoral Program of Higher Education under Grant No. 20090145110007.

W. Gong, Z. Cai, and H. Li are with the School of Computer Science, China University of Geosciences, Wuhan P.R. China (e-mail: cug11100304@yahoo.com.cn; zhcai@cug.edu.cn; huili@vip.sina.com). The corresponding author is Z. Cai.

C.X. Ling is with the Dept. of Computer Science, The University of Western Ontario, London, Canada (e-mail: cling@csd.uwo.ca).

[1]In jDE-2 [14], the strategy adaptation method is also used, however, it only adopts the method proposed in [10].

II. Differential Evolution and Its Variants

In this work, we consider the following numerical optimization problem:

$$\text{Minimize} \quad f(\mathbf{x}), \qquad \mathbf{x} \in S, \tag{1}$$

where $S \subseteq \mathbb{R}^D$ is a compact set, $\mathbf{x} = [x_1, x_2, \cdots, x_D]^T$, and D is the dimension of the decision variables. Generally, for each variable x_i it satisfies a boundary constraint

$$L_i \leq x_i \leq U_i, i = 1, 2, \cdots, D. \tag{2}$$

Recently, using EAs to solve the global optimization has been very active, producing different kinds of EA for optimization in the continuous domain, such as genetic algorithms [17], evolution strategy [18], evolutionary programming [19], particle swarm optimization (PSO) [20], immune clonal algorithm [21], differential evolution [2], etc.

A. Differential Evolution

The DE algorithm [2] is a simple EA for global numerical optimization. It creates new candidate solutions by combining the parent individual and several other individuals of the same population. A candidate replaces the parent only if it has better fitness value. The pseudo-code of the original DE algorithm is shown in Algorithm 1. Where D is the number of decision variables; NP is the population size; F is the mutation scaling factor; CR is the crossover rate; $x_{i,j}$ is the j-th variable of the solution $\mathbf{x}_i$; $\mathbf{u}_i$ is the offspring. $\text{rndint}(1, D)$ is a uniformly distributed random integer number between 1 and D. $\text{rndreal}_j[0, 1)$ is a uniformly distributed random real number in $[0, 1)$, generated anew for each value of j. Many mutation strategies to create a candidate are available. In Algorithm 1, the classic "DE/rand/1/bin" strategy is illustrated (see lines 6 - 13 of Algorithm 1). More details on "DE/rand/1/bin" and other DE strategies can be found in [3] and [4]. In [3], the vector $\mathbf{x}_i$ is referred to as *target vector*. The vectors $\mathbf{x}_{r_1}$, $\mathbf{v}_i$, and $\mathbf{u}_i$ are named as *base vector*, *mutant vector*, and *trial vector*, respectively.

From Algorithm 1, we can see that there are only three control parameters in DE. These are NP, F and CR. As for the terminal conditions, we can either fix the maximum number of fitness function evaluations (NFFEs) *Max_NFFEs* or the precision of a desired solution *VTR* (value to reach).

In DE, many schemes have been proposed [3], [4] that use different mutation strategies and/or recombination operations in the reproduction stage. In order to distinguish among its schemes, the notation "DE/a/b/c" is used, where "DE" denotes the DE algorithm; "a" specifies the vector to be mutated; "b" is the number of difference vectors used; and "c" denotes the crossover scheme, *binomial* or *exponential*. In line 9 of Algorithm 1, the mutation strategy is called "DE/rand/1", which is a classic strategy of DE [3]. Other well-known mutation strategies are listed as follows.

1) "DE/best/1":

$$\mathbf{v}_i = \mathbf{x}_{best} + F \cdot \left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right) \tag{3}$$

2) "DE/best/2":

$$\mathbf{v}_i = \mathbf{x}_{best} + F \cdot \left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right) + F \cdot \left(\mathbf{x}_{r_4} - \mathbf{x}_{r_5}\right) \tag{4}$$

Algorithm 1 The DE algorithm with DE/rand/1/bin strategy

```
1: Generate the initial population
2: Evaluate the fitness for each individual
3: while The halting criterion is not satisfied do
4:    for i = 1 to NP do
5:       Select uniform randomly r1 ≠ r2 ≠ r3 ≠ i
6:       j_rand = rndint(1, D)
7:       for j = 1 to D do
8:          if rndreal_j[0, 1) < CR or j == j_rand then
9:             u_{i,j} = v_{i,j} = x_{r1,j} + F · (x_{r2,j} − x_{r3,j})
10:         else
11:            u_{i,j} = x_{i,j}
12:         end if
13:      end for
14:   end for
15:   for i = 1 to NP do
16:      Evaluate the offspring u_i
17:      if f(u_i) is better than or equal to f(x_i) then
18:         Replace x_i with u_i
19:      end if
20:   end for
21: end while
```

3) "DE/rand/2":

$$\mathbf{v}_i = \mathbf{x}_{r_1} + F \cdot \left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right) + F \cdot \left(\mathbf{x}_{r_4} - \mathbf{x}_{r_5}\right) \tag{5}$$

4) "DE/current-to-best/1"[2]:

$$\mathbf{v}_i = \mathbf{x}_i + F \cdot \left(\mathbf{x}_{best} - \mathbf{x}_i\right) + F \cdot \left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right) \tag{6}$$

where $\mathbf{x}_{best}$ represents the best individual in the current generation, r_1, r_2, r_3, r_4, and $r_5 \in \{1, \cdots, NP\}$, and $r_1 \neq r_2 \neq r_3 \neq r_4 \neq r_5 \neq i$.

Generally, different strategy has different characteristics and is suitable for a set of problems. For example:

- The "DE/rand/1/bin" strategy is a classic DE strategy, which is usually less greedy, slower convergence speed, and more reliable than the strategies based on the best-so-far solution. Hence, this strategy is more suitable for multi-modal problems. It has been widely used in the DE literature [2], [23], [6], [24], [25], etc.
- The best-so-far solution based strategies such as "DE/best/1", "DE/current-to-best/1", always converge faster and are more suitable for unimodal functions [26], [11]. In addition, the "DE/best/1" strategy was succeeded in solving some design problems, e.g., the digital design problems [27] and the optimal design of shell-and-tube heat exchangers [28].
- As stated in [11], the strategies based on the two difference vectors, e.g., "DE/rand/2", are able to provide better perturbation than the one difference vector based strategy. However, the performance of the two difference vectors based strategies needs to be further investigated.

B. Some Variants of DE

In the DE literature, there are many improved variants. In this section, we only briefly describe three of them, i.e., jDE [23], SaDE [11], and JADE [15], [16], since these methods show good performance and we will compare them with our approach in this work.

[2]"DE/current-to-best" is also referred to as "DE/target-to-best/" [3], [22].

1) The jDE method: In jDE [23], the parameters CR_i and F_i are encoded in each individual X_i (i.e., $X_i = \langle \mathbf{x}_i, CR_i, F_i \rangle$). They are updated as follows:

$$F_i = \begin{cases} \mathrm{rndreal}_i[0.1, 1], & \mathrm{rndreal}[0,1] < \tau_1 \\ F_i, & \text{otherwise} \end{cases} \quad (7)$$

$$CR_i = \begin{cases} \mathrm{rndreal}_i[0, 1], & \mathrm{rndreal}[0,1] < \tau_2 \\ CR_i, & \text{otherwise} \end{cases} \quad (8)$$

where $\mathrm{rndreal}[a, b]$ is a uniformly distributed random number between a and b. $\tau_1 = 0.1$ and $\tau_2 = 0.1$ indicate probabilities to adjust factors F_i and CR_i. The newly generated F_i and CR_i are obtained before the mutation and crossover operations. Therefore, they influence the following recombination and selection. The method has been proved efficient based on some benchmark experimental results [23].

2) The SaDE method: Qin *et al.* [11] proposed the SaDE method, in which four strategies ("DE/rand/1/bin", "DE/rand-to-best/2/bin", "DE/rand/2/bin", and "DE/current-to-rand/1") are adaptively selected based on their previous experiences of generating promising solutions. In addition, the crossover rates CR_i are also adaptively changed according to its previous experiences.

Denote $p_k, k = 1, 2, \cdots, K$ as the probability of applying the k-th strategy, where K is the total number of strategies in the strategy pool. p_k is initialized as $1/K$. The stochastic universal selection method is used to select the strategy for each target vector based on the probability p_k. p_k is updated after LP generations in the following manner:

$$p_k = \frac{S_{k,G}}{\sum_{k=1}^{K} S_{k,G}} \quad (9)$$

where

$$S_{k,G} = \frac{\sum_{g=G-LP}^{G-1} ns_{k,g}}{\sum_{g=G-LP}^{G-1} ns_{k,g} + \sum_{g=G-LP}^{G-1} nf_{k,g}} + \epsilon \quad (10)$$

where $G (G > LP)$ is the generation counter; $ns_{k,g}$ and $nf_{k,g}$ are the respective numbers of the offspring vectors generated by the k-th strategy that survive or fail in the selection operation in the last LP generations; $S_{k,G}$ is the success rate of the trial vector generated by the k-th strategy and successfully entering the next generation; ϵ is a small constant value to avoid the possible null success rates.

In SaDE, the mutation factors F_i are independently generated at each generation as follows:

$$F_i = \mathrm{rndn}_i(0.5, 0.3) \quad (11)$$

where $\mathrm{rndn}_i(0.5, 0.3)$ means a random number generated anew for the i-th target vector based on a normal distribution with mean 0.5 and standard deviation 0.3.

The crossover rates of the k-th strategy $CR_{i,k}$ are also independently generated at each generation according to

$$CR_{i,k} = \mathrm{rndn}_i(\mathrm{CRm}_k, 0.1) \quad (12)$$

and truncated to $[0, 1]$. Where CRm_k is initialized as 0.5. Denote $\mathrm{CRMemory}_k$ as the memory to store the CR values with respect to the k-th strategy that generated trial vectors successfully entering the next generation with the previous LP generations. After LP generations, the median value saved in $\mathrm{CRMemory}_k$ is calculated to overwrite CRm_k at each generation. Through experiments, they concluded that SaDE is effective in obtaining high quality solutions.

3) The JADE method: Recently, Zhang and Sanderson [15], [16] proposed a DE variant, namely JADE, which obtains very competitive results when solving some unconstrained benchmark problems and real-world problems. In JADE, the authors originally implemented two mutation strategies in [15], i.e., "DE/current-to-$p$best" without archive and "DE/current-to-$p$best" with archive:

1) "DE/current-to-$p$best/1 (without archive)":

$$\mathbf{v}_i = \mathbf{x}_i + F_i \cdot \left(\mathbf{x}_{best}^p - \mathbf{x}_i\right) + F_i \cdot \left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right) \quad (13)$$

2) "DE/current-to-$p$best/1 (with archive)":

$$\mathbf{v}_i = \mathbf{x}_i + F_i \cdot \left(\mathbf{x}_{best}^p - \mathbf{x}_i\right) + F_i \cdot \left(\mathbf{x}_{r_2} - \tilde{\mathbf{x}}_{r_3}\right) \quad (14)$$

In the latter one, an archive, $\mathbf{A}$, is used to store the inferior solutions recently explored in the evolutionary search. Where the $\mathbf{x}_{best}^p$ is a $p$best solution, which is randomly selected as one of the top $100p\%$ solutions with $p \in (0, 1]$. $\mathbf{x}_i$, $\mathbf{x}_{r_2}$, and $\mathbf{x}_{best}^p$ are chosen from the current population $\mathbf{P}$; $\tilde{\mathbf{x}}_{r_3}$ is randomly chosen from the union, $\mathbf{P} \cup \mathbf{A}$, of the archive and current population. Later on, in order to solve the large scale problems and further increase the population diversity, the same authors proposed other two strategies, "DE/rand-to-$p$best" without archive and "DE/rand-to-$p$best" with archive, in the following manner [16]:

3) "DE/rand-to-$p$best/1 (without archive)":

$$\mathbf{v}_i = \mathbf{x}_{r_1} + F_i \cdot \left(\mathbf{x}_{best}^p - \mathbf{x}_{r_1}\right) + F_i \cdot \left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right) \quad (15)$$

4) "DE/rand-to-$p$best/1 (with archive)":

$$\mathbf{v}_i = \mathbf{x}_{r_1} + F_i \cdot \left(\mathbf{x}_{best}^p - \mathbf{x}_{r_1}\right) + F_i \cdot \left(\mathbf{x}_{r_2} - \tilde{\mathbf{x}}_{r_3}\right) \quad (16)$$

At each generation, for each target vector the crossover rate CR_i is independently generated as follows:

$$CR_i = \mathrm{rndn}_i(\mu_{CR}, 0.1) \quad (17)$$

and truncated to the interval $[0, 1]$. Where μ_{CR} is the mean value to generate CR_i. It is updated as follows:

$$\mu_{CR} = (1 - c) \cdot \mu_{CR} + c \cdot \mathrm{mean}_A(S_{CR}) \quad (18)$$

where c is a constant in $[0, 1]$; $\mathrm{mean}_A(\cdot)$ is the usual arithmetic mean operation; and S_{CR} is the set of all successful crossover rates CR_i at generation g.

In order to maintain the population diversity, for each target vector the mutation factor F_i is independently calculated as:

$$F_i = \mathrm{rndc}_i(\mu_F, 0.1) \quad (19)$$

and then truncated to be 1.0 if $F_i > 1.0$ or regenerated if $F_i \leq 0$. $\mathrm{rndc}_i(\mu_F, 0.1)$ is a random number generated according to the Cauchy distribution with location parameter μ_F and scale parameter 0.1. The location parameter μ_F is updated in the following manner:

$$\mu_F = (1 - c) \cdot \mu_F + c \cdot \mathrm{mean}_L(S_F) \quad (20)$$

where S_F is the set of all successful mutation factors F_i at generation g; and $\text{mean}_L(\cdot)$ is the Lehmer mean:

$$\text{mean}_L(S_F) = \frac{\sum_{i=1}^{|S_F|} F_i^2}{\sum_{i=1}^{|S_F|} F_i} \tag{21}$$

III. Our Approach

In this section we describe a family of improved DE variants in detail, which implement the strategy adaptation mechanisms (SaMs) for DE, such that they can adaptively choose a more suitable strategy for a specific problem at hand. First, we describe the motivations of this work. Second, three SaMs are presented in detail. Finally, one of the SaMs combined with the JADE method is algorithmically illustrated.

A. Motivations

In the DE algorithm, there are many mutation strategies, however, choosing the best among different mutation strategies available for DE is not easy for a specific problem [10], [11]. Until now, no single mutation strategy has turned out to be best for all problems which, of course, doesn't come as a surprise with regard to the No Free Lunch theorems [29]. To have a better choice of DE's strategies, Feoktistov and Janaqi [30] introduced a generalization of DE's strategies. Their approach leaded to a new universal formula of differentiation. In [31], Iorio and Li proposed a rotation-invariant strategy "DE/current-to-rand/1" to solve the rotated multi-objective optimization problems. Qin and Suganthan [10] proposed a self-adaptive DE algorithm. The aim of their work was to allow DE to switch between two schemes: "DE/rand/1/bin" and "DE/best/2/bin" and also to adapt the F and CR values. Mezura-Montes *et al*. [26] presented an empirical comparison of the generation schemes of DE. Ali and Fatti [32] proposed a point generation scheme that uses an approximation to the probability distribution of trial points in DE. Recently, Qin *et al*. [11] extended their previous work [10]. In their proposed SaDE approach, four mutation strategies were adopted. Different CR values were also used for different strategies. Inspired by the idea of PSO, in [22], Das *et al.* proposed a hybrid mutation strategy, where the global and local manners of the "DE/target-to-best/1/bin" are implemented. In [15], [16], Zhang and Sanderson presented four mutation strategies, in which the high-quality solutions in the current population and/or the archived inferior solutions recently explored can be used to guide the search.

Adaptation or self-adaptation is highly beneficial for adjusting control parameters and operators, especially when done without any user interaction [14]. Two good reviews related to the operator and parameter adaptation of EAs can be found in [33] and [34]. Besides the parameter and operator adaptation in EAs, the adaptation of multiple methods is also efficient. Recently, using the adaptive multiple methods for population evolution has become popular. Ong and Keane [35] proposed an adaptive memetic algorithm, in which a local search (LS) method can be adaptively chosen from the LS pool to locally improve the solutions at runtime. In [35], the reward, which is measured using the relative improvements contributed by the LS to each solution, is adopted to decide which LS method will be selected for the following local improvement. SaDE [10], [11] implements adaptive multi-strategies in the DE framework for global optimization. Vrugt *et al.* [36] proposed a self-adaptive multimethod search for global optimization. In [36], multiple different search algorithms are run concurrently and a self-adaptive learning strategy is implemented to automatically adjust the number of offspring of different algorithms generated at each generation.

To the best of our knowledge, among many variants of DE, the study on using multiple-strategies in DE is scarce. In [12], the authors adopted a neural network to update the weights of different strategies of DE. In [13], Zamuda *et al.* used the uniform selection method to choose the strategy, where a fixed selection probability for each strategy is used. SaDE [11] implements different strategies and updates their weight in the search based on their previous success rate. However, the strategy adaptation method proposed in SaDE is relatively complex to implement. Based on these considerations, we propose three simple approaches to implement the strategy adaptation for DE in the following section. Our approach is different from the previous multiple methods used in EAs.

B. Strategy Adaptation Mechanisms

In our proposed method, for the i-th individual X_i, a strategy parameter, $\eta_i \in [0, 1)$, is used to control the selection of the strategy. Suppose that we have K strategies in the strategy pool, for the i-th target vector its mutation strategy ($S_i = \{1, 2, \cdots, K\}$) is obtained as:

$$S_i = \lfloor \eta_i \times K \rfloor + 1 \tag{22}$$

For example, if $K = 4$ and $\eta_i \in [0, 0.25)$, then $S_i = 1$. It means that, if X_i is the target vector, then the first strategy in the pool will be selected to generate the mutant vector.

To implement the strategy adaptation, we need to address two questions: First, which mutation strategies should be chosen to form the strategy pool? Second, how do we update the strategy parameter η_i?

As mentioned above, there are many strategies of DE, and different strategy has different characteristics. However, there is no theoretical study on the choice of the optimal pool size and the selection of strategies to form the strategy pool until now [11]. In order to select different mutation strategies to form the strategy pool, in this work, we choose four strategies proposed in [15] and [16] to form the strategy pool (i.e., 1. "DE/current-to-$p$best" without archive; 2. "DE/rand-to-$p$best" without archive; 3. "DE/current-to-$p$best" with archive; and 4. "DE/rand-to-$p$best" with archive). These strategies have been introduced in Section II-B. The reasons for choosing these strategies are two-fold. First, they have obtained good performance individually as shown in [15] and [16]. Second, the two strategies without archive converge faster and are more suitable to the low-dimensional problems; on the other hand, the strategies with archive can provide higher population diversity, and hence, they are more suitable to the high-dimensional problems. Note that other strategies can also be possible to be chosen into the pool, these 4 strategies can be treated as instances used as test-bed for the integration method.

In order to address the second question, since η_i is a real parameter, many techniques are possible to handle this parameter. In this work, we introduce two adaptive approaches to update the strategy parameter η_i as follows:

1) The First Approach: This method is inspired by the idea of parameter adaptation of JADE proposed in [15], [16]. At each generation g, for the i-th solution the strategy parameter η_i is independently generated in the following manner:

$$\eta_i = \begin{cases} \text{rndn}_i(\mu_s, 1/6), & \text{if } g = 1 \\ \text{rndn}_i(\mu_s, 0.1), & \text{otherwise} \end{cases} \quad (23)$$

where $\text{rndn}_i(\mu_s, 0.1)$ indicates a normal distribution of mean μ_s and standard deviation 0.1. If $\eta_i \notin [0, 1)$, then it is truncated to $[0, 1)$. At the first generation $g = 1$, the standard deviation is $1/6$ to ensure that the initial η_i is generated in the range $[0, 1)$. If $g > 1$, the standard deviation is set to be 0.1 in the similar way done in [11] and [15]. The reason is that too large standard deviation makes the adaptation not function efficiently [16].

Denote H_s as the set of all successful strategy parameters η_i's at generation g. The mean μ_s is initialized to be 0.5 and then updated at the end of each generation as follows:

$$\mu_s = (1 - c) \times \mu_s + c \times \text{mean}_A(H_s) \quad (24)$$

where c is a positive constant in $[0, 1]$ and $\text{mean}_A(\cdot)$ is the usual arithmetic mean operation. If $c = 0$, no adaptation of strategies takes place. If $c = 1$, only the instant mean value of H_s is active. For other cases $0 < c < 1$, both the previous μ_s and the mean value of H_s are active. It means that both the previous strategy value and the current successful strategy values affect the strategy selection in the next generation.

2) The Second Approach: Inspired by the parameter self-adaptation proposed in jDE [23], this approach calculates as:

$$\eta_i = \begin{cases} \text{rndreal}[0, 1), & \text{rndreal}[0, 1] < \delta \\ \eta_i, & \text{otherwise} \end{cases} \quad (25)$$

where $\text{rndreal}[a, b)$ is a uniformly distributed random number generated in $[a, b)$. $\delta \in [0, 1]$ indicates the probability to adjust the strategy parameter η_i; $\delta = 0.1$ is used in this work.

3) The Uniform Approach: When showing the superiority of the adaptive methods, it is necessary to compare them with the uniform approach, i.e., at each generation a strategy is uniformly selected from the pool for each target vector. The uniform approach can be views as a baseline. The strategy parameter η_i in the uniform approach is calculated as:

$$\eta_i = \text{rndreal}_i[0, 1) \quad (26)$$

Note that according to the classification of parameter control methods used in EAs [34], the first and the second approaches are the adaptive methods. The principle of these two methods is "Better control parameter values tend to generate individuals that are more likely to survive and thus these values should be propagated." [15]. The third method presented here is only a baseline to compare with the adaptive methods.

C. Handling Boundary Constraint of Variables

After using the mutation strategy of DE to generate a new solution, if one or more of the variables in the new solution are beyond their boundaries, i.e. $x_i \notin [L_i, U_i]$, the following repair rule is applied:

$$x_i = \begin{cases} L_i + \text{rndreal}_i[0, 1] \times (U_i - L_i), & \text{if } x_i < L_i \\ U_i - \text{rndreal}_i[0, 1] \times (U_i - L_i), & \text{if } x_i > U_i \end{cases} \quad (27)$$

where $\text{rndreal}_i[0, 1]$ is uniformly distributed random number from [0,1] in the i-th dimension.

Algorithm 2 JADE with Strategy Adaptation Mechanism

```
Initialize the population P randomly
Evaluate the fitness for each individual in P
Set μ_CR = 0.5; μ_F = 0.5; μ_s = 0.5; A = φ; g = 1; K = 4
while The halting criterion is not satisfied do
    S_CR = φ; S_F = φ; H_s = φ                                          ⇐
    for i = 1 to NP do
        Generate η_i according to Eqn. (23)                             ⇐
        Calculate the strategy index S_i = ⌊η_i × K⌋ + 1                ⇐
        Randomly choose x^p_best as one of the 100p% best solutions
        Generate CR_i using Eqn. (17)
        if S_i == 2 or S_i == 4 then
            F_i = rndn_i(μ_F, 0.1)                                      ⇐
        else
            F_i = rndc_i(μ_F, 0.1)
        end if
        if S_i == 1 or S_i == 2 then
            Select r_1, r_2, r_3 from P randomly r_1 ≠ r_2 ≠ r_3 ≠ i
        else if S_i == 3 or S_i == 4 then
            Select r_1, r_2 from P randomly r_1 ≠ r_2 ≠ i; and select
            r_3 from P ∪ A
        end if
        j_rand = rndint(1, D)
        for j = 1 to D do
            if rndreal_j[0, 1) < CR or j == j_rand then
                if S_i == 1 then
                    u_{i,j} = x_{i,j} + F_i(x^p_{best,j} − x_{i,j}) + F_i(x_{r_2,j} − x_{r_3,j})
                else if S_i == 2 then
                    u_{i,j} = x_{r_1,j} + F_i(x^p_{best,j} − x_{r_1,j}) + F_i(x_{r_2,j} − x_{r_3,j})
                else if S_i == 3 then
                    u_{i,j} = x_{i,j} + F_i(x^p_{best,j} − x_{i,j}) + F_i(x_{r_2,j} − x̃_{r_3,j})
                else if S_i == 4 then
                    u_{i,j} = x_{r_1,j} + F_i(x^p_{best,j} − x_{r_1,j}) + F_i(x_{r_2,j} − x̃_{r_3,j})
                end if
            else
                u_{i,j} = x_{i,j}
            end if
        end for
    end for
    for i = 1 to NP do
        Evaluate the offspring u_i
        if f(u_i) is better than or equal to f(x_i) then
            Update the archive A with the inferior solution x_i
            CR_i → S_CR; F_i → S_F; η_i → H_s                           ⇐
            Replace x_i with u_i
        end if
    end for
    Update the μ_CR, μ_F, and μ_s                                       ⇐
    g = g + 1
end while
```

D. DE with Strategy Adaptation

To make the description clearer, the pseudo-code of the proposed DE with the first SaM is shown in Algorithm 2. In this work, the parameter adaptation mechanism proposed in JADE [15] is used. To this point of view, this approach can be regarded as an improved JADE variant. Modified steps with

TABLE I

THE 16 BENCHMARK FUNCTIONS USED IN OUR EXPERIMENTAL STUDY, WHERE D IS THE NUMBER OF VARIABLES AND $S \subseteq \mathbb{R}^D$. EACH OF THEM HAS A GLOBAL MINIMUM VALUE OF 0. A DETAIL DESCRIPTION OF ALL FUNCTIONS CAN BE FOUND IN [19] AND [37].

Name	Test Functions	S
Sphere	$f_{01} = \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$
Schwefel 2.22	$f_{02} = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	$[-10, 10]^D$
Schwefel 1.2	$f_{03} = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)^2$	$[-100, 100]^D$
Schwefel 2.21	$f_{04} = \max_i \{\lvert x_i \rvert, 1 \le i \le D\}$	$[-100, 100]^D$
Rosenbrock	$f_{05} = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-30, 30]^D$
Step	$f_{06} = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	$[-100, 100]^D$
Quartic	$f_{07} = \sum_{i=1}^{D} x_i^4 + random[0, 1)$	$[-1.28, 1.28]^D$
Schwefel 2.26	$f_{08} = \sum_{i=1}^{D} (-x_i \sin(\sqrt{\lvert x_i \rvert})) + 418.98288727243369 \times D$	$[-500, 500]^D$
Rastrigin	$f_{09} = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10)$	$[-5.12, 5.12]^D$
Ackley	$f_{10} = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + \exp(1)$	$[-32, 32]^D$
Griewank	$f_{11} = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}}) + 1$	$[-600, 600]^D$
Penalized 1	$f_{12} = \frac{\pi}{D}\{10\sin^2(\pi y_i) + \sum_{i=1}^{D-1}(y_i - 1)^2 \cdot [1 + 10\sin^2(\pi y_{i+1})] + (y_D - 1)^2\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$	$[-50, 50]^D$
Penalized 2	$f_{13} = \frac{1}{10}\{\sin^2(3\pi x_1) + \sum_{i=1}^{D-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_D - 1)^2[1 + \sin^2(2\pi x_D)]\} + \sum_{i=1}^{D} u(x_i, 5, 100, 4)$	$[-50, 50]^D$
Neumaier 3	$f_{14} = \sum_{i=1}^{D}(x_i - 1)^2 + \sum_{i=2}^{D} x_i x_{i-1} + \frac{D(D+4)(D-1)}{6}$	$[-D^2, D^2]^D$
Salomon	$f_{15} = 1 - \cos(2\pi \parallel x \parallel) + 0.1 \parallel x \parallel$, where $\parallel x \parallel = \sum_{i=1}^{D} x_i$	$[-100, 100]^D$
Alpine	$f_{16} = \sum_{i=1}^{D} \lvert x_i \sin x_i + 0.1 x_i \rvert$	$[-10, 10]^D$

respect to JADE are marked with a left arrow "⇐". Our approach is referred to as SaJADE, i.e., the **S**trategy **a**daptation-based **JADE** method. It is worth pointing out that our proposed SaM can also be used in other DE variants. Moreover, in our proposed SaM, different parameter adaptation methods of DE can be adopted for different strategies according to their characteristics. For example, in Algorithm 2, we can see that for the "DE/rand-to-$p$best" strategies the mutation factor $F_i = \mathrm{rndn}_i(\mu_F, 0.1)$. The reason is that "DE/rand-to-$p$best" can provide higher population diversity, the relatively small F_i values are able to obtain faster convergence rate for the low- and moderate-dimensional problems. The mean value of μ_F is also updated as Eqn. (20).

Compared with the variants of DE proposed in [12], [13], and [11], where the multi-strategies of DE are also used, the main differences between our proposed SaMs and these variants are as follows:

- Our SaMs are controlled by one single strategy parameter η, which is a real parameter. It can be adjusted adaptively. Other parameter adaptation techniques in EAs are also possible to be used to update the strategy parameter η.
- Compared with SWAF [12], the weights of strategies are updated by the neural network based on the previous successful rates. This approach is relatively complex to implement. Also, there are some parameters of the neural network that need to be fine-tuned.
- In [13], Zamuda *et al.* presented a multiple-strategies DE variant. However, for each strategy they only used the pre-defined selection probability; and then a uniform randomly generated parameter r_s is used to determine which strategy will be selected. Thus, this method is not a strategy adaptation approach.
- In SaDE [11], each strategy has its own probability, which is updated by Eqns (9) and (10) according to previous experiences. It is relatively complex to implement.
- In general, our proposed three approaches are different from the above-mentioned variants. They are very simple and easy to implement.

On the complexity of SaJADE shown in Algorithm 2, our algorithm does not increase the overall complexity with respect to JADE. The additional complexity of SaJADE is the parameter adaptation of η_i, which takes $O(NP)$ operations. Since the total complexity of JADE is $O(G \cdot NP \cdot (D + \log(NP)))$ [16], where G is the maximal number of generations, SaJADE has the same total complexity of $O(G \cdot NP \cdot (D + \log(NP)))$. In general, the population size NP is set to be the proportional to the problem dimension D in the DE literature. Thus, the total complexity of SaJADE is $O(G \cdot D^2)$, which is the same as the classic DE algorithm, JADE, and many other DE variants.

IV. EXPERIMENTAL RESULTS

In order to verify the performance of our proposed SaM, twenty scalable benchmark functions are chosen from the literature as the test suit. Functions $f_{01} - f_{13}$ are chosen from [19]. Functions $f_{14} - f_{16}$ are selected from [37]. The rest four functions (F_{06}, F_{07}, F_{09}, and F_{10}) are selected from [38]. For functions $f_{01} - f_{16}$, they are briefly described in Table I. For functions F_{06}, F_{07}, F_{09}, and F_{10}, they can be found in [38]. A more detailed description of these functions can be found in [19], [37], [25], and [38].

Functions $f_{01} - f_{04}$ are unimodal. The Rosenbrock's function f_{05} is a multi-modal function when $D > 3$ [39]. Function f_{06} is the step function, which has one minimum and is discontinuous. Function f_{07} is a noisy quartic function. Functions $f_{08} - f_{16}$ are multi-modal functions where the number of local minima increases exponentially with the problem dimension. They appear to be the most difficult class of problems for many optimization algorithms. Functions F_{06}, F_{07}, F_{09}, and F_{10} are multi-modal. Function F_{09} are separable, and the remaining 3 functions are non-separable. The shifted and/or rotated features make these 4 functions very difficult to solve.

A. Experimental Setup

In the experiments, we first compare the performance of different strategy adaptation methods proposed in Section III-B. Hence two approaches combined with JADE are implemented: SaJADE1 with the first SaM and SaJADE2 with the second SaM. We also implement the strategy adaptation method with learning period $LP = 50$ proposed in [11] into JADE, namely SaJADE3. In addition, JADE with the uniform strategy selection (Eqn. (26)) is implemented, namely Uniform-JADE, as a baseline. In Uniform-JADE, the four strategies used in SaJADE are also adopted. Secondly, we compare the performance of SaJADE (with the first SaM) with those of jDE [23], SaDE [11], JADE-wo, and JADE-w [15] directly, where JADE-wo means JADE without archive and JADE-w means JADE with archive; both algorithms adopt the "DE/current-to-$p$best" strategy [16]. For all experiments, we use the following parameters unless a change is mentioned[3].

- Dimension of each function: $D = 30$ and $D = 100$;
- Population size: $NP = 100$, if $D = 30$; $NP = 400$, if $D = 100$ [15], [16];
- $\mu_{CR} = 0.5, \mu_F = 0.5$, and $\mu_s = 0.5$ [15], [16];
- $c = 0.1$ and $p = 0.05$ [15], [16];
- Value to reach: For functions $f_{01} - f_{06}$ and $f_{08} - f_{16}$, VTR = 10^{-8}; for functions $f_{07}, F_{06}, F_{07}, F_{09}$, and F_{10}, VTR = 10^{-2} [38], [15];
- Max_NFFEs[4]: If $D = 30$: For $f_{01}, f_{06}, f_{10}, f_{12}$, and f_{13}, Max_NFFEs = $150,000$; for $f_{03} - f_{05}$, Max_NFFEs = $500,000$; for f_{02} and f_{11}, Max_NFFEs = $200,000$; for $f_{07} - f_{09}$, $f_{14} - f_{16}$, and $F_{06} - F_{10}$, Max_NFFEs = $300,000$. If $D = 100$: For all functions Max_NFFEs = $1,000,000$ (i.e., $D \times 10,000$) [38].

Moreover, in our experiments, each function is optimized over 50 independent runs[5]. We also use the same set of initial random populations to evaluate different algorithms in a similar way done in [24].

B. Performance Criteria

Five performance criteria are selected from the literature [38], [25] to evaluate the performance of the algorithms. These criteria are described as follows.

- **Error** [38]: The error of a solution $\mathbf{x}$ is defined as $f(\mathbf{x}) - f(\mathbf{x}^*)$, where $\mathbf{x}^*$ is the global minimum of the function. The minimum error is recorded when the Max_NFFEs is reached in 50 runs. The average and standard deviation of the error values are calculated as well.
- **NFFEs** [38]: The NFFEs is also recorded when the VTR is reached. The average and standard deviation of the NFFEs values are calculated.
- **Successful rate (S_r)** [38]: The successful run of an algorithm indicates that the algorithm can result in a function value no worse than the VTR before the Max_NFFEs condition terminates the trial. The successful rate S_r is calculated as the number of successful runs divided by the total number of runs.
- **Convergence graphs** [38]: The convergence graphs show the median error performance of the best solution over the total runs, in the respective experiments.
- **Acceleration rate (AR)** [25]: This criterion is used to compare the convergence speeds between our approach and other algorithms. It is defined as follows: $AR = \frac{\mathrm{NFFEs}_{\mathrm{other}}}{\mathrm{NFFEs}_{\mathrm{SaJADE}}}$, where $AR > 1$ indicates our approach is faster than its competitor.

C. Comparison on Different Strategy Adaptation Methods

In this section, the performance of SaJADE1, SaJADE2, SaJADE3, and Uniform-JADE is compared to show the superiority of the adaptive strategy selection methods. The parameters for all algorithms are the same as described in Section IV-A. The results of all functions at $D = 30$ are tabulated in Table II. The best and the second best results are highlighted in **boldface** and *italic*, respectively. Since for most of the functions the four algorithms can solve them, the NFFEs are used to compare them. Except for functions f_{15} and F_{10}, the error values of the final solutions are used, because no algorithm can solve the two functions.

From Table II, we can see that SaJADE1 (JADE with the first SaM) obtains the overall best results. It ranks 1 on 17 out of 20 functions in terms of the NFFEs. In addition, SaJADE1 is able to provide the greatest overall successful rate ($\sum S_r = 17.48$). The SaJADE3 method obtains the second overall best results with respect to the NFFEs, followed by SaJADE2. Furthermore, Table II also indicates that all of the three SaJADE methods is better than Uniform-JADE. This confirms the superiority of the strategy adaptation approaches.

Since SaJADE1 outperforms all other strategy adaptation methods, in the following sections, we only compare the results of SaJADE1 (referred to as SaJADE because of no confusion) with those of other DE variants.

D. Comparison of SaJADE with Other DE Variants

In this section we compare the performance of SaJADE with that of jDE, SaDE, JADE-wo, and JADE-w in terms of three aspects: i) the quality of the final solutions; ii) the convergence speed; and iii) the success rate S_r. The parameters of all algorithms are used as mentioned in Section IV-A. For all test functions, the dimensions of $D = 30$ and $D = 100$ are used. In addition, JADE-wo and JADE-w with the "DE/rand-to-$p$best/1" strategy (referred to as rJADE-wo and rJADE-w, respectively) are also tested on all problems. However, due to the tight space limitation, we do not report the results but only show the convergence curves in Figure 1 on the selected functions. The paired Wilcoxon signed-rank test at $\alpha = 0.05$ is adopted to compare the significance between two algorithms. The Wilcoxon's test is a non-parametric statistical hypothesis test, which can be used as an alternative to the paired t-test when the results cannot be assumed to be normally distributed [40]. There are two reasons to use the Wilcoxon's

[3]For jDE and SaDE, some specific parameters (e.g., τ_1, τ_2 in jDE and LP in SaDE) are set as in [23] and [11], respectively.

[4]The Max_NFFEs for functions $f_{01} - f_{13}$ are mainly set as in [19], except for f_{05}, f_{08}, and f_{09}, they are less than the values in [19], since SaJADE is able to obtain the global optimum of these functions within the Max_NFFEs. For functions $F_{06} - F_{10}$, the Max_NFFEs are set as in [38].

[5]All the algorithms are implemented in standard C++. The source code may be obtained from the authors upon request.

TABLE II

THE PERFORMANCE COMPARISON OF DIFFERENT STRATEGY ADAPTATION METHODS FOR ALL FUNCTIONS AT $D = 30$.

F	SaJADE1			SaJADE2			SaJADE3			Uniform-JADE		
	Mean	Std	S_r	Mean	Std	S_r	Mean	Std	S_r	Mean	Std	S_r
f_{01}	**2.40E+04**	**5.55E+02**	1.00	2.88E+04	8.83E+02	1.00	*2.82E+04*	*7.21E+02*	1.00	2.89E+04	8.04E+02	1.00
f_{02}	**3.90E+04**	**1.44E+03**	1.00	5.01E+04	2.46E+03	1.00	*4.91E+04*	*2.07E+03*	1.00	5.10E+04	2.10E+03	1.00
f_{03}	**7.88E+04**	**3.63E+03**	1.00	*8.79E+04*	*4.27E+03*	1.00	8.97E+04	4.59E+03	1.00	8.94E+04	5.09E+03	1.00
f_{04}	**2.09E+05**	**8.25E+03**	1.00	2.88E+05	6.38E+03	1.00	*2.86E+05*	*6.95E+03*	1.00	2.89E+05	6.35E+03	1.00
f_{05}	**1.18E+05**	**3.61E+03**	1.00	1.26E+05	5.08E+03	0.98	*1.26E+05*	*3.70E+03*	0.98	1.27E+05	3.72E+03	0.92
f_{06}	**9.20E+03**	**2.25E+02**	1.00	1.07E+04	3.94E+02	1.00	*1.05E+04*	*3.26E+02*	1.00	1.06E+04	3.85E+02	1.00
f_{07}	**2.26E+04**	**4.59E+03**	1.00	*2.61E+04*	*5.55E+03*	1.00	2.71E+04	6.07E+03	1.00	2.65E+04	4.89E+03	1.00
f_{08}	**1.01E+05**	**3.98E+03**	1.00	1.06E+05	2.11E+03	1.00	*1.04E+05*	*2.54E+03*	1.00	1.07E+05	2.05E+03	1.00
f_{09}	**1.27E+05**	**4.16E+03**	1.00	1.32E+05	2.29E+03	1.00	*1.31E+05*	*2.40E+03*	1.00	1.31E+05	2.74E+03	1.00
f_{10}	**3.61E+04**	**8.47E+02**	1.00	4.45E+04	1.21E+03	1.00	*4.41E+04*	*1.51E+03*	1.00	4.47E+04	1.71E+03	1.00
f_{11}	**2.51E+04**	**7.64E+02**	1.00	3.07E+04	1.40E+03	1.00	3.03E+04	3.10E+03	1.00	*3.02E+04*	*9.57E+02*	1.00
f_{12}	**2.17E+04**	**7.32E+02**	1.00	2.63E+04	1.24E+03	1.00	*2.60E+04*	*1.18E+03*	1.00	2.65E+04	9.63E+02	1.00
f_{13}	**2.55E+04**	**1.07E+03**	1.00	3.32E+04	2.04E+03	1.00	3.19E+04	1.80E+03	1.00	*3.17E+04*	*1.69E+03*	1.00
f_{14}	*2.18E+05*	*2.05E+04*	1.00	**2.15E+05**	**2.50E+04**	1.00	2.23E+05	2.74E+04	1.00	2.25E+05	2.46E+04	1.00
$f_{15}{}^{\star}$	1.76E-01	4.28E-02	0.00	**1.60E-01**	**4.95E-02**	0.00	*1.74E-01*	*4.43E-02*	0.00	1.77E-01	4.43E-02	0.00
f_{16}	**1.51E+05**	**7.01E+04**	0.72	2.89E+05	0.00E+00	0.02	*2.34E+05*	*2.82E+04*	0.06	2.54E+05	5.40E+04	0.10
F_{06}	**1.04E+05**	**7.97E+03**	0.96	1.13E+05	8.37E+03	0.86	*1.12E+05*	*8.56E+03*	0.92	1.15E+05	7.44E+03	0.80
F_{07}	**3.29E+04**	**4.02E+03**	0.80	*3.57E+04*	*4.74E+03*	0.74	3.57E+04	5.29E+03	0.80	3.63E+04	4.74E+03	0.74
F_{09}	**1.04E+05**	**2.76E+03**	1.00	1.06E+05	2.42E+03	1.00	*1.05E+05*	*2.29E+03*	1.00	1.07E+05	2.24E+03	1.00
$F_{10}{}^{\star}$	**2.66E+01**	**4.44E+00**	0.00	*2.68E+01*	*5.20E+00*	0.00	3.11E+01	5.60E+00	0.00	3.19E+01	5.99E+00	0.00
$\sum S_r$	17.48			16.60			16.76			16.56		

* indicates that the error values of the final solutions are used, since no algorithm can solve the corresponding problem within the Max_NFFEs.

TABLE III

MEAN AND STANDARD DEVIATION OF THE ERROR VALUES OF THE BEST-SO-FAR SOLUTIONS OVER 50 INDEPENDENT RUNS FOR ALL TEST FUNCTIONS AT $D = 30$.

F	Max_NFFEs	jDE	JADE-wo	JADE-w	SaDE	SaJADE
f_{01}	150,000	1.46E-28 ± 1.78E-28†	*9.93E-62 ± 5.34E-61*†	2.69E-56 ± 1.41E-55†	3.42E-37 ± 3.63E-37†	**1.10E-79 ± 7.52E-79**
f_{02}	200,000	9.02E-24 ± 6.01E-24†	*5.53E-28 ± 3.16E-27*†	3.18E-25 ± 2.05E-24†	3.51E-25 ± 2.74E-25†	**1.35E-47 ± 7.53E-47**
f_{03}	500,000	1.16E-13 ± 1.73E-13†	1.93E-56 ± 7.02E-56†	**6.11E-81 ± 1.62E-80**‡	1.54E-14 ± 4.56E-14†	*1.17E-77 ± 3.39E-77*
f_{04}	500,000	2.44E-14 ± 1.65E-13†	1.34E-09 ± 5.71E-10†	5.29E-14 ± 2.05E-14†	**6.39E-27 ± 8.27E-27**‡	*1.26E-19 ± 1.35E-19*
f_{05}	500,000	*1.04E-03 ± 1.37E-03*†	4.78E-01 ± 1.31E+00†	1.59E-01 ± 7.89E-01	7.98E-02 ± 5.64E-01†	**1.60E-30 ± 6.32E-30**
f_{06}	10,000	6.13E+02 ± 1.72E+02†	*3.12E+00 ± 1.54E+00*†	5.62E+00 ± 1.87E+00†	5.07E+01 ± 1.34E+01†	**0.00E+00 ± 0.00E+00**
	150,000	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
f_{07}	300,000	3.35E-03 ± 8.68E-04†	6.59E-04 ± 2.43E-04†	*6.14E-04 ± 2.55E-04*†	2.06E-03 ± 5.21E-04†	**4.10E-04 ± 1.48E-04**
f_{08}	100,000	**1.70E-10 ± 1.71E-10**‡	4.14E-05 ± 2.37E-05†	2.62E-04 ± 3.59E-04†	*1.13E-08 ± 1.08E-08*‡	6.83E-07 ± 2.70E-06
	300,000	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
f_{09}	100,000	**3.32E-04 ± 6.39E-04**‡	*2.68E-03 ± 1.90E-03*‡	1.33E-01 ± 9.74E-02	2.43E+00 ± 1.60E+00†	1.54E-01 ± 2.25E-01
	300,000	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
f_{10}	50,000	2.37E-04 ± 7.10E-05†	*1.10E-09 ± 7.45E-10*†	3.35E-09 ± 2.84E-09†	3.81E-06 ± 8.26E-07†	**1.12E-12 ± 1.07E-12**
	150,000	8.26E-15 ± 1.32E-15	4.14E-15 ± 0.00E+00	4.14E-15 ± 0.00E+00	4.14E-15 ± 0.00E+00	4.14E-15 ± 0.00E+00
f_{11}	50,000	7.29E-06 ± 1.05E-05†	*1.44E-14 ± 7.05E-14*†	1.57E-08 ± 1.09E-07†	2.52E-09 ± 1.24E-08†	**0.00E+00 ± 0.00E+00**
	200,000	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
f_{12}	50,000	7.03E-08 ± 5.74E-08†	*1.84E-17 ± 4.52E-17*†	1.67E-15 ± 1.02E-14†	8.25E-12 ± 5.12E-12†	**2.10E-23 ± 6.89E-23**
	150,000	5.99E-30 ± 5.87E-30	1.57E-32 ± 0.00E+00	1.57E-32 ± 0.00E+00	1.57E-32 ± 0.00E+00	1.57E-32 ± 0.00E+00
f_{13}	50,000	1.80E-05 ± 1.42E-05†	*3.16E-13 ± 9.79E-13*†	1.87E-10 ± 1.09E-09†	1.93E-09 ± 1.53E-09†	**3.83E-21 ± 1.56E-20**
	150,000	1.80E-27 ± 2.62E-27	1.35E-32 ± 0.00E+00	1.35E-32 ± 0.00E+00	1.35E-32 ± 0.00E+00	1.35E-32 ± 0.00E+00
f_{14}	300,000	7.31E-01 ± 1.19E+00†	1.72E-03 ± 3.05E-03†	**1.68E-09 ± 1.97E-09**‡	1.25E+02 ± 2.68E+02†	*2.88E-09 ± 2.43E-09*
f_{15}	300,000	1.98E-01 ± 1.41E-02†	2.02E-01 ± 1.41E-02†	2.00E-01 ± 1.63E-02†	**1.56E-01 ± 5.01E-02**	*1.76E-01 ± 4.28E-02*
f_{16}	300,000	**6.08E-10 ± 8.36E-10**	2.61E-06 ± 1.21E-06†	2.78E-05 ± 8.43E-06†	2.94E-06 ± 3.47E-06†	*1.44E-07 ± 4.92E-07*
F_{06}	300,000	2.93E+01 ± 2.79E+01†	7.00E+00 ± 1.87E+01†	*2.56E+00 ± 6.22E+00*	1.68E+01 ± 2.60E+01†	**1.59E-01 ± 7.89E-01**
F_{07}	300,000	1.17E-02 ± 9.90E-03	1.57E-02 ± 1.13E-02†	**5.96E-03 ± 7.39E-03**‡	1.54E-02 ± 9.60E-03†	*1.04E-02 ± 8.48E-03*
F_{09}	100,000	**1.30E-05 ± 3.17E-05**‡	*2.87E-03 ± 1.82E-03*‡	1.35E+00 ± 6.08E-01†	1.46E+00 ± 1.02E+00†	1.13E-01 ± 1.60E-01
	300,000	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
F_{10}	300,000	5.54E+01 ± 9.44E+00†	2.88E+01 ± 5.33E+00†	*2.82E+01 ± 5.32E+00*	7.57E+01 ± 1.02E+01†	**2.66E+01 ± 4.44E+00**
$w/t/l$		15/2/3	18/0/2	13/4/3	17/1/2	–

† indicates SaJADE is significantly better than its competitor by the Wilcoxon signed-rank test at $\alpha = 0.05$.
‡ means that the corresponding algorithm is better than our proposed SaJADE method.

test: i) Although the t-test, one of the parametric statistical test, is popular in evolutionary computing [19], [22], however, recent studies indicate that the parametric statistical analysis is not appreciate especially when tackling the multiple-problem results [41], [42], [43], [44]. ii) The Wilcoxon's test is employed since this test is included in well-known software packages (e.g., SPSS, SAR, OriginPro, Matlab, etc.).

1) Comparison on the Quality of the Final Solutions: For all test functions, the mean and standard deviation of the error values of the best-so-far solutions over 50 independent runs are respectively summarized in Tables III and IV at $D = 30$ and $D = 100$. Similar to the methods used in [15], the intermediate results are also reported for the functions where several algorithms can obtain the global optimum of these functions. In these cases, the Wilcoxon signed-rank test is only compared with the intermediate results. In the last row of each table, according to the Wilcoxon's test, the results are summarized as "$w/t/l$", which means that SaJADE wins

TABLE IV

MEAN AND STANDARD DEVIATION OF THE ERROR VALUES OF THE BEST-SO-FAR SOLUTIONS OVER 50 INDEPENDENT RUNS FOR ALL TEST FUNCTIONS AT $D = 100$.

F	Max_NFFEs	jDE	JADE-wo	JADE-w	SaDE	SaJADE
f_{01}	1,000,000	2.09E-20 ± 9.27E-21†	5.13E-62 ± 4.82E-62†	*1.21E-85 ± 2.27E-85*†	1.09E-27 ± 6.65E-28†	**1.62E-92 ± 2.92E-92**
f_{02}	1,000,000	1.82E-12 ± 4.30E-13†	5.19E-36 ± 7.12E-36†	*9.20E-42 ± 2.96E-41*†	1.09E-15 ± 2.10E-16†	**4.03E-51 ± 1.41E-50**
f_{03}	1,000,000	7.47E+03 ± 7.43E+03†	6.85E-03 ± 5.87E-03†	**4.79E-05 ± 4.63E-05**	4.96E+00 ± 1.61E+00†	*5.06E-05 ± 5.74E-05*
f_{04}	1,000,000	1.60E+00 ± 1.34E-01†	1.62E-01 ± 3.05E-02†	*3.09E-03 ± 3.90E-03*	1.90E-01 ± 1.80E-01†	**2.42E-03 ± 3.73E-03**
f_{05}	1,000,000	9.20E+01 ± 1.41E+01†	4.96E+01 ± 1.15E+01†	*2.77E+01 ± 6.81E+00*†	8.49E+01 ± 1.04E+01†	**2.45E+01 ± 1.43E+00**
f_{06}	40,000	3.18E+04 ± 2.52E+03†	*1.14E+02 ± 1.42E+01*†	1.25E+02 ± 1.40E+01†	1.59E+03 ± 1.70E+02†	**4.19E+01 ± 5.27E+00**
	1,000,000	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
f_{07}	1,000,000	2.08E-02 ± 2.88E-03†	2.04E-03 ± 4.28E-04†	*1.60E-03 ± 3.33E-04*†	6.85E-03 ± 1.34E-03†	**8.59E-04 ± 1.39E-04**
f_{08}	1,000,000	**2.81E-08 ± 2.30E-08**‡	3.94E+03 ± 2.75E+02‡	9.11E+03 ± 4.18E+02	*1.89E+01 ± 3.51E+01*‡	9.04E+03 ± 3.88E+02
f_{09}	1,000,000	**6.01E+00 ± 2.36E+00**‡	*1.03E+02 ± 3.95E+00*‡	1.82E+02 ± 8.44E+00†	1.05E+02 ± 4.84E+00‡	1.67E+02 ± 7.59E+00
f_{10}	200,000	4.20E-01 ± 5.34E-02†	7.79E-06 ± 1.89E-06†	*4.05E-07 ± 1.06E-07*†	5.98E-03 ± 6.95E-04†	**7.28E-09 ± 3.49E-09**
	1,000,000	1.73E-11 ± 3.15E-12	7.69E-15 ± 0.00E+00	7.84E-15 ± 7.03E-16	1.05E-14 ± 1.76E-15	7.69E-15 ± 0.00E+00
f_{11}	200,000	9.25E-01 ± 5.88E-02†	8.87E-04 ± 3.22E-03†	*3.54E-10 ± 2.34E-09*†	3.84E-03 ± 8.15E-03†	**2.39E-15 ± 2.80E-15**
	1,000,000	0.00E+00 ± 0.00E+00	8.87E-04 ± 3.25E-03	0.00E+00 ± 0.00E+00	2.96E-04 ± 1.46E-03	0.00E+00 ± 0.00E+00
f_{12}	200,000	1.44E+00 ± 3.11E-01†	2.38E-11 ± 1.20E-11†	*4.62E-14 ± 3.70E-13*†	8.96E-06 ± 2.10E-06†	**1.66E-17 ± 1.48E-17**
	1,000,000	4.47E-21 ± 1.90E-21	4.71E-33 ± 0.00E+00	4.71E-33 ± 0.00E+00	6.75E-30 ± 4.63E-30	4.71E-33 ± 0.00E+00
f_{13}	200,000	6.04E+01 ± 1.10E+01†	2.83E-08 ± 3.48E-08†	*1.13E-10 ± 1.58E-10*†	7.81E-03 ± 3.27E-03†	**7.24E-15 ± 1.01E-14**
	1,000,000	1.91E-17 ± 1.08E-17	1.35E-32 ± 0.00E+00	1.35E-32 ± 0.00E+00	5.56E-27 ± 5.23E-27	1.35E-32 ± 0.00E+00
f_{14}	1,000,000	2.07E+05 ± 4.46E+04†	1.51E+05 ± 1.98E+04†	**7.90E+04 ± 1.43E+04**	1.69E+05 ± 1.39E+04†	*8.20E+04 ± 1.47E+04*
f_{15}	1,000,000	3.80E-01 ± 3.93E-02†	3.28E-01 ± 4.54E-02†	*2.98E-01 ± 1.41E-02*†	3.60E-01 ± 4.93E-02†	**2.66E-01 ± 4.79E-02**
f_{16}	1,000,000	4.78E-03 ± 4.63E-04†	*1.12E-11 ± 5.08E-11*†	9.55E-05 ± 3.98E-04†	5.78E-03 ± 1.71E-03†	**1.84E-23 ± 1.30E-22**
F_{06}	1,000,000	8.90E+01 ± 4.16E-01†	1.26E+02 ± 2.92E+01†	*3.60E+01 ± 2.78E+01*	1.85E+02 ± 4.26E+01†	**3.14E+01 ± 2.83E+01**
F_{07}	1,000,000	6.68E-01 ± 1.33E-01†	1.90E-01 ± 2.83E-01	*8.50E-02 ± 4.50E-01*	2.81E-01 ± 3.50E-01†	**7.29E-02 ± 2.04E-01**
F_{09}	1,000,000	**1.37E-01 ± 1.26E-01**‡	1.13E+02 ± 5.72E+00‡	2.00E+02 ± 6.26E+00†	*1.07E+02 ± 5.24E+00*‡	1.77E+02 ± 7.07E+00
F_{10}	1,000,000	4.76E+02 ± 2.71E+01†	**4.22E+02 ± 1.52E+01**‡	4.67E+02 ± 2.93E+01†	5.31E+02 ± 2.31E+01†	*4.51E+02 ± 1.73E+01*
$w/t/l$		17/0/3	15/1/4	14/6/0	17/0/3	–

† indicates SaJADE is significantly better than its competitor by the Wilcoxon signed-rank test at $\alpha = 0.05$.
‡ means that the corresponding algorithm is better than our proposed SaJADE method.

in w functions, ties in t functions, and loses in l functions, compared with its competitors.

From Table III, it is clear that our approach is able to obtain consistently better error values of the best-so-far solutions than its competitors. SaJADE significantly outperforms jDE, JADE-wo, JADE-w, and SaDE on $15, 18, 13$, and 17 out of 20 functions, respectively. For functions f_{08}, f_{09}, and F_{09}, jDE is significantly better than SaJADE. SaJADE is worse than JADE-wo for 2 functions, JADE-w for 3 functions, and SaDE for 2 functions.

For the functions at $D = 100$, according to Table IV, a similar conclusion to SaJADE can be drawn about the error values of jDE, JADE-wo, JADE-w, and SaDE, i.e., on the majority of the functions, our approach performs significantly better than other DE variants.

In general, the SaJADE approach is able to provide the overall highest quality of the final solution among other DE variants for functions at $D = 30$ and $D = 100$. Our proposed strategy adaptation can enhance the performance of JADE in terms of the quality of the final solution.

2) Multiple-problem Statistical Analysis: In Tables II and III, only the single-problem statistical analysis by the Wilcoxon signed-rank test is used. As stated in [44], the multiple-problem statistical analysis is also important to check the behavior of the stochastic algorithms. In this part, we present the results of the multiple-problem Wilcoxon signed-rank test at $\alpha = 0.05$ in Table V. Note that when several algorithms can obtain the global optimum of a specific function, only the mean values of the intermediate results are considered from Tables II and III; otherwise, the mean values of the final solutions are used. Hence, there are overall 40 functions (20 functions at $D = 30$ and 20 functions at $D = 100$) to make the multiple-problem statistical analysis. From Table V, it is clear that SaJADE obtains higher R^+ values than R^- values in all cases. According to the p-value, we can see that SaJADE is significantly better than other DE variants, since in all cases the p-values are less than 0.05. The results reconfirm that the overall performance of SaJADE is better than other compared DE variants in terms of the quality of the final solutions.

TABLE V

MULTI-PROBLEM STATISTICAL ANALYSIS BY THE WILCOXON SIGNED-RANK TEST AT $\alpha = 0.05$ (SAJADE TO ITS COMPETITORS).

	R^+	R^-	p-value	significant
jDE	656	164	6.38E-04	Yes
JADE-wo	624	196	3.36E-03	Yes
JADE-w	702	118	3.30E-05	Yes
SaDE	679	141	1.60E-04	Yes

3) Comparison on the Convergence Speed and Successful Rate: Besides the quality of the final solutions, the convergence velocity and successful rate S_r are also very important to measure the performance of an algorithm. Tables VI and VIII respectively summarize the mean and standard deviation of the NFFEs of the successful runs at $D = 30$ and $D = 100$. In addition, the successful rate S_r is also shown in these two tables (within parentheses). Moreover, in Tables VII and IX the AR values are tabulated for the functions that have been solved by several algorithms. Some representative convergence graphs of jDE, JADE-wo, JADE-w, rJADE-wo, rJADE-w, SaDE, and SaJADE are shown in Figure 1.

From Tables VI and VIII, we can see that SaJADE requires the overall lowest NFFEs to reach the VTR on the majority of

TABLE VII
THE AR VALUES AT $D = 30$ (SAJADE TO ITS COMPETITORS).

F	jDE	JADE-wo	JADE-w	SaDE
f_{01}	2.46	1.21	1.26	1.82
f_{02}	2.08	1.30	1.41	1.84
f_{03}	4.52	1.29	0.99	3.95
f_{04}	1.40	2.16	1.48	0.81
f_{05}	3.95	1.30	1.04	2.38
f_{06}	2.42	1.18	1.25	1.72
f_{07}	4.66	1.23	1.32	2.38
f_{08}	0.90	1.20	1.17	0.99
f_{09}	0.92	1.04	1.13	1.07
f_{10}	2.48	1.26	1.31	1.90
f_{11}	2.47	1.28	1.37	1.83
f_{12}	2.46	1.27	1.34	1.79
f_{13}	2.52	1.38	1.48	1.85
f_{14}	NA	NA	0.96	NA
f_{16}	1.69	NA	NA	1.88
F_{06}	2.44	1.29	1.05	2.37
F_{07}	3.08	1.37	1.06	3.08
F_{09}	0.80	0.92	1.11	1.05
Avg	2.1671	1.1544	1.0988	1.7247

TABLE IX
THE AR VALUES AT $D = 100$ (SAJADE TO ITS COMPETITORS).

F	jDE	JADE-wo	JADE-w	SaDE
f_{01}	4.01	1.46	1.22	2.74
f_{02}	3.66	1.40	1.28	2.85
f_{06}	4.04	2.03	1.23	2.26
f_{07}	NA	1.44	1.34	4.32
f_{10}	3.98	1.47	1.21	2.90
f_{11}	3.93	1.44	1.23	2.71
f_{12}	4.57	1.41	1.23	2.60
f_{13}	4.76	1.49	1.25	2.78
f_{16}	NA	1.13	0.96	NA
F_{07}	NA	1.61	0.97	3.04
Avg	3.5714	1.3320	1.0610	2.6411

the functions compared with other DE variants. SaJADE also obtains the greatest overall successful rate, $\sum S_r = 17.48$ at $D = 30$ and $\sum S_r = 9.90$ at $D = 100$. Furthermore, Tables VII and IX show that SaJADE converges faster than its competitors, especially for functions at $D = 100$. For example, compared with JADE-wo in Table IX, the AR value is 1.3320, which indicates that SaJADE is on average 33.2% faster than JADE-wo for these functions. Although for the successful functions at $D = 100$ the AR value of SaJADE to JADE-w is 1.0610, however, for 8 out of 10 functions in Table IX the AR values are greater than 1.20, which means that SaJADE is about 20% faster than JADE-w for these functions. Additionally, Figure 1 shows that SaJADE is able to provide faster convergence speed than other DE variants on the majority of the functions.

4) Compared with Reported Results: A further comparative study of SaJADE to the reported results of recent advance EAs is also provided here. The results are shown in Table X. The results of the adaptive LEP and Best Lévy algorithms are obtained from Table III in [45], JADE-wo and JADE-w from Table 4.10 in [16], and jDE from Table III in [23]. The results in Table X indicate the superior performance of SaJADE in terms of the quality of the final solutions. SaJADE obtains the best results on 7 out of 9 functions. On the rest 2 functions, SaJADE provides the second best results.

E. Analysis of Strategy Adaptation

In our proposed SaJADE method, the strategy adaptation mechanism (SaM) is integrated into JADE to adaptively determine a more suitable strategy at different stages of evolution process for these different problems at hand. In order to investigate the adaptation characteristics of SaJADE, the evolution trend of the parameter μ_s is plotted on the selected function in Figure 2 with the mean curves and error bars. The error bars are the standard deviations of μ_s over 50 independent runs. They can clearly show the evolution trend of μ_s. For the clarity, there are only 20 error bars plotted for each figure. This is a sample average standard deviation over all runs. The evolutions of μ_{CR} and μ_F are also plotted in this figure. In addition, the error values of all function at $D = 30$ are shown in Table XI for JADE-wo, JADE-w, rJADE-wo, rJADE-w, and SaJADE. When several algorithms obtain the global optimum of a function, only the intermediate results are reported.

According to the results shown in Tables III, IV, and XI, from Figure 2 we can see that the μ_s can adaptively adjust with respect to the chosen problems, which means that our proposed SaM can choose a more suitable strategy for these different problems adaptively. For example, for function f_{01} at $D = 30$, JADE without archive is better than JADE with archive, hence the μ_s tends to lower values as shown in Figure 2 (a). However, when $D = 100$ for f_{01}, JADE-w shows better performance than JADE-wo, in this case μ_s obtains higher values in Figure 2 (b). Similar conclusions can also be drawn about μ_s on other functions from Figure 2. In addition, Figure 2 indicates that SaJADE is also able to maintain the adaptation of parameters μ_{CR} and μ_F as shown in JADE [15], [16]. Moreover, Table XI indicates that on the majority of the functions SaJADE obtains the best results compared with JADE methods. This is because of the cooperation among the different strategies in SaJADE.

By carefully looking at the results shown in Figure 2, we can observe that the μ_s values are most varied between 0.3 and 0.7. It means that Strategy 1 and Strategy 4 have seldom chances to be executed. For example, in Figure 1 (i) JADE with Strategy 4, i.e., rJADE-w, is able to provide the best result among four JADE methods, so ideally the μ_s values in Figure 2 (f) should be greater than 0.8, not around 0.6. With respect to this phenomenon, this may be because the four strategies used in our approach perform very similarly: All of them are likely to generate the promising offspring and lead to a *successful update*, i.e., the trial vector generated by the strategy is better than its target vector [16]. Thus, the μ_s values are almost around the initial value 0.5. However, if some of the strategies in the pool are very poor for a problem, the first SaM can still pursuit the better strategy, unrelated to the order of the strategies in the pool. In order to verify this expectation, we select four different strategies as the pool in Algorithm 2 to optimize function f_{01} at $D = 30$. The four strategies are: 1) "DE/best/1/bin", 2) "DE/rand/2/bin", 3) "DE/rand/3/bin", and 4) "DE/rand/4/bin". The initial values of μ_s are set to be 0.1, 0.5, and 0.9, respectively. All other parameters are kept unchanged as mentioned in Section IV-A. In addition, JADE with each of the above strategy is also tested on f_{01}. All results

TABLE VI

NFFEs REQUIRED TO OBTAIN ACCURACY LEVELS LESS THAN VTR FOR ALL FUNCTIONS AT $D = 30$. "NA" INDICATES THE ACCURACY LEVEL IS NOT OBTAINED AFTER MAX_NFFEs. THE SUCCESSFUL RATE S_r IS SHOWN IN THE PARENTHESES.

F	jDE	JADE-wo	JADE-w	SaDE	SaJADE
f_{01}	5.89E+04 ± 1.15E+03 (1.00)	*2.90E+04 ± 8.72E+02* (1.00)	3.03E+04 ± 8.54E+02 (1.00)	4.35E+04 ± 6.06E+02 (1.00)	**2.40E+04 ± 5.55E+02** (1.00)
f_{02}	8.12E+04 ± 1.27E+03 (1.00)	*5.08E+04 ± 2.49E+03* (1.00)	5.48E+04 ± 2.89E+03 (1.00)	7.19E+04 ± 9.36E+02 (1.00)	**3.90E+04 ± 1.44E+03** (1.00)
f_{03}	3.56E+05 ± 1.58E+04 (1.00)	1.02E+05 ± 4.60E+03 (1.00)	**7.78E+04 ± 3.88E+03** (1.00)	3.11E+05 ± 2.09E+04 (1.00)	*7.88E+04 ± 3.63E+03* (1.00)
f_{04}	2.93E+05 ± 1.39E+04 (1.00)	4.50E+05 ± 1.06E+04 (1.00)	3.08E+05 ± 5.18E+03 (1.00)	**1.68E+05 ± 4.82E+03** (1.00)	*2.09E+05 ± 8.25E+03* (1.00)
f_{05}	4.66E+05 ± 0.00E+00 (0.02)	1.53E+05 ± 5.50E+03 (0.88)	*1.22E+05 ± 5.43E+03* (0.96)	2.81E+05 ± 1.10E+04 (0.88)	**1.18E+05 ± 3.61E+03** (1.00)
f_{06}	2.22E+04 ± 8.48E+02 (1.00)	*1.09E+04 ± 4.14E+02* (1.00)	1.15E+04 ± 3.73E+02 (1.00)	1.58E+04 ± 4.66E+02 (1.00)	**9.20E+03 ± 2.25E+02** (1.00)
f_{07}	1.06E+05 ± 2.59E+04 (1.00)	*2.79E+04 ± 5.86E+03* (1.00)	2.99E+04 ± 7.48E+03 (1.00)	5.40E+04 ± 1.15E+04 (1.00)	**2.26E+04 ± 4.59E+03** (1.00)
f_{08}	**9.09E+04 ± 2.05E+03** (1.00)	1.21E+05 ± 1.91E+03 (1.00)	1.17E+05 ± 2.21E+03 (1.00)	*9.94E+04 ± 1.97E+03* (1.00)	1.01E+05 ± 3.98E+03 (1.00)
f_{09}	**1.17E+05 ± 3.84E+03** (1.00)	1.32E+05 ± 2.06E+03 (1.00)	1.43E+05 ± 1.93E+03 (1.00)	1.35E+05 ± 3.22E+03 (1.00)	*1.27E+05 ± 4.16E+03* (1.00)
f_{10}	8.95E+04 ± 1.50E+03 (1.00)	*4.54E+04 ± 1.17E+03* (1.00)	4.72E+04 ± 1.58E+03 (1.00)	6.85E+04 ± 8.06E+02 (1.00)	**3.61E+04 ± 8.47E+02** (1.00)
f_{11}	6.20E+04 ± 2.01E+03 (1.00)	*3.20E+04 ± 1.96E+03* (1.00)	3.44E+04 ± 5.12E+03 (1.00)	4.58E+04 ± 1.47E+03 (1.00)	**2.51E+04 ± 7.64E+02** (1.00)
f_{12}	5.34E+04 ± 1.30E+03 (1.00)	*2.74E+04 ± 1.11E+03* (1.00)	2.91E+04 ± 1.39E+03 (1.00)	3.89E+04 ± 7.64E+02 (1.00)	**2.17E+04 ± 7.32E+02** (1.00)
f_{13}	6.43E+04 ± 1.59E+03 (1.00)	*3.51E+04 ± 1.99E+03* (1.00)	3.76E+04 ± 3.26E+03 (1.00)	4.71E+04 ± 1.10E+03 (1.00)	**2.55E+04 ± 1.07E+03** (1.00)
f_{14}	NA ± NA (0.00)	NA ± NA (0.00)	**2.10E+05 ± 2.41E+04** (1.00)	NA ± NA (0.00)	*2.18E+05 ± 2.05E+04* (1.00)
f_{15}	NA ± NA (0.00)	NA ± NA (0.00)	NA ± NA (0.00)	NA ± NA (0.00)	NA ± NA (0.00)
f_{16}	*2.55E+05 ± 1.79E+04* (1.00)	NA ± NA (0.00)	NA ± NA (0.00)	2.84E+05 ± 0.00E+00 (0.02)	**1.51E+05 ± 7.01E+04** (0.72)
F_{06}	2.53E+05 ± 0.00E+00 (0.02)	1.33E+05 ± 1.11E+04 (0.74)	*1.09E+05 ± 6.15E+03* (0.84)	2.46E+05 ± 1.49E+04 (0.28)	**1.04E+05 ± 7.97E+03** (0.96)
F_{07}	1.01E+05 ± 1.66E+04 (0.66)	4.52E+04 ± 5.76E+03 (0.56)	*3.49E+04 ± 2.14E+04* (0.80)	1.02E+05 ± 1.44E+04 (0.62)	**3.29E+04 ± 4.02E+03** (0.80)
F_{09}	**8.34E+04 ± 3.69E+03** (1.00)	*9.60E+04 ± 1.85E+03* (1.00)	1.15E+05 ± 2.09E+03 (1.00)	1.10E+05 ± 2.76E+03 (1.00)	1.04E+05 ± 2.76E+03 (1.00)
F_{10}	NA ± NA (0.00)	NA ± NA (0.00)	NA ± NA (0.00)	NA ± NA (0.00)	NA ± NA (0.00)
$\sum S_r$	14.70	15.18	16.60	14.80	17.48

TABLE VIII

NFFEs REQUIRED TO OBTAIN ACCURACY LEVELS LESS THAN VTR FOR ALL FUNCTIONS AT $D = 100$. "NA" INDICATES THE ACCURACY LEVEL IS NOT OBTAINED AFTER MAX_NFFEs. THE SUCCESSFUL RATE S_r IS SHOWN IN THE PARENTHESES.

F	jDE	JADE-wo	JADE-w	SaDE	SaJADE
f_{01}	5.38E+05 ± 5.12E+03 (1.00)	1.96E+05 ± 3.46E+03 (1.00)	*1.64E+05 ± 2.61E+03* (1.00)	3.67E+05 ± 4.84E+03 (1.00)	**1.34E+05 ± 2.82E+03** (1.00)
f_{02}	7.45E+05 ± 5.74E+03 (1.00)	2.85E+05 ± 3.80E+03 (1.00)	*2.61E+05 ± 3.97E+03* (1.00)	5.80E+05 ± 5.09E+03 (1.00)	**2.03E+05 ± 2.21E+03** (1.00)
f_{06}	2.04E+05 ± 3.30E+03 (1.00)	1.03E+05 ± 8.58E+04 (1.00)	*6.23E+04 ± 4.68E+03* (1.00)	1.14E+05 ± 3.14E+03 (1.00)	**5.05E+04 ± 8.58E+02** (1.00)
f_{07}	NA ± NA (0.00)	2.09E+05 ± 3.09E+04 (1.00)	*1.94E+05 ± 2.66E+04* (1.00)	6.27E+05 ± 1.03E+05 (0.98)	**1.45E+05 ± 2.04E+04** (1.00)
f_{10}	7.82E+05 ± 5.45E+03 (1.00)	2.88E+05 ± 4.04E+03 (1.00)	*2.37E+05 ± 2.81E+03* (1.00)	5.69E+05 ± 6.21E+03 (1.00)	**1.97E+05 ± 3.77E+03** (1.00)
f_{11}	5.30E+05 ± 6.46E+03 (1.00)	1.95E+05 ± 6.33E+03 (0.92)	*1.66E+05 ± 6.39E+03* (1.00)	3.65E+05 ± 1.60E+04 (0.96)	**1.35E+05 ± 2.55E+03** (1.00)
f_{12}	5.11E+05 ± 6.17E+03 (1.00)	1.58E+05 ± 3.65E+03 (1.00)	*1.38E+05 ± 2.48E+03* (1.00)	2.91E+05 ± 3.64E+03 (1.00)	**1.12E+05 ± 1.94E+03** (1.00)
f_{13}	6.53E+05 ± 1.01E+04 (1.00)	2.05E+05 ± 1.56E+04 (1.00)	*1.71E+05 ± 4.11E+03* (1.00)	3.82E+05 ± 6.64E+03 (1.00)	**1.37E+05 ± 3.77E+03** (1.00)
f_{16}	NA ± NA (0.00)	3.92E+05 ± 1.55E+05 (1.00)	**3.34E+05 ± 4.89E+04** (0.86)	NA ± NA (0.00)	*3.47E+05 ± 1.03E+05* (1.00)
F_{07}	NA ± NA (0.00)	2.90E+05 ± 1.69E+04 (0.82)	**1.75E+05 ± 4.40E+03** (0.80)	5.47E+05 ± 2.43E+04 (0.78)	*1.80E+05 ± 4.08E+03* (0.88)
F_{08}	NA ± NA (0.00)	NA ± NA (0.00)	NA ± NA (0.00)	NA ± NA (0.00)	9.50E+05 ± 0.00E+00 (0.02)
F_{09}	9.98E+05 ± 0.00E+00 (0.02)	NA ± NA (0.00)	NA ± NA (0.00)	NA ± NA (0.00)	NA ± NA (0.00)
$\sum S_r$	7.02	9.74	9.66	8.72	9.90

TABLE X

COMPARATIVE STUDY OF SAJADE TO THE REPORTED RESULTS OF RECENT ADVANCE EAS. FOR EACH FUNCTION, THE FIRST COLUMN SHOWS THE MEAN VALUES; THE SECOND COLUMN IS THE STANDARD DEVIATION (IN PARENTHESIS).

F	Max_NFFEs	SaJADE	JADE-wo	JADE-w	jDE	Adaptive LEP	Best Lévy
f_{01}	150,000	**1.10E-79 (7.52E-79)**	*1.8E-60 (8.4E-60)*	1.3E-54 (9.2E-54)	1.1E-28 (1.0E-28)	6.32E-04 (7.6E-05)	6.59E-04 (6.4E-05)
f_{03}	150,000	*2.95E-20 (6.99E-20)*	2.8E-15 (8.2E-15)	**8.5E-22 (3.6E-21)**	0.090075 (0.080178)	0.041850 (0.059696)	30.628906 (22.113122)
f_{05}	150,000	**2.48E-15 (1.70E-14)**	3.2E-01 (1.1E+00)	5.6E-01 (1.4E+00)	*3.1E-15 (8.3E-15)*	43.40 (31.52)	57.75 (41.60)
f_{08}	150,000	**0.00E+00 (0.00E+00)**	4.7E+00 (2.3E+01)	2.4E+00 (1.7E+01)	**0.0E+00**[a] **(7.3E-12)**	1100.3[a] (58.2)	670.6[a] (52.2)
f_{09}	150,000	*8.87E-14 (2.51E-13)*	1.4E-11 (1.0E-11)	3.8E-11 (2.0-11)	**1.5E-15 (4.8E-15)**	5.85 (2.07)	12.50 2.29
f_{10}	150,000	**4.14E-15 (0.00E+00)**	**4.4E-15 (0.0E+00)**	**4.4E-15 (0.0E+00)**	7.7E-15 (1.4E-15)	1.9E-02 (1.0E-03)	3.1E-02 (2.0E-03)
f_{11}	150,000	**0.00E+00 (0.00E+00)**	**0.0E+00 (0.0E+00)**	2.0E-04 (1.4E-03)	**0.0E+00 (0.0E+00)**	2.4E-02 (2.8E-02)	1.8E-02 (1.7E-02)
f_{12}	150,000	**1.57E-32 (0.00E+00)**	**1.6E-32 (5.5E-48)**	**1.6E-32 (5.5E-48)**	6.6E-30 (7.9E-30)	6.0E-06 (1.0E-06)	3.0E-05 (4.0E-06)
f_{13}	150,000	**1.35E-32 (0.00E+00)**	**1.3E-32 (1.1E-47)**	**1.3E-32 (1.1E-47)**	5.0E-29 (1.4E-15)	9.8E-05 (1.2E-05)	2.6E-04 (3.0E-05)

[a] indicates the error value is used based on the reported results.

are averaged over 50 independent runs. The convergence graph and the evolution trend of μ_s are shown in Figure 3.

Figure 3 indicates that for different initial values of μ_s, the strategy parameter μ_s can still converge to around 0.15. It means that the first strategy "DE/best/1/bin" is always selected after some generations. The reason is that for the unimodal Sphere function f_{01}, the first strategy can provide the successful update all the time; while the other three strategies

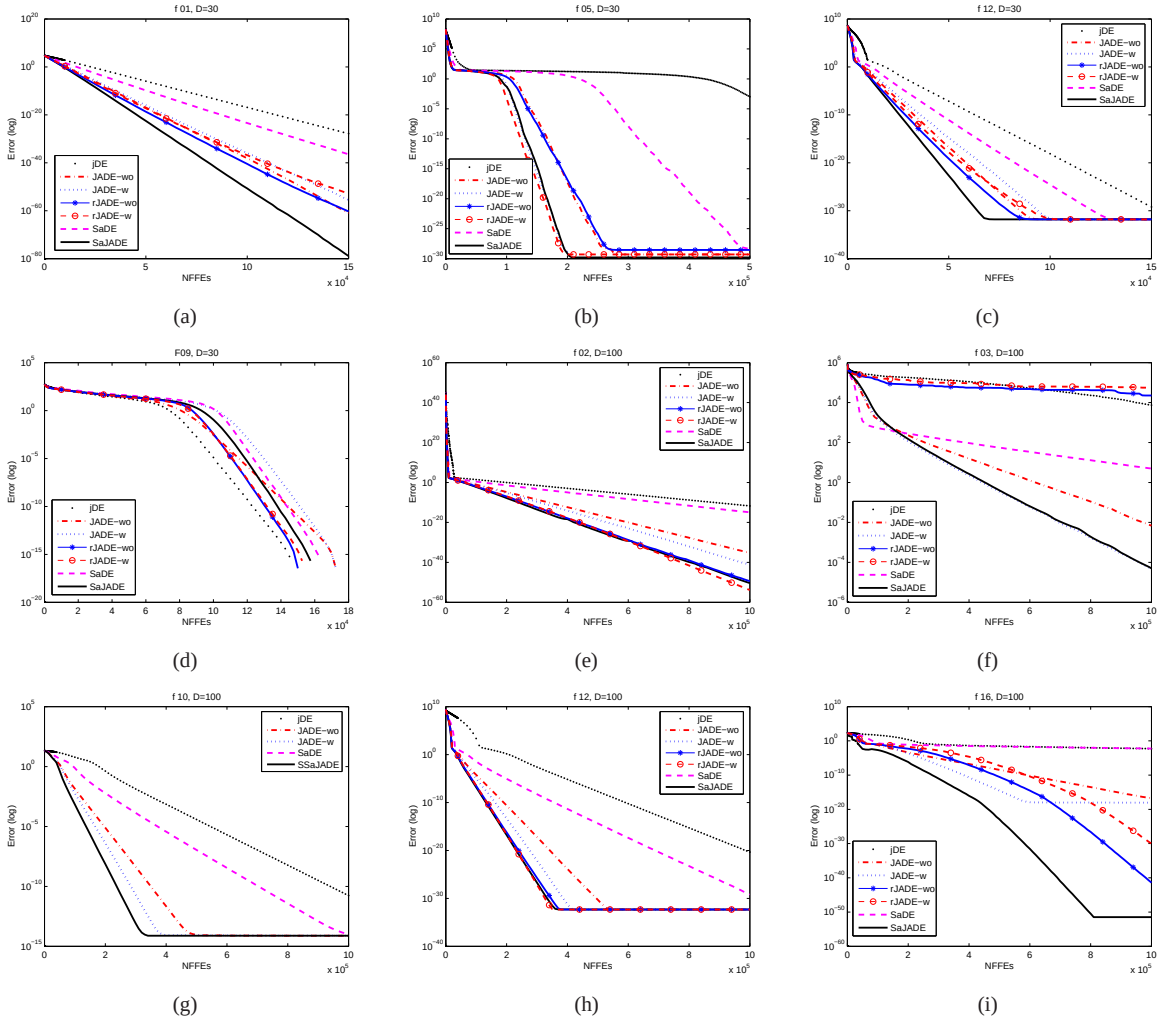


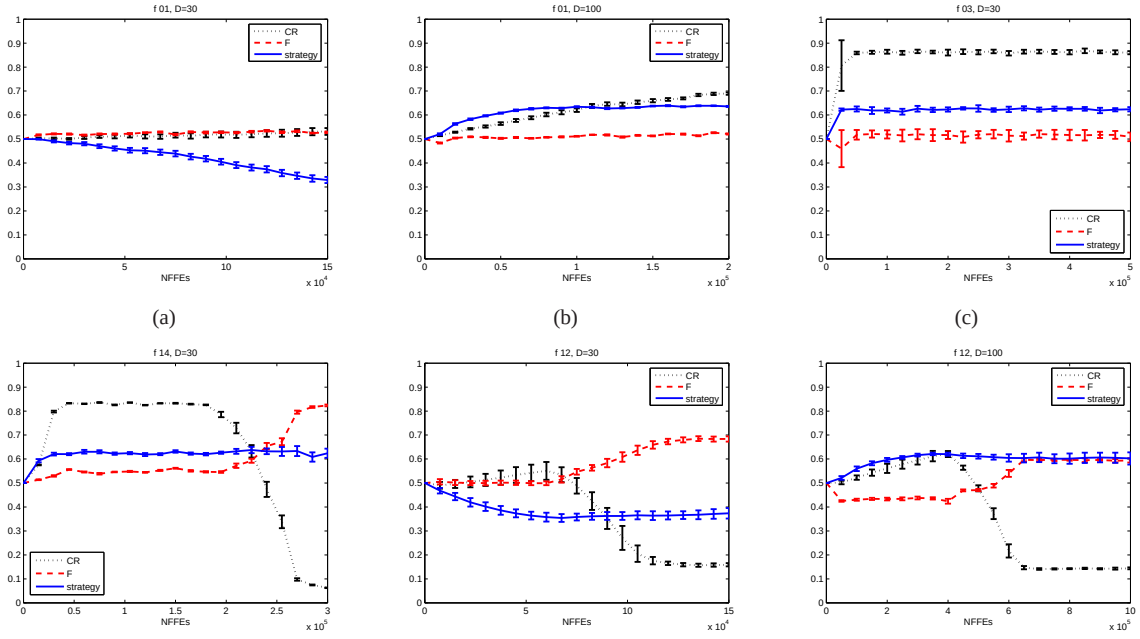
Fig. 1. Convergence graph of jDE, JADE-wo, JADE-w, rJADE-wo, rJADE-w, SaDE and SaJADE on the selected functions. (a) f_{01} ($D = 30$). (b) f_{05} ($D = 30$). (c) f_{12} ($D = 30$). (d) F_{09} ($D = 30$). (e) f_{02} ($D = 100$). (f) f_{03} ($D = 100$). (g) f_{10} ($D = 100$). (h) f_{12} ($D = 100$). (i) f_{16} ($D = 100$).

TABLE XI

ERROR VALUES AT $D = 30$ FOR ANALYSIS OF STRATEGY ADAPTATION OF SAJADE.

F	NFFEs	JADE-wo	JADE-w	rJADE-wo	rJADE-w	SaJADE
f_{01}	150,000	*9.93E-62 ± 5.34E-61*†	2.69E-56 ± 1.41E-55†	4.92E-61 ± 2.18E-60†	1.62E-53 ± 1.14E-52†	**1.10E-79 ± 7.52E-79**
f_{02}	200,000	5.53E-28 ± 3.16E-27†	3.18E-25 ± 2.05E-24†	*9.41E-31 ± 4.79E-30*†	7.55E-28 ± 3.25E-27†	**1.35E-47 ± 7.53E-47**
f_{03}	500,000	1.93E-56 ± 7.02E-56†	**6.11E-81 ± 1.62E-80**‡	6.18E-01 ± 2.14E+00†	1.71E+00 ± 4.04E+00†	*1.17E-77 ± 3.39E-77*
f_{04}	500,000	1.34E-09 ± 5.71E-10†	5.29E-14 ± 2.05E-14†	*6.33E-16 ± 3.25E-16*†	1.32E-15 ± 6.69E-16†	**1.26E-19 ± 1.35E-19**
f_{05}	500,000	4.78E-01 ± 1.31E+00†	1.59E-01 ± 7.89E-01	*2.85E-29 ± 6.09E-29*†	1.48E-01 ± 1.05E+00†	**1.60E-30 ± 6.32E-30**
f_{06}	10,000	3.12E+00 ± 1.54E+00†	5.62E+00 ± 1.87E+00†	*1.00E-01 ± 3.00E-01*†	1.18E+00 ± 1.03E+00†	**0.00E+00 ± 0.00E+00**
f_{07}	300,000	6.59E-04 ± 2.43E-04†	6.14E-04 ± 2.55E-04†	*4.58E-04 ± 2.04E-04*	4.90E-04 ± 1.98E-04†	**4.10E-04 ± 1.48E-04**
f_{08}	100,000	4.14E-05 ± 2.37E-05†	2.62E-04 ± 3.59E-04†	*2.50E-09 ± 4.13E-09*‡	**5.44E-09 ± 5.67E-09**‡	6.83E-07 ± 2.70E-06
f_{09}	100,000	**2.68E-03 ± 1.90E-03**‡	1.33E-01 ± 9.74E-02	*4.76E-03 ± 4.38E-03*‡	8.95E-03 ± 9.64E-03‡	1.54E-01 ± 2.25E-01
f_{10}	50,000	1.10E-09 ± 7.45E-10†	3.35E-09 ± 2.84E-09†	*1.38E-10 ± 2.44E-10*†	4.97E-10 ± 5.35E-10†	**1.12E-12 ± 1.07E-12**
f_{11}	50,000	1.44E-14 ± 7.05E-14†	1.57E-08 ± 1.09E-07†	*6.66E-18 ± 4.66E-17*†	7.55E-17 ± 5.28E-16†	**0.00E+00 ± 0.00E+00**
f_{12}	50,000	1.84E-17 ± 4.52E-17†	1.67E-15 ± 1.02E-14†	*7.15E-20 ± 1.98E-19*†	1.60E-18 ± 4.85E-18†	**2.10E-23 ± 6.89E-23**
f_{13}	50,000	3.16E-13 ± 9.79E-13†	1.87E-10 ± 1.09E-09†	*2.36E-17 ± 5.45E-17*†	2.55E-15 ± 9.71E-15†	**3.83E-21 ± 1.56E-20**
f_{14}	300,000	1.72E-03 ± 3.05E-03†	**1.68E-09 ± 1.97E-09**‡	1.63E-02 ± 7.01E-02†	8.15E-09 ± 1.74E-09†	*2.88E-09 ± 2.43E-09*
f_{15}	300,000	2.02E-01 ± 1.41E-02†	2.00E-01 ± 1.63E-12†	**1.58E-01 ± 4.94E-02**	2.52E-01 ± 5.04E-02†	*1.76E-01 ± 4.28E-02*
f_{16}	300,000	2.61E-06 ± 1.21E-06†	2.78E-05 ± 8.43E-06†	*1.53E-07 ± 5.24E-07*	1.57E-07 ± 4.00E-07	**1.44E-07 ± 4.92E-07**
F_{06}	300,000	7.00E+00 ± 1.87E+01†	2.56E+00 ± 6.22E+00	*2.21E+00 ± 1.12E+01*†	3.49E+00 ± 1.53E+01†	**1.59E-01 ± 7.89E-01**
F_{07}	300,000	1.57E-02 ± 1.13E-02†	*5.96E-03 ± 7.39E-03*‡	1.29E-02 ± 1.02E-02	**4.83E-03 ± 6.19E-03**‡	1.04E-02 ± 8.48E-03
F_{09}	100,000	**2.87E-03 ± 1.82E-03**‡	1.35E+00 ± 6.08E-01†	3.69E-03 ± 3.61E-03‡	*3.55E-03 ± 4.98E-03*‡	1.13E-01 ± 1.60E-01
F_{10}	300,000	2.88E+01 ± 5.33E+00†	*2.82E+01 ± 5.32E+00*	4.01E+01 ± 2.23E+01†	4.25E+01 ± 2.10E+01†	**2.66E+01 ± 4.44E+00**
$w/t/l$		18/0/2	13/4/3	13/4/3	15/1/4	–

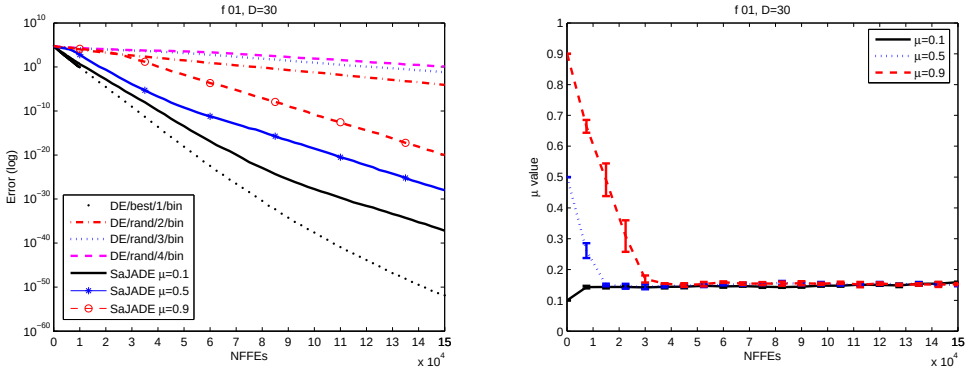
† indicates SaJADE is significantly better than its competitor by the Wilcoxon signed-rand test at $\alpha = 0.05$.
‡ means that the corresponding algorithm is better than our proposed SaJADE method.



(a) (b) (c)

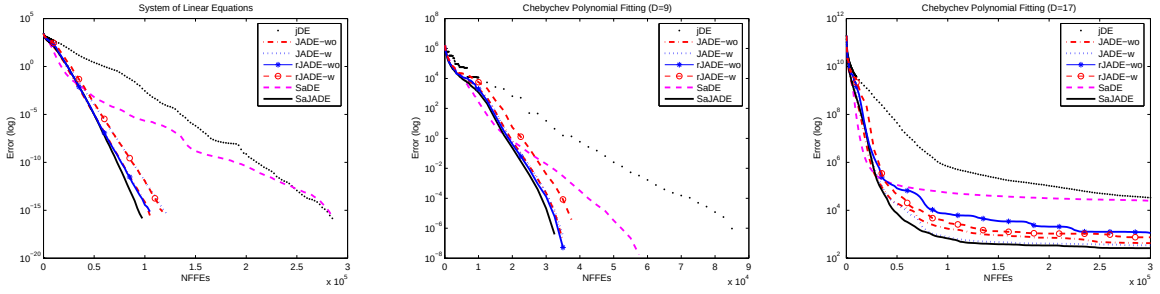
(d) (e) (f)

Fig. 2. Adaptation characteristics of μ_{CR}, μ_F, and μ_s on the selected functions. (a) f_{01} ($D = 30$). (b) f_{01} ($D = 100$). (c) f_{03} ($D = 30$). (d) f_{14} ($D = 30$). (e) f_{12} ($D = 30$). (f) f_{12} ($D = 100$).



(a) (b)

Fig. 3. Analysis of the adaptation characteristics of μ_s with different initial values on the function f_{01} at $D = 30$. (a) Convergence graph of JADE with different strategies and SaJADE with different initial values of μ_s. (b) The evolution trend of μ_s in SaJADE with different initial values.



(a) (b) (c)

Fig. 4. Convergence graph of jDE, JADE-wo, JADE-w, rJADE-wo, rJADE-w, SaDE and SaJADE on the real-world problems. (a) System of linear equations. (b) Chebychev polynomial fitness problem at $D = 9$. (c) Chebychev polynomial fitness problem at $D = 17$.

TABLE XII

MEAN AND STANDARD DEVIATION OF THE FINAL VALUES OF THE BEST-SO-FAR SOLUTIONS OVER 50 INDEPENDENT RUNS FOR THE LINEAR EQUATIONS PROBLEM AND THE CHEBYCHEV POLYNOMIAL FITTING PROBLEM.

Prob	Max_NFFEs	jDE	JADE-wo	JADE-w	SaDE	SSaJADE
LES	300,000	1.53E+05 ± 1.90E+04	*6.54E+04 ± 2.15E+03*	7.40E+04 ± 2.30E+03	8.75E+04 ± 3.28E+04	**6.09E+04 ± 3.24E+03**
CPF($D = 9$)	100,000	7.33E+04 ± 9.08E+03	*3.29E+04 ± 1.49E+03*	3.33E+04 ± 1.35E+03	4.57E+04 ± 4.41E+03	**3.10E+04 ± 1.22E+03**
CPF($D = 17$)*	300,000	3.34E+04 ± 4.50E+04	4.24E+02 ± 6.36E+02	*3.44E+02 ± 2.77E+02*	2.50E+04 ± 3.82E+04	**2.61E+02 ± 3.32E+02**

* indicates that the error values of the final solutions are used, since no algorithm can solve the corresponding problem within the Max_NFFEs.

perform badly (as shown in Figure 3 (a)) because of the high diversity generated by these strategies. From this experiment, we can see that the first SaM is still able to adaptively determine a more suitable strategy for the problem at hand. The influence of the order of strategies is not very significant on the performance of our approach. However, Figure 3 (b) indicates that if the initial value of μ_s is far away from the best strategy, the first SaM might need a relative long run to pursuit the best strategy, and hence, deteriorates the performance. The reason might be that when we update the mean value of H_s in Eqn. (24), we only consider the successful strategy parameter η_i. We don't consider the fitness improvement of the successful strategy, which may also affect the performance of our approach. We leave this work in our future work.

F. Analysis of the Simplicity

In [35], Ong and Keane stated that the simplicity of an algorithm is also very important. The simplicity means ease of implementation and a minimum numbers of control parameters of the algorithm [35]. As described in Section III-B, we can see that our proposed three SaMs are all very simple and easy to be implemented. They don't increase the complexity of the original DE algorithm. Compared with JADE [15], the first SaM introduce one additional parameter μ_s, this parameter is insensitive for different problems[6]. Similar to the parameter adaptation in jDE [23], in the second SaM, the parameter δ is easily set as the τ_1 and τ_2.

G. Comparison on Two Real-World Problems

In this section, two real-world problems are used to evaluate the capability of solving the real-world problems of our approach. The two problems are the systems of linear equations problem (LES, for short) [46] and the Chebychev polynomial fitting problem (CPF, for short) [3]. The LES problem is defined at $D = 10$. The CPF problem is defined at $D = 9$ and $D = 17$. Both of the two problems have the minimal values of 0. Since we don't make any modifications of these problems, we omit to describe them. More details can be found in [46] and [3]. These two real-world problems have been widely used in the EA literature, for example, in [47], [48], and [49].

We use SaJADE, jDE, JADE-wo, JADE-w, and SaDE to optimize the two problems. Note that in this experiment our purpose is only to test the potential of solving the real-world problems of SaJADE. Table XII shows the mean and standard deviation of the final results of the best-so-far solutions over 50 runs. The Max_NFFEs for each problem are shown in column 2 of this table. Since for the LES problem and the CPF problem at $D = 9$ all of the algorithms can obtain the optimal value, in Table XII the NFFEs are reported in the second and third columns. The error values of CPF at $D = 17$ are reported in the fourth column. From Table XII, it is clear that SaJADE provides consistently the best results among the five DE variants. Additionally, some representative convergence graphs are shown in Figure 4. The convergence curves of rJADE-wo and rJADE-w are also plotted in Figure 4. It can be seen that SaJADE obtains the fastest convergence speed compared with other DE variants.

[6]Experiments on the effect of the initial values of μ_s are not reported due to the tight space limitation. Interested readers can contact the authors for more details.

H. Discussion

The DE algorithm is an efficient and versatile population-based, direct search algorithm for global optimization, which has been widely used in many scientific fields. In the original DE algorithm and many DE variants, there are many mutation strategies available. However, the choice of the best mutation strategy is difficult for a specific problem. In the previous DE variants, the study on the adaptive strategies of DE is scarce [12], [13], and [11]. Due to these considerations, in this work, we propose a family of improved DE variants, which use several simple methods to implement the strategy adaptation. Experiments have been conducted on 20 benchmark problems and two real-world problems. From the experimental results we can summarize that:

1) Among three different strategy adaptation methods, our proposed first method combined with JADE obtains the overall best performance as shown in Section IV-C. Moreover, all of the strategy adaptation methods is able to provide the better results than the uniform strategy selection method.
2) According to the experimental results shown in Section IV-E, we can see that the first SaM is able to adaptively select a more suitable strategy for a specific problem. In addition, all of the proposed SaMs are very simple and ease of implementation.
3) The proposed SaJADE method (JADE combined with our proposed first SaM) provides better, or highly competitive, results compared with other DE variants considered in this work not only for the benchmark problems, but for two real-world problems. Moreover, SaJADE is able to enhance the performance of JADE in terms of the quality of the final solutions and the convergence speed.
4) Although SaJADE shows good overall performance, it maybe still be trapped in the local optima occasionally,

e.g., for f_{16} and F_{06} at $D = 30$. The reason might be the rapid decrease of the population diversity. The possible improvement of SaJADE is the population restart method as proposed in [50] or [51]. However, this is beyond the scope of this work, we leave it in our future work.

V. Conclusions and Future Work

In this paper, we describe a family of improved DE variants, where two simple strategy adaptation mechanisms (SaMs) have been implemented to adaptively select a more suitable strategy with respect to the chosen problems. In the proposed SaM, a strategy parameter η is used control the selection of different strategies. $\eta \in [0, 1)$ is a real number, thus there are possible many methods that can be used to update it. In this work, two methods inspired by the ideas from the parameter adaptation in the DE literature are presented to update the parameter η. Our proposed SaMs are combined with JADE, which is a recently proposed DE variants, for the numerical optimization. Experiments have been conducted on 20 benchmark problems and two real-world problems. The results verify our expectation that SaM is able to adaptively determine a more suitable strategy for a specific problem. Compared with other state-of-the-art DE variants, our approach performs better, or highly comparably, in terms of the quality of the final solutions and the convergence rate. Furthermore, SaJADE is able to enhance the performance of JADE.

Possible direction for the future work includes adopting the population restart method to further improve the reliability of SaJADE. In addition, we will also investigate other parameter adaptive methods of EAs, e.g., the adaptation methods proposed in evolution strategy, to handle the strategy parameter.

Acknowledgment

The authors thank Prof. Y.-S. Ong for his constructive comments on this paper. The first author would also like to thank Prof. J. Brest and Prof. P. N. Suganthan for providing the jDE and SaDE source codes, respectively. They are also grateful to the associate editor and three anonymous reviewers for their valuable comments and suggestions on this paper.

References

[1] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford, UK: Oxford University Press, 1996.

[2] R. Storn and K. Price, "Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces," *J. of Global Optim.*, vol. 11, no. 4, pp. 341–359, Dec 1997.

[3] K. Price, R. Storn, and J. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Berlin: Springer-Verlag, 2005.

[4] R. Storn and K. Price, "Home page of differential evolution," 2008. [Online]. Available: http://www.ICSI.Berkeley.edu/~storn/code.html

[5] B. Alatas, E. Akin, and A. Karci, "MODENAR: Multi-objective differential evolution algorithm for mining numeric association rules," *Appl. Soft Comput.*, vol. 8, no. 1, pp. 646–656, Jan. 2008.

[6] S. Das, A. Abraham, and A. Konar, "Automatic clustering using an improved differential evolution algorithm," *IEEE Trans. on Syst. Man, Cybern. A, Syst., Humans*, vol. 38, no. 1, pp. 218–237, Feb 2008.

[7] V. Feoktistov, *Differential Evolution: In Search of Solutions*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[8] U. Chakraborty, *Advances in Differential Evolution*. Berlin: Springer-Verlag, 2008.

[9] G. C. Onwubolu and D. Davendra, *Differential Evolution: A Handbook for Global Permutation-Based Combinatorial Optimization*. Berlin: Springer-Verlag, 2009.

[10] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *IEEE Congr. on Evol. Comput. (CEC2005)*. IEEE, 2005, pp. 1785–1791.

[11] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. on Evol. Comput.*, vol. 13, no. 2, pp. 398–417, Apr 2009.

[12] X.-F. Xie and W.-J. Zhang, "SWAF: Swarm algorithm framework for numerical optimization," in *Proc. Genetic Evol. Comput. Conf., GECCO 2004, Part I*, ser. LNCS, vol. 3102, 2004, pp. 238–250.

[13] A. Zamuda, J. Brest, B. Bošković, and V. Žumer, "Large scale global optimization using differential evolution with self-adaptation and cooperative co-evolution," in *2008 IEEE World Congr. on Comput. Intell.* IEEE Press, 2008, pp. 3719–3726.

[14] J. Brest, B. Bošković, S. Greiner, V. Žumer, and M. S. Maučec, "Performance comparison of self-adaptive and adaptive differential evolution algorithms," *Soft Comput.*, vol. 11, no. 7, pp. 617–629, May 2007.

[15] J. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Trans. on Evol. Comput.*, no. 5, pp. 945–958, Oct 2009.

[16] ——, *Adaptive Differential Evolution: A Robust Approach to Multimodal Problem Optimization*. Berlin: Springer-Verlag, 2009.

[17] W. Zhong, J. Liu, M. Xue, and L. Jiao, "A multiagent genetic algorithm for global numerical optimization," *IEEE Trans. on Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 2, pp. 1128–1141, 2004.

[18] X. Yao and Y. Liu, "Fast evolution strategies," *Control and Cybern.*, vol. 26, no. 3, pp. 467–496, 1997.

[19] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Trans. on Evol. Comput.*, vol. 3, no. 2, pp. 82–102, Jul 1999.

[20] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. on Evol. Comput.*, vol. 10, no. 3, pp. 281–295, 2006.

[21] L. Jiao, Y. Li, M. Gong, and X. Zhang, "Quantum-inspired immune clonal algorithm for global optimization," *IEEE Trans. on Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 5, pp. 1234–1253, 2008.

[22] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Trans. on Evol. Comput.*, vol. 13, no. 3, pp. 526–553, 2009.

[23] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Trans. on Evol. Comput.*, vol. 10, no. 6, pp. 646–657, Dec 2006.

[24] N. Noman and H. Iba, "Accelerating differential evolution using an adaptive local search," *IEEE Trans. on Evol. Comput.*, vol. 12, no. 1, pp. 107–125, Feb 2008.

[25] S. Rahnamayan, H. Tizhoosh, and M. Salama, "Opposition-based differential evolution," *IEEE Trans. on Evol. Comput.*, vol. 12, no. 1, pp. 64–79, Feb 2008.

[26] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. C. Coello, "A comparative study of differential evolution variants for global optimization," in *Proc. Genetic Evol. Comput. Conf., GECCO 2006*, M. Cattolico, Ed., July 8-12, 2006, pp. 485–492.

[27] R. Storn, "Differential evolution design of an IIR-filter with requirements for magnitude and group delay," in *in IEEE International Conf. on Evol. Comput. ICEC96*. IEEE Press, 1996, pp. 268–273.

[28] B. V. Babu and S. A. Munawar, "Optimal design of shell-and-tube heat exchangers by different strategies of differential evolution," 2001. [Online]. Available: http://discovery.bits-pilani.ac.in/ bvbabu/HEDE.pdf

[29] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. on Evol. Comput.*, vol. 1, no. 1, pp. 67–82, 1997.

[30] V. Feoktistov and S. Janaqi, "Generalization of the strategies in differential evolution," in *18th International Parallel and Distributed Processing Symposium (IPDPS'04) - Workshop 6*, vol. 7. Los Alamitos, CA, USA: IEEE Computer Society, 2004, p. 165a.

[31] A. W. Iorio and X. Li, "Solving rotated multi-objective optimization problems using differential evolution," in *AI 2004: Advances in Artificial Intelligence, Proceedings*. Springer-Verlag, LNAI Vol. 3339, 2004, pp. 861–872.

[32] M. M. Ali and L. P. Fatti, "A differential free point generation scheme in the differential evolution algorithm," *J. of Global Optim.*, vol. 35, no. 4, pp. 551–572, 2006.

[33] J. E. Smith and T. C. Fogarty, "Operator and parameter adaptation in genetic algorithms," *Soft Comput.*, vol. 1, no. 2, pp. 81–87, 1997.

[34] Á. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Trans. on Evol. Comput.*, vol. 3, no. 2, pp. 124–141, Jul 1999.
[35] Y.-S. Ong and A. J. Keane, "Meta-Lamarckian learning in memetic algorithms," *IEEE Trans. on Evol. Comput.*, vol. 8, no. 2, pp. 99–110, Apr 2004.
[36] J. Vrugt, B. Robinson, and J. Hyman, "Self-adaptive multimethod search for global optimization in real-parameter spaces," *IEEE Trans. on Evol. Comput.*, vol. 13, no. 2, pp. 243–259, Apr. 2009.
[37] M. M. Ali, C. Khompatraporn, and Z. B. Zabinsky, "A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems," *J. of Global Optim.*, vol. 31, no. 4, pp. 635–672, 2005.
[38] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the CEC2005 special session on real-parameter optimization," 2005. [Online]. Available: http://www.ntu.edu.sg/home/EPNSugan
[39] Y.-W. Shang and Y.-H. Qiu, "A note on the extended rosenbrock function," *Evol. Comput.*, vol. 14, no. 1, pp. 119–126, 2006.
[40] C. Shaw, K. Williams, and R. Assassa, "Patients' views of a new nurse-led continence service," *Journal of Clinical Nursing*, vol. 9, no. 4, pp. 574–584, 2003.
[41] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *Journal Of Machine Learning Research*, vol. 7, pp. 1–30, 2006.
[42] S. García and F. Herrera, "An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons," *Journal Of Machine Learning Research*, vol. 9, pp. 2677–2694, 2008.
[43] S. García, A. Fernandez, J. Luengo, and F. Herrera, "A study of statistical techniques and performance measures for genetics-based machine learning: Accuracy and interpretability," *Soft Comput.*, vol. 13, no. 10, pp. 959–977, 2009.
[44] S. García, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC'2005 special session on real parameter optimization," *Journal of Heuristics, (in press)*, 2009.
[45] C.-Y. Lee and X. Yao, "Evolutionary programming using mutations based on the Lévy probability distribution," *IEEE Trans. on Evol. Comput.*, vol. 8, no. 1, pp. 1–13, 2004.
[46] F. Herrera and M. Lozano, "Gradual distributed real-coded genetic algorithms," *IEEE Trans. on Evol. Comput.*, vol. 4, no. 1, pp. 43–63, 2000.
[47] F. Herrera, M. Lozano, and A. M. Sánchez, "Hybrid crossover operators for real-coded genetic algorithms: an experimental study," *Soft Comput.*, vol. 9, no. 4, pp. 280–298, 2005.
[48] C. García-Martínez, M. Lozano, F. Herrera, D. Molina, and A. M. Sánchez, "Global and local real-coded genetic algorithms based on parent-centric crossover operators," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1088–1113, 2008.
[49] M. Lozano, F. Herrera, and J. R. Cano, "Replacement strategies to preserve useful diversity in steady-state genetic algorithms," *Inf. Sci.*, vol. 178, no. 23, pp. 4421–4433, 2008.
[50] A. Auger and N. Hansen, "A restart CMA evolution strategy with increasing population size," in *IEEE Congr. on Evol. Comput. (CEC2005)*, vol. 2. IEEE, 2005, pp. 1769–1776.
[51] F. Peng, K. Tang, G. Chen, and X. Yao, "Multi-start JADE with knowledge transfer for numerical optimization," in *IEEE Congr. on Evol. Comput. (CEC2009)*. IEEE, 2009, pp. 1889–1895.

Wenyin Gong received the B.Eng., M.Eng., and PhD degrees in computer science from China University of Geosciences, Wuhan, China, in 2004, 2007, and 2010, respectively. He has published over 20 research papers in journals and international conferences. He is interested in differential evolution, memetic algorithms, multiobjective optimization, and their applications.

Zhihua Cai received the Bsc degree from Wuhan University, China, in 1986, the Msc degree from Beijing University of Technology, China, in 1992, and the PhD degree from China University of Geosciences, in 2003. He is currently a faculty member at School of Computer Science, China University of Geosciences, Wuhan, China. His main research areas include data mining, machine learning, evolutionary computation, and their applications.

Charles Ling received the MSc and PhD degrees from the Department of Computer Science at the University of Pennsylvania in 1987 and 1989, respectively. Since then, he has been a faculty member of computer science at the University of Western Ontario, Canada. His main research areas include machine learning (theory, algorithms, and applications), cognitive modeling, and AI in general. He has published more than 120 research papers in journals (such as Machine Learning, JMLR, JAIR, TKDE, and Cognition) and international conferences (such as IJCAI, ICML, and ICDM). He has been an associate editor for the IEEE Transactions on Knowledge and Data Engineering, and guest editor for several journals. He is also the director of Data Mining Lab, leading data mining development in CRM, bioinformatics, and the Internet. He has managed several data mining projects for major banks and insurance companies in Canada.

Hui Li received the PhD degree in management engineering from Huazhong University of Science and Technology, Wuhan, China, in 2008. Currently, she is a faculty member at School of Computer Science, China University of Geosciences, Wuhan, China. Her research areas include operational research and evolutionary algorithm.