JADE: Adaptive Differential Evolution with Optional External Archive

Jingqiao Zhang, *Student Member, IEEE,* and Arthur C. Sanderson, *Fellow, IEEE,*

***Abstract*— A new differential evolution (DE) algorithm, JADE, is proposed to improve optimization performance by implementing a new mutation strategy "DE/current-to-$p$best" with optional external archive and updating control parameters in an adaptive manner. The DE/current-to-$p$best is a generalization of the classic "DE/current-to-best," while the optional archive operation utilizes historical data to provide information of progress direction. Both operations diversify the population and improve the convergence performance. The parameter adaptation automatically updates the control parameters to appropriate values and avoids a user's prior knowledge of the relationship between the parameter settings and the characteristics of optimization problems. It is thus helpful to improve the robustness of the algorithm. Simulation results show that JADE is better than, or at least comparable to, other classic or adaptive DE algorithms, the canonical particle swarm optimization, and other evolutionary algorithms from the literature in terms of convergence performance for a set of 20 benchmark problems. JADE with an external archive shows promising results for relatively high dimensional problems. In addition, it clearly shows that there is no fixed control parameter setting suitable for various problems or even at different optimization stages of a single problem.**

***Index Terms*— Adaptive parameter control, differential evolution, evolutionary optimization, external archive.**

I. Introduction

DIFFERENTIAL evolution (DE) has been shown to be a simple yet efficient evolutionary algorithm for many optimization problems in real-world applications [1]–[5]. Its performance, however, is still quite dependent on the setting of control parameters such as the mutation factor and the crossover probability according to both experimental studies [6] and theoretical analyses [7]. Although there are already suggestions for parameter settings [2], [6], [8], the interaction between the parameter setting and the optimization performance is still complicated and not completely understood. This is mainly because there is no fixed parameter setting that is suitable for various problems or even at different evolution stages of a single problem.

The trial-and-error method for tuning the control parameters usually requires tedious optimization trials, even for an algorithm (e.g., the classic DE/rand/1/bin) which fixes its parameters throughout the evolutionary search. Motivated by this consideration, different adaptive or self-adaptive mechanisms [9]–[18] have been recently introduced to dynamically update the control parameters without a user's prior knowledge of the relationship between the parameter setting and the characteristics of optimization problems. In addition, the parameter adaptation, if well designed, is capable of improving an algorithm's convergence performance.

Parameter adaptation mechanisms proposed in the aforementioned references can be categorized based on how the control parameters are changed. According to the classification scheme introduced by Angeline [19] and Eiben *et al.* [20], [21], we can interpret three classes of parameter control mechanisms as follows.

1) *Deterministic Parameter Control*: The control parameter is altered by some deterministic rules without taking into account any feedback from the evolutionary search. One example is the time-dependent change of the mutation rates proposed by Holland [22].
2) *Adaptive Parameter Control*: Feedback from the evolutionary search is used to dynamically change the control parameters. Examples of this class are Rechenberg's "1/5-th rule" [23] and the fuzzy-logic adaptive DE in [11] and [12]. Several newly proposed DE algorithms, SaDE [13], jDE [15], and SaNSDE [18], and the algorithm proposed in this paper can be also classified into this category.[1]
3) *Self-adaptive Parameter Control*: A method of "the evolution of evolution" is used to conduct the self-adaptation of control parameters. The control parameters are directly associated with individuals and undergo mutation and recombination/crossover. Since better parameter values tend to generate individuals that are more likely to survive, these values can be propagated to more offspring. The SPDE in [9] for multiobjective optimization (MOO) and the DESAP in [10] belong to this category.

Adaptive or self-adaptive parameter control, if well designed, can enhance the robustness of an algorithm by dynamically adapting the parameters to the characteristic of different fitness

Manuscript received November 12, 2007; revised January 27, 2008 and March 30, 2008; accepted May 8, 2008. Current version published September 30, 2009. Funding for this work was provided in part by Grant Number IIS-0329837 from the National Science Foundation. This work is also supported in part by the Center for Automation Technologies and Systems (CATS) under a block grant from the New York State Office of Science, Technology, and Academic Research (NYSTAR).

The authors are with the Department of Electrical, Computer and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180 USA (e-mail: zhangj14@rpi.edu; sandea@rpi.edu).

Digital Object Identifier 10.1109/TEVC.2009.2014613

[1]Some of these algorithms are classified as "self-adaptive" in the original papers. However, although each individual in the population is associated with its own control parameters, these parameters themselves do not undergo mutation and crossover in the optimization process. Thus, according to the definition of Eiben *et al.*, these algorithms can be considered as adaptive schemes.

landscapes. It is thus applicable to various optimization problems without trial and error. In addition, the convergence rate can be improved if the control parameters are adapted to appropriate values at different evolution stages of a specific problem.

The adaptive and self-adaptive DE algorithms have shown faster and more reliable convergence performance than the classic DE algorithms without parameter control for many benchmark problems [13]–[17]. It is of interest to find the most suitable DE variant and use it as the underlying scheme where parameter adaptation operations can be introduced. Some adaptive algorithms [10], [14], [18] are developed based on the classic DE/rand/1/bin that is known to be robust but less efficient in terms of convergence rate. Others [13], [14], [16] simultaneously implement DE/rand/1/bin and one or more greedy DE variants such as DE/current-to-best/1/bin and dynamically update the probability of using each variant to generate offspring.

So far, there has been no method developed solely based on a greedy DE variant (such as DE/current-to-best/1/bin and DE/best/1/bin) that utilizes the information of the best solution(s) in the current population. The reason seems straightforward: a greedy variant is usually less reliable and may lead to problems such as premature convergence, especially when solving multimodal problems [8], [24]. However, we note that a well-designed parameter adaptation scheme is usually beneficial to enhance the robustness of an algorithm. Hence, in an adaptive DE algorithm, the reliability of greedy DE variants becomes a less crucial problem, while their fast convergence becomes more attractive.

Other than the best solutions in the current population, historical data is another source that can be used to improve the convergence performance. One example is the particle swarm optimization (PSO) [25], where the best solutions explored in the history are used to direct the movement of the current population (swarm). In this paper, instead of the *best* solutions previously explored, we are interested in a set of recently explored *inferior* solutions and consider their difference from the current population as a promising direction toward the optimum. These solutions are also utilized to improve the population diversity.

In view of the above considerations, we introduce a new greedy mutation strategy, "DE/current-to-$p$best" with optional external archive, and use it as the basis of our parameter adaptive algorithm JADE. An earlier version of the JADE algorithm was presented in [26]. As a generalization of DE/current-to-best, DE/current-to-$p$best utilizes not only the best solution information but also the information of other good solutions. To be specific, any of the top $100p\%$, $p \in (0, 1]$, solutions can be randomly chosen in DE/current-to-$p$best to play the role designed exclusively for the single best solution in DE/current-to-best. In addition, the difference between the archived inferior solutions and the current population can be incorporated into the mutation operation. In spite of its greedy property, the proposed strategy is able to diversify the population so that the problems such as premature convergence can be alleviated. The reliability of the algorithm is further improved by the adaptive parameter control.

The rest of the paper is organized as follows. Section II describes the basic procedure of differential evolution and introduces notations and terminologies that are useful for the review of various adaptive or self-adaptive DE algorithms in Section III. The new algorithm, JADE, is elaborated in Section IV, with detailed explanations on the DE/current-to-$p$best with optional archive and the adaptive parameter control mechanism. Simulation results are presented in Section V for the comparison of JADE with other evolutionary algorithms. Finally, concluding remarks are summarized in Section VI.

II. Basic Operations of DE

In this section, we describe the basic operations of differential evolution and introduce necessary notations and terminologies which facilitate the explanation of different adaptive DE algorithms later.

Differential evolution follows the general procedure of an evolutionary algorithm. The initial population $\{\mathbf{x}_{i,0} = (x_{1,i,0}, x_{2,i,0}, \ldots, x_{D,i,0}) | i = 1, 2, \ldots, NP\}$ is randomly generated according to a uniform distribution $x_j^{\text{low}} \le x_{j,i,0} \le x_j^{\text{up}}$, for $j = 1, 2, \ldots, D$, where D is the dimension of the problem and NP is the population size. After initialization, DE enters a loop of evolutionary operations: mutation, crossover, and selection.

Mutation: At each generation g, this operation creates mutation vectors $\mathbf{v}_{i,g}$ based on the current parent population $\{\mathbf{x}_{i,g} | i = 1, 2, \ldots, NP\}$. The following are different mutation strategies frequently used in the literature:

1) "DE/rand/1"

$$\mathbf{v}_{i,g} = \mathbf{x}_{r0,g} + F_i \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) \tag{1}$$

2) "DE/current-to-best/1"

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F_i \cdot (\mathbf{x}_{\text{best},g} - \mathbf{x}_{i,g}) + F_i \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) \tag{2}$$

3) "DE/best/1"

$$\mathbf{v}_{i,g} = \mathbf{x}_{\text{best},g} + F_i \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) \tag{3}$$

where the indices $r0$, $r1$ and $r2$ are distinct integers uniformly chosen from the set $\{1, 2, \ldots, NP\} \backslash \{i\}$, $\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}$ is a difference vector to mutate the corresponding parent $\mathbf{x}_{i,g}$, $\mathbf{x}_{\text{best},g}$ is the best vector at the current generation g, and F_i is the mutation factor which usually ranges on the interval $(0, 1+)$. In classic DE, $F_i = F$ is a fixed parameter used to generate all mutation vectors at all generations, while in many adaptive DE algorithms each individual i is associated with its own mutation factor F_i.

The above mutation strategies can be generalized by implementing multiple difference vectors other than $\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}$. The resulting strategy is named as "DE/–/k" depending on the number k of difference vectors adopted.

It should be noted that some components of the trial vector may violate predefined boundary constraints. Possible solutions to tackle this problem include resetting schemes, penalty schemes, etc. Constrained problems, however, are not the main focus of this paper. Thus, we follow a simple method which sets the violating components to be the middle of the

violated bounds and the corresponding components of the parent individual [2], i.e.,

$$v_{j,i,g} = (x_j^{\text{low}} + x_{j,i,g})/2, \text{ if } v_{j,i,g} < x_j^{\text{low}}$$
$$v_{j,i,g} = (x_j^{\text{up}} + x_{j,i,g})/2, \text{ if } v_{j,i,g} > x_j^{\text{up}}$$

where $v_{j,i,g}$ and $x_{j,i,g}$ denote the jth components of the mutation vector $\mathbf{v}_{i,g}$ and the parent vector $\mathbf{x}_{i,g}$ at generation g, respectively. This method works well especially when the optimal solution is located near or on the boundary.

Crossover: After mutation, a binomial crossover operation forms the final trial/offspring vector $\mathbf{u}_{i,g} = (u_{1,i,g}, u_{2,i,g}, \ldots, u_{D,i,g})$

$$u_{j,i,g} = \begin{cases} v_{j,i,g}, & \text{if rand(0,1)} \le CR_i \text{ or } j = j_{\text{rand}}, \\ x_{j,i,g}, & \text{otherwise} \end{cases} \quad (4)$$

where rand(a, b) is a uniform random number on the interval [a, b] and independently generated for each j and each i, $j_{rand} = \text{randint}(1, D)$ is an integer randomly chosen from 1 to D and newly generated for each i, and the crossover probability $CR_i \in [0, 1]$ roughly corresponds to the average fraction of vector components that are inherited from the mutation vector. In classic DE, $CR_i = CR$ is a fixed parameter used to generate all trial vectors at all generations, while in many adaptive DE algorithms each individual i is associated with its own crossover probability CR_i.

Selection: The selection operation selects the better one from the parent vector $\mathbf{x}_{i,g}$ and the trial vector $\mathbf{u}_{i,g}$ according to their fitness values $f(\cdot)$. For example, if we have a minimization problem, the selected vector is given by

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g}, & \text{if } f(\mathbf{u}_{i,g}) < f(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g}, & \text{otherwise} \end{cases} \quad (5)$$

and is used as a parent vector in the next generation.

The operation in (5) is called a *successful update* if the trial vector $\mathbf{u}_{i,g}$ is better than the parent $\mathbf{x}_{i,g}$, i.e., the improvement or evolution progress $\Delta_{i,g} = f(\mathbf{x}_{i,g}) - f(\mathbf{u}_{i,g})$ is positive. Accordingly, the control parameters F_i and CR_i used to generate $\mathbf{u}_{i,g}$ are called a *successful mutation factor* and a *successful crossover probability*, respectively.

The above one-to-one selection procedure is generally kept fixed in different DE algorithms, while the crossover may have variants other than the binomial operation in (4). Thus, a classic differential evolution algorithm is historically named, for example, DE/rand/1/bin, connoting its DE/rand/1 mutation strategy and binomial crossover operation.

III. Adaptive DE Algorithms

This section briefly reviews some recent adaptive and self-adaptive DE algorithms that dynamically update control parameters as the evolutionary search proceeds. It provides a convenient way to compare these parameter adaptation mechanisms with the one proposed in JADE later.

A. DESAP

In [10], an algorithm DESAP is proposed to not only dynamically adapt mutation and crossover parameters η and δ (which is usually the case in other adaptive or self-adaptive algorithms) but also the population size π. The base strategy of DESAP is slightly different from the classic DE/rand/1/bin and rather similar to the scheme introduced in [9]—while δ and π have the same meaning as CR and NP, respectively, η refers to the probability of applying an additional normally distributed mutation after crossover. The ordinary mutation factor F is indeed kept fixed in DESAP.

In DESAP, each individual i is associated with its own control parameters δ_i, η_i, and π_i. These parameters undergo crossover and mutation, in a way similar to the corresponding vector $\mathbf{x}_i$. The new values of these control parameters survive together with $\mathbf{u}_i$ if $f(\mathbf{u}_i) < f(\mathbf{x}_i)$. In spite of its simple reasoning, DESAP performance is not satisfactory. It outperforms the conventional DE only in one of the five De Jong's test problems, while other results are very similar. Indeed, as the author states, DESAP is mainly used to demonstrate a possibility of further reducing control parameters by adaptively updating the population size as well as other control parameters.

B. FADE

The fuzzy adaptive differential evolution (FADE), which was introduced by Liu and Lampinen [11], is a new variant of DE that uses fuzzy logic controllers to adapt the control parameters F_i and CR_i for the mutation and crossover operations. A similar method has also been independently proposed for multiobjective DE optimization [12]. Similar to many other adaptive DE algorithms (except DESAP [10]), the population size is assumed to be tuned in advance and kept fixed throughout the evolution process of FADE. The fuzzy-logic controlled approach is tested with a set of 10 standard benchmark functions and shows better results than the classic DE when problem dimension is high.

C. SaDE

SaDE [13] is proposed by Qin and Suganthan to simultaneously implement two mutation strategies "DE/rand/1" and "DE/current-to-best/1." It adapts the probability of generating offspring by either strategy based on their success ratios in the past 50 generations. It is believed that this adaptation procedure can gradually evolve the most suitable mutation strategy at different learning stages for the problem under consideration This is similar to the scheme proposed in [27], [28], where competing heuristics (including several DE variants, simplex methods, and evolution strategies) are simultaneously adopted and the probabilities of generating offspring by any of these heuristics are adapted dynamically.

In SaDE, the mutation factors are independently generated at each generation according to a normal distribution with mean 0.5, standard deviation 0.3, and truncated to the interval (0, 2]. It states that this scheme can keep both local (with small F_i values) and global (with large F_i values) search ability to generate potentially good mutation vectors throughout the

evolution process. The crossover probabilities are randomly generated according to an independent normal distribution with mean CR_m and standard deviation 0.1. To the contrary of F_i, the CR_i values remain fixed for five generations before the next re-generation. The mean CR_m is initialized to be 0.5. To adapt CR to proper values, the authors update CR_m every 25 generations based on the recorded successful CR values since the last CR_m update.

To speed up the convergence of SaDE, the authors further apply a local search procedure (quasi-Newton method) to some good individuals after 200 generations.

While it is originally proposed to solve unconstrained optimization problems, SaDE is further extended by implementing five candidate mutation strategies in solving a set of constrained optimization problems [14].

D. jDE

Brest *et al.* [15] proposed a new adaptive DE, jDE, which is based on the classic DE/rand/1/bin. Similar to other schemes, jDE fixes the population size during the optimization while adapting the control parameters F_i and CR_i associated with each individual. The initialization process sets $F_i = 0.5$ and $CR_i = 0.9$ for each individual. Instead of SaDE's method which randomizes F_i and CR_i and updates CR_m by recorded successful values, jDE regenerates (with probabilities $\tau_1 = \tau_2 = 0.1$ at each generation) new values for F_i and CR_i according to uniform distributions on [0.1, 1] and [0, 1], respectively. It is believed that better parameter values tend to generate individuals which are more likely to survive and thus these values should be propagated to the next generation.

Experimental results suggest that jDE performs remarkably better than the classic DE/rand/1/bin, FEP and CEP [29], the adaptive LEP and Best Levy [30], and the FADE algorithm [11]. jDE is further extended by adapting two mutation strategies and the new algorithm, named jDE-2 [17], shows results competitive to the original SaDE (which implements a local quasi-Newton search procedure after 200 generations) on a set of 25 benchmark functions.

IV. JADE

In this section, we propose a new DE algorithm, JADE, which implements a mutation strategy "DE/current-to-$p$best" with optional archive and controls F and CR in an adaptive manner. JADE adopts the same crossover and selection operations as described in (4) and (5) in Section II.

A. DE/Current-to-pbest

DE/rand/1 is the first mutation strategy developed for DE [1], [2], and is said to be the most successful and widely used scheme in the literature [31]. However, [6] indicates that DE/best/2 may have some advantages over DE/rand/1, and [32] favors DE/best/1 for most technical problems investigated. Also, the authors of [33] argue that the incorporation of best solution information is beneficial and use DE/current-to-best/1 in their algorithm. Compared to DE/rand/k, greedy strategies such as DE/current-to-best/k and DE/best/k benefit from their fast convergence by incorporating best solution information in the evolutionary search. However, the best solution information may also cause problems such as premature convergence due to the resultant reduced population diversity.

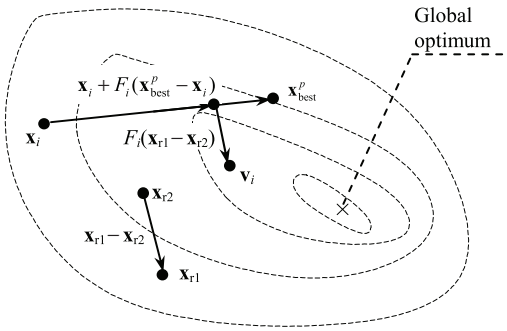

Fig. 1. Illustration of the DE/current-to-$p$best/1 mutation strategy adopted in JADE. The dashed curves display the contours of the optimization problem. $\mathbf{v}_i$ is the mutation vector generated for individual $\mathbf{x}_i$ using the associated mutation factor F_i.

In view of the fast but less reliable convergence performance of existing greedy strategies (2) and (3), a new mutation strategy, named DE/current-to-$p$best with optional archive, is proposed to serve as the basis of the adaptive DE algorithm JADE proposed in this paper. As illustrated in Fig. 1, in DE/current-to-$p$best/1 (without archive), a mutation vector is generated in the following manner:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F_i \cdot (\mathbf{x}^p_{\text{best},g} - \mathbf{x}_{i,g}) + F_i \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) \quad (6)$$

where $x^p_{\text{best},g}$ is randomly chosen as one of the top $100p\%$ individuals in the current population with $p \in (0, 1]$, and F_i is the mutation factor that is associated with $\mathbf{x}_i$ and is re-generated at each generation by the adaptation process introduced later in (10). DE/current-to-$p$best is indeed a generalization of DE/current-to-best. Any of the top $100p\%$ solutions can be randomly chosen to play the role of the single best solution in DE/current-to-best.

Recently explored inferior solutions, when compared to the current population, provide additional information about the promising progress direction. Denote $\mathbf{A}$ as the set of archived inferior solutions and $\mathbf{P}$ as the current population. In DE/current-to-$p$best/1 with archive, a mutation vector is generated as follows:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F_i(\mathbf{x}^p_{\text{best},g} - \mathbf{x}_{i,g}) + F_i(\mathbf{x}_{r1,g} - \tilde{\mathbf{x}}_{r2,g}) \quad (7)$$

where $\mathbf{x}_{i,g}$, $\mathbf{x}_{r1,g}$ and $\mathbf{x}^p_{\text{best},g}$ are selected from $\mathbf{P}$ in the same way as in (6), while $\tilde{\mathbf{x}}_{r2,g}$ is randomly chosen from the union, $\mathbf{P} \cup \mathbf{A}$, of the current population and the archive.

The archive operation is made very simple to avoid significant computation overhead. The archive is initiated to be empty. Then, after each generation, the parent solutions that fail in the selection process of (5) are added to the archive. If the archive size exceeds a certain threshold, say NP, then some solutions are randomly removed from the archive to keep the archive size at NP. It is clear that (6) is a special case of (7) by setting the archive size to be zero.

TABLE I
PSEUDO CODE OF JADE WITH ARCHIVE

line♯	Procedure of JADE with Archive
01	**Begin**
02	Set $\mu_{CR} = 0.5$; $\mu_F = 0.5$; A $= \emptyset$
03	Create a random initial population $\{\mathbf{x}_{i,0} \vert i = 1, 2, \ldots, NP\}$
04	**For** $g = 1$ to G
05	$S_F = \emptyset$; $S_{CR} = \emptyset$;
06	**For** $i = 1$ to NP
07	Generate $CR_i = \text{randn}_i(\mu_{CR}, 0.1)$, $F_i = \text{randc}_i(\mu_F, 0.1)$
08	Randomly choose $\mathbf{x}^p_{\text{best},g}$ as one of the $100p\%$ best vectors
09	Randomly choose $\mathbf{x}_{r1,g} \neq \mathbf{x}_{i,g}$ from current population $\mathbf{P}$
10	Randomly choose $\tilde{\mathbf{x}}_{r2,g} \neq \mathbf{x}_{r1,g} \neq \mathbf{x}_{i,g}$ from $\mathbf{P} \cup \mathbf{A}$
11	$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F_i \cdot (\mathbf{x}^p_{\text{best},g} - \mathbf{x}_{i,g}) + F_i \cdot (\mathbf{x}_{r1,g} - \tilde{\mathbf{x}}_{r2,g})$
12	Generate $j_{\text{rand}} = \text{randint}(1, D)$
13	**For** $j = 1$ to D
14	**If** $j = j_{\text{rand}}$ or rand(0, 1)$< CR_i$
15	$u_{j,i,g} = v_{j,i,g}$
16	**Else**
17	$u_{j,i,g} = x_{j,i,g}$
18	**End If**
19	**End For**
20	**If** $f(\mathbf{x}_{i,g}) \leq f(\mathbf{u}_{i,g})$
21	$\mathbf{x}_{i,g+1} = \mathbf{x}_{i,g}$
22	**Else**
23	$\mathbf{x}_{i,g+1} = \mathbf{u}_{i,g}$; $\mathbf{x}_{i,g} \rightarrow \mathbf{A}$; $CR_i \rightarrow S_{CR}$, $F_i \rightarrow S_F$
24	**End If**
25	**End for**
26	Randomly remove solutions from $\mathbf{A}$ so that $\vert\mathbf{A}\vert \leq NP$
27	$\mu_{CR} = (1 - c) \cdot \mu_{CR} + c \cdot \text{mean}_A(S_{CR})$
28	$\mu_F = (1 - c) \cdot \mu_F + c \cdot \text{mean}_L(S_F)$
29	**End for**
30	**End**

The archive provides information about the progress direction and is also capable of improving the diversity of the population. In addition, as shown later in the parameter adaptation operations (10)–(12), we encourage large F values that are helpful to increase the population diversity. Thus, although it is biased toward the direction of potential optimum (possibly a local minimum), the proposed DE/current-to-$p$best/1 is not liable to be trapped to a local minimum. As a simple comparison, DE/rand/1 searches a relatively small region which shows no bias to any special directions, while DE/current-to-$p$best/1 with archive searches a relatively large region which is biased toward promising progress directions.

The pseudo code of JADE is presented in Table I. We next introduce the parameter adaptation of F and CR.

B. Parameter Adaptation

At each generation g, the crossover probability CR_i of each individual $\mathbf{x}_i$ is independently generated according to a normal distribution of mean μ_{CR} and standard deviation 0.1

$$CR_i = \text{randn}_i(\mu_{CR}, 0.1) \tag{8}$$

and then truncated to [0, 1]. Denote S_{CR} as the set of all successful crossover probabilities CR_i's at generation g. The mean μ_{CR} is initialized to be 0.5 and then updated at the end of each generation as

$$\mu_{CR} = (1 - c) \cdot \mu_{CR} + c \cdot \text{mean}_A(S_{CR}) \tag{9}$$

where c is a positive constant between 0 and 1 and $\text{mean}_A(\cdot)$ is the usual arithmetic mean.

Similarly, at each generation g, the mutation factor F_i of each individual $\mathbf{x}_i$ is independently generated according to a Cauchy distribution with location parameter μ_F and scale parameter 0.1

$$F_i = \text{randc}_i(\mu_F, 0.1) \tag{10}$$

and then truncated to be 1 if $F_i \geq 1$ or regenerated if $F_i \leq 0$. Denote S_F as the set of all successful mutation factors in generation g. The location parameter μ_F of the Cauchy distribution is initialized to be 0.5 and then updated at the end of each generation as

$$\mu_F = (1 - c) \cdot \mu_F + c \cdot \text{mean}_L(S_F) \tag{11}$$

where $\text{mean}_L(\cdot)$ is the Lehmer mean

$$\text{mean}_L(S_F) = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F}. \tag{12}$$

C. Explanations of the Parameter Adaptation

The adaptation of μ_{CR} is based on the following principle. Better control parameter values tend to generate individuals that are more likely to survive and thus these values should be propagated to the following generations. The basic operation is thus to record recent successful crossover probabilities and use them to guide the generation of new CR_i's. The standard deviation is set to be relatively small in (8) because otherwise the adaptation does not function efficiently; e.g., in the extreme case of an infinite standard derivation, the truncated normal distribution becomes independent of the value of μ_{CR}.

Compared to CR, there are two distinct operations in the adaptation of μ_F. First, F_i's are generated according to a truncated Cauchy distribution. Compared to a normal distribution, the Cauchy distribution is more helpful to diversify the mutation factors and thus avoid premature convergence which often occurs in greedy mutation strategies (such as DE/best, DE/current-to-best, and DE/current-to-$p$best) if the mutation factors are highly concentrated around a certain value.

Second, the adaptation of μ_F places more weight on larger successful mutation factors by using the Lehmer mean in (12), instead of an arithmetic mean as used in the μ_{CR} adaptation. The Lehmer mean is therefore helpful to propagate larger mutation factors, which in turn improve the progress rate. To the contrary, an arithmetic mean of S_F tends to be smaller than the optimal value of the mutation factor and thus leads to a smaller μ_F and causes premature convergence at the end. The tendency of a smaller μ_F is mainly due to the discrepancy between success probability and progress rate in the evolutionary search. Indeed, the DE/current-to-$p$best with small F_i is similar to a $(1 + 1)$ evolution strategy (ES) [23] in the sense that both generate an offspring in the small neighborhood of the base vector. For $(1 + 1)$ ES, it is known that the smaller the mutation variance, usually the higher the

successful probability (this is strictly proven for the corridor and sphere functions [23], [34]). However, a mutation variance close to 0 obviously leads to a trivial evolution progress. A simple yet efficient method is to place more weight on larger successful mutation factors so as to achieve faster progress in the evolutionary search.

Regarding the constant c in (9) and (11), no parameter adaptation takes place if $c = 0$. Otherwise, the *life span* of a successful CR_i or F_i is roughly $1/c$ generations; i.e., after $1/c$ generations, the old value of μ_{CR} or μ_F is reduced by a factor of $(1-c)^{1/c} \to 1/e \approx 37\%$, when c is close to zero.

D. Discussion of Parameter Settings

The classic DE has two control parameters F and CR that need to be tuned by the user. These parameters are known to be problem-dependent and thus tedious trial and error is usually required to find their appropriate values for each specific problem. To the contrary, it is expected that the two new parameters c and p of JADE are insensitive to different problems according to their roles in JADE: c controls the rate of parameter adaptation and p determines the greediness of the mutation strategy. As shown in Section V, JADE usually performs best with $1/c \in [5, 20]$ and $p \in [5\%, 20\%]$; i.e., the life span of μ_{CR} and μ_F values ranges from 5 to 20 generations, and we consider the top 5–20% high-quality solutions in the mutation.

V. Simulation

In this section, JADE is applied to minimize a set of 13 scalable benchmark functions of dimensions $D = 30$ or 100 [29], [35] and a set of Dixon–Szegö functions of lower dimension $D = 2 \sim 6$ [36], as shown in Tables II and III.

JADE is compared with two recent adaptive DE algorithms jDE and SaDE, the classic DE/rand/1/bin, and a canonical PSO algorithm [37]. It is also compared with rand-JADE and nona-JADE, its two variants, to identify the benefits from its different components. For fair comparison, we set the parameters of JADE to be fixed, $p = 0.05$ and $c = 0.1$, in all simulations. We follow the parameter settings in the original paper of jDE [15] and SaDE [13], except that the quasi-Newton local search is disabled in SaDE. For PSO, the parameter values are chosen from [37], as they usually work better than the values proposed in other literature we studied. The parameters of DE/rand/1/bin are set to be $F = 0.5$ and $CR = 0.9$, as used or recommended in [1], [8], [15], [38].

Summarized in Table II are the 13 scalable benchmark functions. While all these functions have an optimal value $f^* = 0$, some different characteristics are briefly summarized as follows. f_1–f_4 are continuous unimodal functions. f_5 is the Rosenbrock function which is unimodal for $D = 2$ and 3 but may have multiple minima in high dimension cases [39]. f_6 is a discontinuous step function, and f_7 is a noisy quartic function. f_8–f_{13} are multimodal and the number of their local minima increases exponentially with the problem dimension [29]. In addition, f_8 is the only bound-constrained function investigated in this paper. The characteristics of the multimodal Dixon–Szegö functions are briefly described in Table III. Interested readers are referred to [36] for details.

TABLE II
Test Functions of Dimension D. Each of Them Has a Global Minimum Value of 0. These Are Sphere, Schwefel 2.22, Schwefel 1.2, Schwefel 2.21, Rosenbrock, Step, Noisy Quartic, Schwefel 2.26, Rastrigin, Ackley, Griewank, and Two Penalized Functions, Respectively [35]

Test functions	Initial range
$f_1(x) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$
$f_2(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	$[-10, 10]^D$
$f_3(x) = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)^2$	$[-100, 100]^D$
$f_4(x) = \max_i \{\lvert x_i \rvert\}$	$[-100, 100]^D$
$f_5(x) = \sum_{i=1}^{D-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$	$[-30, 30]^D$
$f_6(x) = \sum_{i=1}^{D} \lfloor x_i + 0.5 \rfloor^2$	$[-100, 100]^D$
$f_7(x) = \sum_{i=1}^{D} i x_i^4 + rand\,[0, 1)$	$[-1.28, 1.28]^D$
$f_8(x) = \sum_{i=1}^{D} -x_i \sin\sqrt{\lvert x_i \rvert} + D \cdot 418.98288727243369$	$[-500, 500]^D$
$f_9(x) = \sum_{i=1}^{D} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	$[-5.12, 5.12]^D$
$f_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e$	$[-32, 32]^D$
$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^D$
$f_{12}(x) = \frac{\pi}{D}\{10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}(y_i - 1)^2\left[1 + 10\sin^2(\pi y_{i+1})\right] + (y_D - 1)^2\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$ where $y_i = 1 + \frac{1}{4}(x_i + 1)$ and $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a \le x_i \le a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	$[-50, 50]^D$
$f_{13}(x) = 0.1\Big\{\sin^2(3\pi x_1) + \sum_{i=1}^{D-1}(x_i - 1)^2\left[1 + \sin^2(3\pi x_{i+1})\right] + (x_D - 1)^2\left[1 + \sin^2(2\pi x_D)\right]\Big\} + \sum_{i=1}^{D} u(x_i, 5, 100, 4)$	$[-50, 50]^D$

In all simulations, we set the population size NP to be 30, 100, and 400 in the case of $D \le 10$, $= 30$, and $= 100$, respectively. All results reported in this section are obtained based on 50 independent runs.

For clarity, the results of the best and second best algorithms are marked in **boldface** and *italic*, respectively; if not all or most algorithms produce identical results.

A. Comparison of JADE With Other Evolutionary Algorithms

The mean and the standard deviation of the results obtained by each algorithm for f_1–f_{13} are summarized in Tables IV and V. These statistics are calculated for f_1–f_4, and f_7 only at the end of the optimization. For other functions, middle results are also reported because the final results obtained by different algorithms might be identical to zero or a certain

TABLE III
TEST FUNCTIONS OF DIMENSION D. EACH OF THEM HAS A GLOBAL MINIMUM VALUE OF 0. THESE ARE SPHERE, SCHWEFEL 2.22, SCHWEFEL 1.2, SCHWEFEL 2.21, ROSENBROCK, STEP, NOISY QUARTIC, SCHWEFEL 2.26, RASTRIGIN, ACKLEY, GRIEWANK, AND TWO PENALIZED FUNCTIONS, RESPECTIVELY [36]

Functions	D	Initial range	Number of local min	Number of global min	Approx. glo. min
f_{14}: Branin	2	$[-5, 10]\times[0, 15]$	3	3	0.397887
f_{15}: Goldstein-Price	2	$[-2, 2]^2$	4	1	3.00000
f_{16}: Hartman3	3	$[0, 1]^3$	4	1	–3.86278
f_{17}: Hartman6	6	$[0, 1]^6$	4	1	–3.32237
f_{18}: Shekel5	4	$[0, 10]^4$	5	1	–10.1532
f_{19}: Shekel7	4	$[0, 10]^4$	7	1	–10.4029
f_{20}: Shekel10	4	$[0, 10]^4$	10	1	–10.5364

residual error (see the error floor of f_{10} and f_{13} in Fig. 3) due to precision problems of MATLAB.[2] In these cases, only the middle results are compared and may be marked in boldface or italic.

In Tables VI and VII, we summarize the *s*uccess *r*ate (SR) of each algorithm and the average number of *f*unction *e*valuation*s* over *s*uccessful runs (FESS). An experiment is considered as successful if the best solution is found with sufficient accuracy: 10^{-2} for the noisy function f_7 and 10^{-8} for all others. FESS and SR are useful to compare the convergence rate (in successful runs) and the reliability of different algorithms, respectively.

For convenience of illustration, we plot the convergence graph for some 30- and 100-dimensional problems in Figs. 2 and 3, respectively. Note that in these graphs we plot the curves of median values (instead of the mean values reported in the tables), because these curves, together with box-and-whisker diagrams, provide more information when an algorithm may lead to false convergence only occasionally. The box-and-whisker diagrams are plotted at certain generations for the best and second best algorithms. It is helpful to illustrate the spread of results over 50 independent runs.

Important observations about the convergence rate and reliability of different algorithms can be made from the results presented in Figs. 2 and 3, and Tables IV and V. First, these results suggest that the overall convergence rate of JADE with or without archive is the best and second best for the set of problems f_1–f_{13}: JADE without archive works best for relatively low dimensional problems ($D = 30$), while JADE with archive achieves the best performance in the case of high dimensional problems ($D = 100$). It may be because a population size of 400 is not sufficient for various 100-dimensional problems, while the archive can relieve this problem to some extent by introducing more diversity in the mutation operation. In both cases (with and without archive), JADE converges fastest for the optimization of f_1–f_7 and f_{10}–f_{13}, which are either unimodal or multimodal, clean or noisy, continuous or discontinuous. In the cases of f_8 and f_9, jDE performs best and JADE achieves very competitive results. The convergence rates of SaDE and PSO are usually worse than JADE and jDE but better than DE/rand/1/bin.

Second, it is important to compare the reliability of different algorithms, as shown in Tables VI and VII. PSO and DE/rand/1/bin perform worst because the former suffers from frequent premature convergence while the latter usually converges very slowly within a limited number of function evaluations. SaDE is better than PSO and DE/rand/1/bin but is clearly less satisfactory for some problems especially when problem dimension is high. jDE, as expected, is shown to work very well because its underlying mutation strategy "DE/rand/1" is more robust than greedy strategies. It is interesting that the reliability of JADE is similar to that of jDE for all the 13 scalable functions. The high reliability and fast convergence of JADE stem from both the adaptive parameter control and the direction information and diversity improvement provided by the archive assisted "DE/current-to-$p$best/1" mutation strategy.

In the case of low-dimensional Dixon–Szegö functions, simulation results in Table VIII show that there is no obviously superior algorithm. It is similar to the observation in the literature [15], where adaptive algorithms do not show obvious performance improvement over the classic DE/rand/1/bin for this set of functions. This observation is not surprising after further investigation. First, various simulations of DE/rand/1/bin with F and CR chosen from $\{0.1, 0.3, 0.5, 0.7, 0.9\}$ show that the parameter setting of $F = 0.5$ and $CR = 0.9$ consistently leads to the best or near-best performance for all Dixon–Szegö functions. Second, simulation results also indicate that parameter adaptation does not function efficiently within the small number of generations required to optimize these low dimensional problems.

In Table IX, JADE is further compared with other evolutionary algorithms based on their results reported in the literature: the adaptive LEP and Best Levy algorithms in [30, Table III], NSDE in [40, Tables 1–3] and jDE in [15, Table III]. It is clear that JADE works best or second best in most cases and achieves overall better performance than other competitive algorithms.

B. The Benefit of JADE Components

We are interested in identifying the benefit of the two components of JADE without archive: "DE/current-to-$p$best" mutation strategy and parameter adaptation. For this purpose, we consider two variants of JADE, i.e., rand-JADE and nona-JADE. They differ from JADE without archive *only* in that rand-JADE implements DE/rand/1 (instead of DE/current-to-$p$best) mutation strategy, while nona-JADE does not adopt adaptive parameter control [i.e., instead of updating μ_{CR} and μ_F according to (9) and (11), we set $\mu_{CR} = \mu_F = 0.5$ in (8) and (10) throughout the optimization process].

Summarized in Tables VI and VII are the simulation results of JADE, rand-JADE, nona-JADE, and DE/rand/1/bin. The FESS performance indicates that nona-JADE (in most cases) and rand-JADE (in some cases) converge faster than DE/rand/1/bin, due to their relatively greedy mutation strategy and adaptive parameter control, respectively. However, both rand-JADE and nona-JADE suffer from frequent premature

[2]Mathwork Inc. http://www.mathwork.com

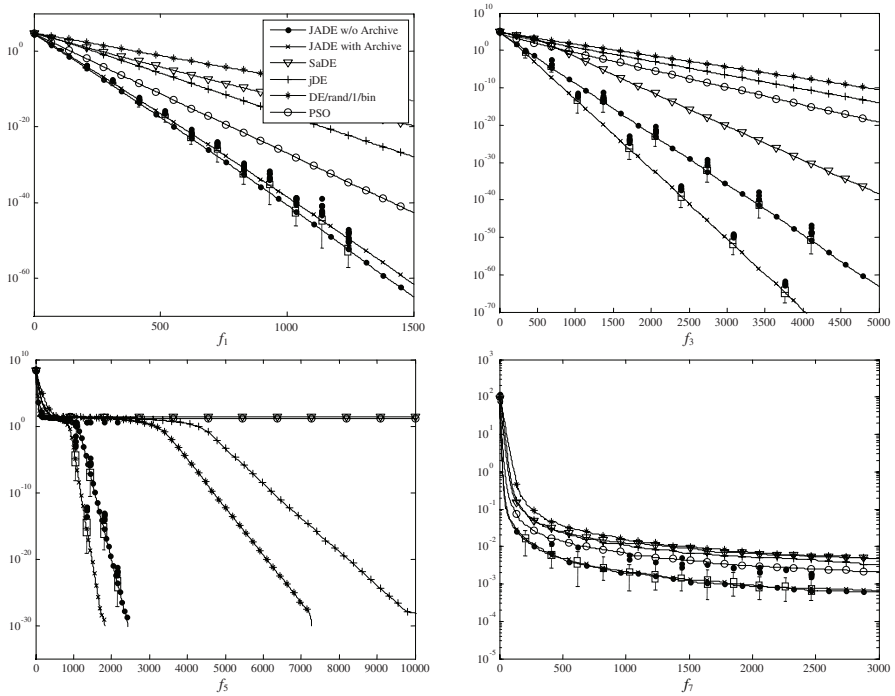

Fig. 2. Convergence graph (median curves and box-and-whisker diagrams) for test functions f_1, f_3, f_5, and $f_7(D = 30)$. The horizontal axis is the number of generations, and the vertical axis is the median of function values over 50 independent runs. The box-and-whisker diagrams are plotted for both JADE and the best one among other algorithms.

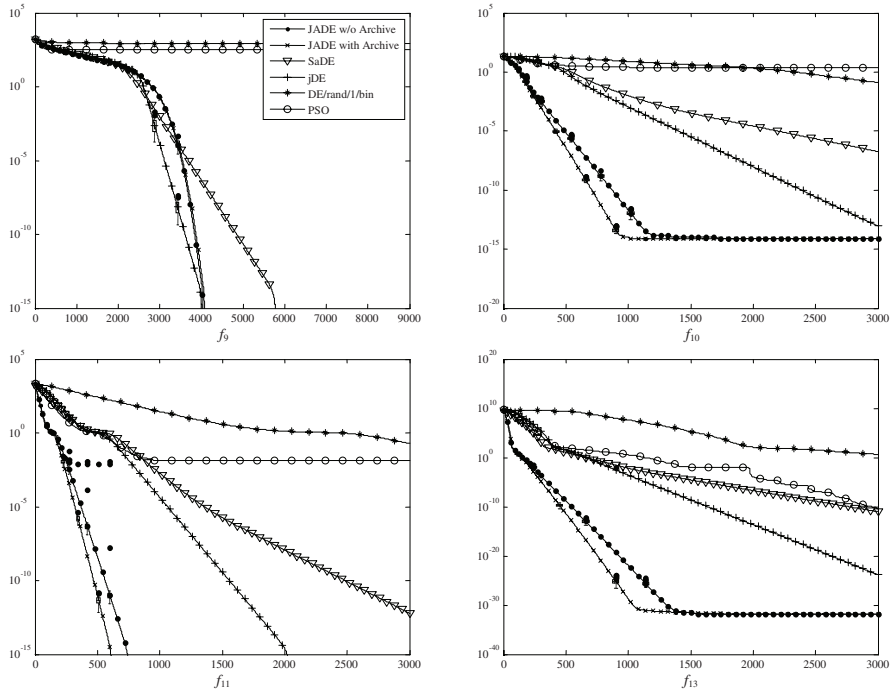

Fig. 3. Convergence graph (median curves and box-and-whisker diagrams) for test functions f_9, f_{10}, f_{11}, and $f_{13}(D = 100)$. The horizontal axis is the number of generations, and the vertical axis is the median of function values over 50 independent runs. The box-and-whisker diagrams are plotted for both JADE and the best one among other algorithms.

convergence for some functions. For example, the success rate of rand-JADE is clearly less satisfactory in the optimization of f_3 and f_5. It implies that rand-JADE is incapable of maintaining sufficiently high population diversity due to the ill-conditioned Hessian matrices of f_3 and f_5 (indicating a narrow valley) around the optimal point.

Compared to its two variants, JADE achieves remarkably better performance in terms of convergence rate and reliability. It indicates a mutually beneficial cooperation between the greedy strategy "DE/current-to-$p$best" and the parameter adaptation. In fact, a greedy mutation strategy may affect the population diversity in two opposite directions: it tends

TABLE IV
EXPERIMENTAL RESULTS OF 30-DIMENSIONAL PROBLEMS f_1–f_{13}, AVERAGED OVER 50 INDEPENDENT RUNS

	Gen	JADE w/o archive Mean (Std Dev)	JADE with archive Mean (Std Dev)	jDE Mean (Std Dev)	SaDE Mean (Std Dev)	DE/rand/1/bin Mean (Std Dev)	PSO Mean (Std Dev)
f_1	1500	**1.8E–60 (8.4E–60)**	*1.3E–54 (9.2E–54)*	2.5E–28 (3.5E–28)	4.5E–20 (6.9E–20)	9.8E–14 (8.4E–14)	9.6E–42 (2.7E–41)
f_2	2000	**1.8E–25 (8.8E–25)**	3.9E–22 (2.7E–21)	*1.5E–23 (1.0E–23)*	1.9E–14 (1.05E–14)	1.6E–09 (1.1E–09)	9.3E–21 (6.3E–20)
f_3	5000	*5.7E–61 (2.7E–60)*	**6.0E–87 (1.9E–86)**	5.2E–14 (1.1E–13)	9.0E–37 (5.43E–36)	6.6E–11 (8.8E–11)	2.5E–19 (3.9E–19)
f_4	5000	*8.2E–24 (4.0E–23)*	**4.3E–66 (1.2E–65)**	1.4E–15 (1.0E–15)	7.4E–11 (1.82E–10)	4.2E–01 (1.1E+00)	4.4E–14 (9.3E–14)
f_5	3000	**8.0E–02 (5.6E–01)**	*3.2E–01 (1.1E+00)*	1.3E+01 (1.4E+01)	2.1E+01 (7.8E+00)	2.1E+00 (1.5E+00)	2.5E+01 (3.2E+01)
	20000	8.0E–02 (5.6E–01)	3.2E–01 (1.1E+00)	8.0E–02 (5.6E–01)	1.8E+01 (6.7E+00)	8.0E–02 (5.6E–01)	1.7E+01 (2.3E+01)
f_6	100	**2.9E+00 (1.2E+00)**	*5.6E+00 (1.6E+00)*	1.0E+03 (2.2E+02)	9.3E+02 (1.8E+02)	4.7E+03 (1.1E+03)	4.5E+01 (2.4E+01)
	1500	0.0E+00 (0.0E+00)	0.0E+00 (0.0E+00)	0.0E+00 (0.0E+00)	0.0E+00 (0.0E+00)	0.0E+00 (0.0E+00)	8.0E–02 (2.7E–01)
f_7	3000	**6.4E–04 (2.5E–04)**	*6.8E–04 (2.5E–04)*	3.3E–03 (8.5E–04)	4.8E–03 (1.2E–03)	4.7E–03 (1.2E–03)	2.5E–03 (1.4E–03)
f_8	1000	*3.3E–05 (2.3E–05)*	7.1E+00 (2.8E+01)	**7.9E–11 (1.3E–10)**	4.7E+00 (3.3E+01)	5.9E+03 (1.1E+03)	2.4E+03 (6.7E+02)
	9000	0.0E+00 (0.0E+00)	7.1E+00 (2.8E+01)	0.0E+00 (0.0E+00)	4.7E+00 (3.3E+01)	5.7E+01 (7.6E+01)	2.4E+03 (6.7E+02)
f_9	1000	**1.0E–04 (6.0E–05)**	*1.4E–04 (6.5E–05)*	1.5E–04 (2.0E–04)	1.2E–03 (6.5E–04)	1.8E+02 (1.3E+01)	5.2E+01 (1.6E+01)
	5000	0.0E+00 (0.0E+00)	0.0E+00 (0.0E+00)	0.0E+00 (0.0E+00)	0.0E+00 (0.0E+00)	7.1E+01 (2.1E+01)	5.2E+01 (1.6E+01)
f_{10}	500	**8.2E–10 (6.9E–10)**	*3.0E–09 (2.2E–09)*	3.5E–04 (1.0E–04)	2.7E–03 (5.1E–04)	1.1E–01 (3.9E–02)	4.6E–01 (6.6E–01)
	2000	4.4E–15 (0.0E+00)	4.4E–15 (0.0E+00)	4.7E–15 (9.6E–16)	4.3E–14 (2.6E–14)	9.7E–11 (5.0E–11)	4.6E–01 (6.6E–01)
f_{11}	500	**9.9E–08 (6.0E–07)**	2.0E–04 (1.4E–03)	*1.9E–05 (5.8E–05)*	7.8E–04 (1.2E–03)	2.0E–01 (1.1E–01)	1.3E–02 (1.7E–02)
	3000	0.0E+00 (0.0E+00)	2.0E–04 (1.4E–03)	0.0E+00 (0.0E+00)	0.0E+00 (0.0E+00)	0.0E+00 (0.0E+00)	1.1E–02 (1.6E–02)
f_{12}	500	**4.6E–17 (1.9E–16)**	*3.8E–16 (8.3E–16)*	1.6E–07 (1.5E–07)	1.9E–05 (9.2E–06)	1.2E–02 (1.0E–02)	1.9E–01 (3.9E–01)
	1500	1.6E–32 (5.5E–48)	1.6E–32 (5.5E–48)	2.6E–29 (7.5E–29)	1.2E–19 (2.0E–19)	1.1E–14 (1.0E–14)	1.9E–01 (3.9E–01)
f_{13}	500	**2.0E–16 (6.5E–16)**	*1.2E–15 (2.8E–15)*	1.5E–06 (9.8E–07)	6.1E–05 (2.0E–05)	7.5E–02 (3.8E–02)	2.9E–03 (4.8E–03)
	1500	1.4E–32 (1.1E–47)	1.4E–32 (1.1E–47)	1.9E–28 (2.2E–28)	1.7E–19 (2.4E–19)	7.5E–14 (4.8E–14)	2.9E–03 (4.8E–03)

TABLE V
EXPERIMENTAL RESULTS OF 100-DIMENSIONAL PROBLEMS f_1–f_{13}, AVERAGED OVER 50 INDEPENDENT RUNS

	Gen	JADE w/o archive Mean (Std Dev)	JADE with archive Mean (Std Dev)	jDE Mean (Std Dev)	SaDE Mean (Std Dev)	DE/rand/1/bin Mean (Std Dev)	PSO Mean (Std Dev)
f_1	2000	*1.2E–48 (1.5E–48)*	**5.4E–67 (1.6E–66)**	5.0E–15 (1.7E–15)	2.9E–08 (3.2E–08)	3.5E+01 (9.6E+00)	6.0E–11 (2.5E–10)
f_2	3000	*1.1E–41 (5.1E–41)*	**9.2E–51 (2.2E–50)**	4.1E–15 (1.1E–15)	1.7E–05 (3.8E–06)	2.3E+00 (5.6E–01)	2.8E–04 (1.3E–03)
f_3	8000	*1.2E–26 (2.0E–26)*	**2.2E–37 (2.5E–37)**	5.4E–02 (2.7E–02)	2.4E–13 (5.2E–13)	2.1E+05 (3.1E+04)	1.2E+02 (6.7E+01)
f_4	15000	1.9E–02 (1.5E–02)	**3.2E–71 (8.3E–71)**	*3.1E–09 (5.9E–10)*	1.1E+00 (4.0E–01)	9.3E+01 (2.8E+00)	4.9E+01 (2.5E+01)
f_5	6000	*5.6E–01 (1.4E+00)*	**4.0E–01 (1.2E+00)**	7.2E+01 (1.1E+01)	9.4E+01 (4.0E–01)	9.5E+01 (1.4E+01)	1.3E+02 (4.8E+01)
	20000	5.6E–01 (1.4E+00)	4.0E–01 (1.2E+00)	9.1E–03 (2.5E–02)	9.1E+01 (3.1E–01)	4.3E+01 (1.2E+01)	6.3E+01 (4.2E+01)
f_6	100	**1.1E+02 (1.5E+01)**	*1.2E+02 (1.3E+01)*	7.1E+04 (6.0E+03)	3.3E+04 (2.1E+03)	1.8E+05 (1.8E+04)	1.9E+04 (4.5E+03)
	1500	1.6E–01 (3.7E–01)	0.0E+00 (0.0E+00)	0.0E+00 (0.0E+00)	0.0E+00 (0.0E+00)	3.2E+02 (6.3E+01)	4.7E+01 (7.9E+01)
f_7	6000	*1.1E–03 (2.1E–04)*	**7.8E–04 (1.4E–04)**	8.1E–03 (9.0E–04)	1.0E–02(4.9E–03)	2.9E–02 (5.7E–03)	9.2E–03 (2.6E–03)
f_8	1000	8.9E+03 (3.0E+02)	8.6E+03 (4.2E+02)	**4.9E+03 (4.1E+02)**	*5.4E+03(3.7E+02)*	3.2E+04 (4.7E+02)	9.5E+03 (1.3E+03)
	9000	1.1E–10 (0.0E+00)	1.1E–10 (0.0E+00)	1.1E–10 (0.0E+00)	1.1E–10 (0.0E+00)	3.0E+04 (9.0E+02)	9.4E+03 (1.2E+03)
f_9	3000	1.9E–01 (3.8E–02)	2.0E–01 (3.7E–02)	**2.1E–04 (2.1E–04)**	*9.1E–03 (1.8E–03)*	8.6E+02 (2.2E+01)	3.4E+02 (4.4E+01)
	9000	0.0E+00 (0.0E+00)	0.0E+00 (0.0E+00)	0.0E+00 (0.0E+00)	0.0E+00 (0.0E+00)	8.1E+02 (1.8E+01)	3.4E+02 (4.4E+01)
f_{10}	500	*7.9E–06 (2.6E–06)*	**4.2E–07 (1.2E–07)**	8.5E–01 (1.2E–01)	1.6E+00 (1.2E–01)	1.5E+01 (5.8E–01)	3.6E+00 (9.3E–01)
	3000	8.9E–15 (2.1E–15)	8.0E–15 (0.0E+00)	9.9E–14 (2.0E–14)	2.1E–07 (1.0E–07)	1.3E–01 (2.4E–02)	2.6E+00 (6.8E–01)
f_{11}	500	*3.9E–04 (2.0E–03)*	**1.5E–04 (1.0E–03)**	1.1E+00 (2.0E–02)	1.1E+00 (1.8E–02)	2.7E+02 (4.4E+01)	1.0E+00 (5.6E–01)
	3000	3.9E–04 (2.0E–03)	1.5E–04 (1.0E–03)	0.0E+00 (0.0E+00)	8.6E–13 (8.2E–13)	2.0E–01 (5.8E–02)	8.8E–02 (2.5E–01)
f_{12}	500	2.2E–11 (1.2E–11)	**2.8E–13 (9.8E–13)**	4.0E+00 (6.8E–01)	2.4E+00 (3.9E–01)	1.8E+09 (5.1E+08)	1.1E+01 (3.4E+00)
	3000	4.7E–33 (2.2E–34)	4.7E–33 (6.8E–49)	1.7E–25 (7.7E–26)	1.4E–11 (9.1E–12)	1.7E+00 (1.5E+00)	1.3E–01 (1.6E–01)
f_{13}	500	*1.9E–09 (1.5E–09)*	**5.8E–12 (5.5E–12)**	3.1E+01 (7.8E+00)	1.2E+01 (1.8E+00)	2.4E+09 (1.1E+09)	9.8E+01 (2.4E+01)
	3000	1.4E–32 (1.1E–47)	1.4E–32 (1.1E–47)	2.1E–24 (1.5E–24)	1.3e–11 (9.6E–12)	5.1E+00 (3.2E+00)	1.4E–01 (5.6E–01)

to decrease the diversity by moving individuals closer to a few best solutions, but it is also possible to increase the diversity by speeding the optimization search in the progress direction. A greedy mutation strategy usually leads to premature convergence in classic DE algorithms without parameter adaptation, because the former effect of diversity decrease plays the key role. However, the parameter adaptation scheme is able to adapt parameters to appropriate

TABLE VI

EXPERIMENTAL RESULTS OF THE 30-DIMENSIONAL PROBLEMS f_1–f_{13}, AVERAGED OVER 50 INDEPENDENT RUNS. NOTE THAT THE BEST AND THE SECOND BEST AMONG JADE AND OTHER COMPETITIVE ALGORITHMS (EXCEPT JADE TWO VARIANTS) ARE MARKED IN BOLDFACE AND ITALIC, RESPECTIVELY

Functions		f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}
JADE	SR	100	100	100	100	98	100	100	100	100	100	100	100	100
(w/o archive)	FESS	**2.9E+4**	**5.2E+4**	*9.4E+4*	*1.7E+5*	*1.5E+5*	**1.1E+4**	**2.9E+4**	1.3E+5	*1.3E+5*	**4.5E+4**	**3.3E+4**	**2.7E+4**	**3.0E+4**
JADE	SR	100	100	100	100	96	100	100	94	100	100	100	100	100
(with archive)	FESS	*3.0E+4*	*5.6E+4*	**7.7E+4**	**7.4E+4**	**1.1E+5**	*1.2E+4*	*3.1E+4*	1.3E+5	*1.3E+5*	*4.7E+4*	*3.7E+4*	*2.9E+4*	*3.1E+4*
jDE	SR	100	100	100	100	98	100	100	100	100	100	100	100	100
	FESS	6.0E+4	8.3E+4	3.4E+5	3.0E+5	5.8E+5	2.3E+4	1.0E+5	**8.9E+4**	**1.2E+5**	9.1E+4	6.3E+4	5.5E+4	6.0E+4
SaDE	SR	100	100	100	100	24	100	100	100	100	100	100	100	100
	FESS	7.3E+4	1.2E+5	1.8E+5	2.9E+5	2.8E+5	2.7E+4	1.3E+5	*1.2E+5*	1.7E+5	1.2E+5	8.0E+4	7.1E+4	7.4E+4
DE/rand	SR	100	100	100	6	98	100	100	60	0	100	100	100	100
/1/bin	FESS	1.1E+5	1.9E+5	4.2E+5	3.5E+5	4.4E+5	4.2E+4	1.5E+5	3.5E+5	–	1.7E+5	1.1E+5	9.9E+4	1.1E+5
PSO	SR	100	100	100	100	4	92	100	0	0	66	36	62	74
	FESS	4.0E+4	6.7E+4	2.6E+5	3.1E+5	1.2E+6	2.4E+4	7.4E+4	–	–	6.5E+4	4.3E+4	4.8E+4	4.3E+4
rand-JADE	SR	100	100	38	66	0	100	84	100	100	84	100	100	100
(w/o archive)	FESS	1.2E+5	1.9E+5	2.9E+5	3.2E+5	–	3.6E+4	2.1E+5	1.6E+5	1.6E+5	2.0E+5	1.4E+5	1.1E+5	1.2E+5
nona-JADE	SR	100	100	100	0	88	100	100	16	0	100	100	100	100
(w/o archive)	FESS	2.8E+4	4.7E+4	2.4E+5	–	4.7E+5	1.1E+4	3.1E+4	2.3E+5	–	4.4E+4	3.0E+4	2.5E+4	2.8E+4

TABLE VII

EXPERIMENTAL RESULTS OF THE 100-DIMENSIONAL PROBLEMS f_1–f_{13}, AVERAGED OVER 50 INDEPENDENT RUNS. NOTE THAT THE BEST AND THE SECOND BEST AMONG JADE AND OTHER COMPETITIVE ALGORITHMS (EXCEPT JADE TWO VARIANTS) ARE MARKED IN BOLDFACE AND ITALIC, RESPECTIVELY

Functions		f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}
JADE	SR	100	100	100	0	86	84	100	100	100	100	96	100	100
(w/o archive)	FESS	*2.0E+5*	*2.9E+5*	*1.3E+6*	–	*2.2E+6*	*8.8E+4*	*2.3E+5*	*1.3E+6*	*1.5E+6*	*2.9E+5*	*1.9E+5*	*1.6E+5*	*1.9E+5*
JADE	SR	100	100	100	100	90	100	100	100	100	100	98	100	100
(with archive)	FESS	**1.6E+5**	**2.7E+5**	**9.6E+5**	**7.7E+5**	**1.5E+6**	**6.2E+4**	**2.0E+5**	1.4E+6	*1.5E+6*	**2.4E+5**	**1.7E+5**	**1.4E+5**	**1.6E+5**
jDE	SR	100	100	0	100	2	100	96	100	100	100	100	100	100
	FESS	5.5E+5	7.6E+5	–	*5.7E+6*	7.7E+6	2.2E+5	2.0E+6	**9.5E+5**	**1.4E+6**	8.0E+5	5.4E+5	5.7E+5	5.8E+5
SaDE	SR	36	0	100	0	0	100	54	100	100	0	100	100	100
	FESS	7.8E+5	–	2.1E+6	–	–	2.6E+5	1.6E+6	1.6E+6	1.8E+6	–	8.0E+5	9.2E+5	9.2E+5
DE/rand	SR	0	0	0	0	0	0	0	0	0	0	0	0	0
/1/bin	FESS	–	–	–	–	–	–	–	–	–	–	–	–	–
PSO	SR	100	10	0	0	0	2	70	0	0	2	36	28	54
	FESS	6.2E+5	1.1E+6	–	–	–	5.0E+5	1.9E+6	–	–	1.1E+6	6.0E+5	9.5E+5	8.7E+5
rand-JADE	SR	0	0	0	0	0	74	0	100	100	0	0	0	0
(w/o archive)	FESS	–	–	–	–	–	5.8E+5	–	2.5E+6	2.5E+6	–	–	–	–
nona-JADE	SR	100	100	0	0	90	100	100	0	0	100	100	100	100
(w/o archive)	FESS	2.0E+5	3.3E+5	–	–	5.0E+6	7.3E+4	3.5E+5	–	–	3.0E+5	2.0E+5	1.7E+5	1.9E+5

values and thus improve the progress rate of "DE/current-to-$p$best" in the promising direction. Thus, the latter effect of diversity increase becomes capable of balancing the former effect and the convergence performance of the algorithm is improved.

C. Evolution of μ_F and μ_{CR} in JADE

In classic DE, the two control parameters F and CR need to be tuned by trial and error for different optimization functions. In JADE, they are controlled by the adaptive parameters μ_F and μ_{CR} which evolve as the algorithm proceeds. The evolution of μ_F and μ_{CR} is plotted in Fig. 4 with mean curves and error bars. The error bars imply a clear evolution trend of μ_F and μ_{CR} over 50 independent runs. For example, μ_F and μ_{CR} change little in the optimization of spherical function f_1. This is reasonable because the shape of the landscape is the same at different evolution stages. For the ellipsoid function f_3, μ_F and μ_{CR} go to steady states after obvious initial changes (i.e., the population, which is initially distributed in

TABLE VIII

EXPERIMENTAL RESULTS OF THE DIXON-SZEGÖ FUNCTIONS $f_{14}-f_{20}$, AVERAGED OVER 50 INDEPENDENT RUNS

	Gen	JADE w/o archive Mean (Std Dev)	JADE with archive Mean (Std Dev)	jDE Mean (Std Dev)	SaDE Mean (Std Dev)	DE/rand/1/bin Mean (Std Dev)	PSO Mean (Std Dev)
f_{14}	200	0.397887 0.0E+00)	0.397887 (0.0E+00)	0.397887 (0.00E+00)	0.397887 (1.4E–13)	0.397887(0.0E+00)	0.397887 0.0E+00)
f_{15}	200	3.00000 (1.1E–15)	3.00000 (5.4E–16)	3.00000 (3.44E–16)	3.00000 (3.0E–15)	3.00000 (6.4E–16)	3.00000 (1.3E–15)
f_{16}	200	–3.86278 (0.0E+00)	–3.86278 (1.3E–16)	–3.86278 (0.00E+00)	–3.86278 (3.1E–15)	–3.86278 (0.0E+00)	–3.86278 (6.2E–17)
f_{17}	200	–3.31044 (3.6E–02)	–3.30806 (3.9E–02)	–3.27707 (5.79E–02)	–3.31425 (2.8E–02)	–3.24608 (5.7E–02)	–3.25800 (5.9E–02)
f_{18}	200	–10.1532 (4.0E–14)	–9.54691 (1.6E+00)	–10.1532 (1.60E–12)	–10.0522(7.1E–01)	–10.1532 (4.2E–16)	–5.79196 (3.3E+00)
f_{19}	200	–10.4029 (9.4E–16)	–10.4029 (2.4E–12)	–10.4029 (2.77E–15)	–10.4029 (6.6E–11)	–10.4029 (2.5E–16)	–7.14128 (3.5E+00)
f_{20}	200	–10.5364 (8.1E–12)	–10.5364 (6.1E–14)	–10.5364 (6.11E–16)	–10.5363 (9.2E–04)	–10.5364 (7.1E–16)	–7.02213 (3.6E+00)

TABLE IX

EXPERIMENTAL RESULTS OF f_1, f_3, f_5, $f_8-f_{13}(D=30)$ AND DIXON-SZEGÖ TEST FUNCTIONS $f_{18}-f_{19}$: THE RESULTS OF JADE WITH AND WITHOUT ARCHIVE ARE CALCULATED BASED ON 50 INDEPENDENT RUNS. THE RESULTS OF ADAPTIVE LEP AND BESTLEVY ARE TAKEN FROM [30, TABLE III], THOSE OF NSDE FROM [40, TABLES 1–3] AND THOSE OF JDE FROM [15, TABLE III]

	Gen	JADE w/o archive Mean (Std Dev)	JADE with archive Mean (Std Dev)	Adaptive LEP[1] Mean (Std Dev)	BestLevy[1] Mean (Std Dev)	NSDE[1,2] Mean (Std Dev)	jDE[1,3] Mean (Std Dev)
f_1	1500	**1.8E–60 (8.4E–60)**	*1.3e–54 (9.2e–54)*	6.32E–04 (7.6E–05)	6.59E–04 (6.4E–05)	7.1E–17	1.1E–28 (1.0E–28)
f_3	1500	2.8E–15 (8.2e–15)	**8.5e–22 (3.6e–21)**	0.041850 (0.059696)	30.628906(22.113122)	*7.9E–16*	0.090075 (0.080178)
f_5	1500	3.2e–01 (1.1e+00)	5.6e–01 (1.4e+00)	43.40 (31.52)	57.75 (41.60)	**5.9E–28**	*3.1E–15 (8.3E–15)*
f_8	1500	*4.7e+00 (2.3e+01)*	**2.4e+00 (1.7e+01)**	1100.3 (58.2)	670.6 (52.2)	–	–
f_9	1500	*1.4e–11 (1.0e–11)*	3.8e–11 (2.0e–11)	5.85 (2.07)	1.3E+01(2.3E+00)	–	**1.5E–15 (4.8E–15)**
f_{10}	1500	**4.4e–15 (0.0e+00)**	**4.4e–15 (0.0e+00)**	1.9E–02 (1.0E–03)	3.1E–02 (2.0E–03)	1.69E–09	7.7E–15 (1.4E–15)
f_{11}	1500	**0.0e+00 (0.0e+00)**	2.0e–04 (1.4e–03)	2.4E–02 (2.8E–02)	1.8E–02 (1.7E–02)	5.8E–16	**0 (0)**
f_{12}	1500	**1.6E–32 (5.5E–48)**	**1.6e–32 (5.5e–48)**	6.0E–06 (1.0E–06)	3.0E–05 (4.0E–06)	5.4E–18	6.6E–30 (7.9E–30)
f_{13}	1500	*1.4E–32 (1.1E–47)*	**1.3e–32 (1.1e–47)**	9.8E–05 (1.2E–05)	2.6E–04 (3.0E–05)	6.4E–17	5.0E–29 (3.9E–29)

[1]In this paper, we define f_8 as a shifted version of that in [15], [30], and [40] so that its optimal value is equal to 0. [2]Only the mean results are reported for NSDE in [40]. [3]The results of jDE reported in [15] are slightly different from but consistent with those in Table IV obtained in our independent experiments.

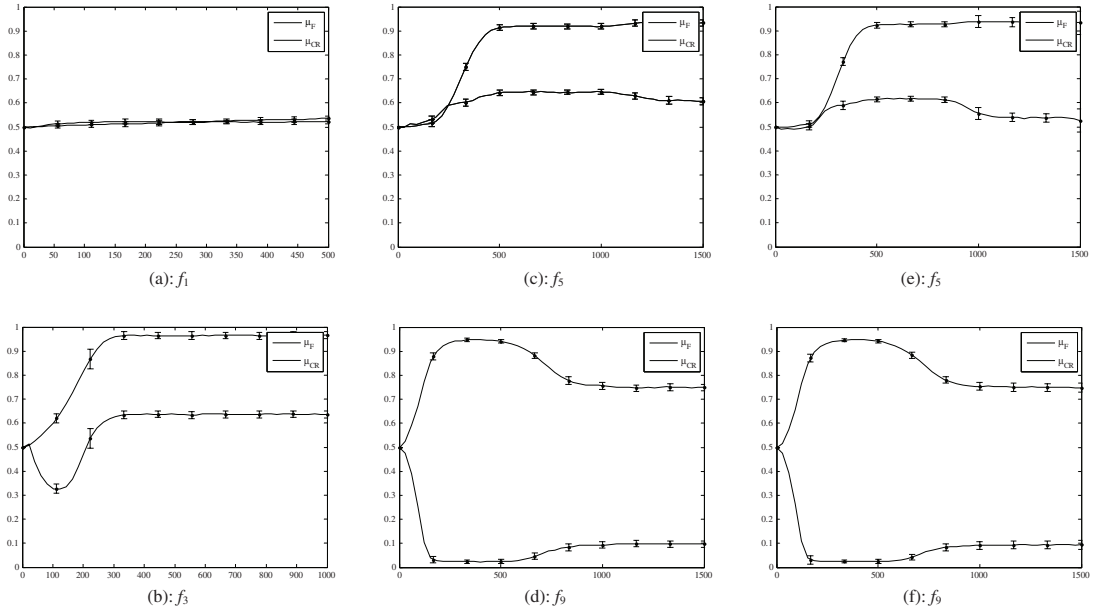

Fig. 4. Evolutions of μ_F and μ_{CR} in JADE in the optimization of f_1, f_3, f_5, and $f_9(D=30)$. (a)–(d): JADE without archive. (e, f): JADE with archive. The horizontal axis is the number of generations. The curve shows the mean and the standard error of μ_F and μ_{CR} calculated based on 50 independent runs.

a squared region, gradually adapts to the ellipsoid landscape). It is similar to the situation of f_5, where μ_F and μ_{CR} reach steady states after the population falls into the narrow valley of the Rosenbrock function. For f_9, the landscape shows different shapes as the algorithm proceeds and thus μ_F and μ_{CR} evolve to different values accordingly.

These observations are consistent with intuition; i.e., there is no fixed parameter setting of F or CR that is suitable for

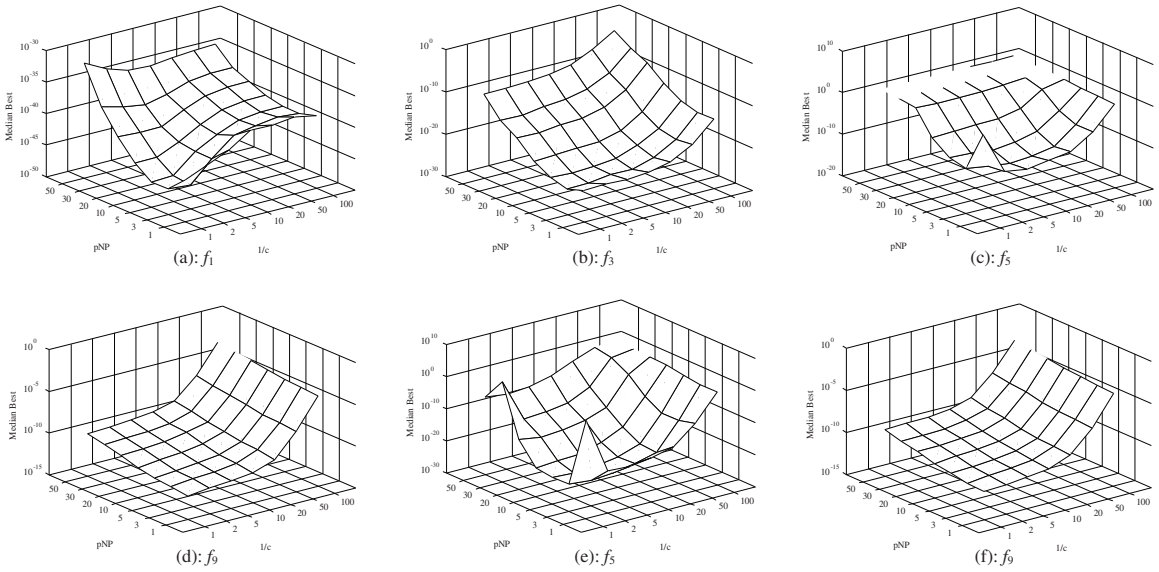

Fig. 5. Median of best-so-far function values of f_1, f_3, f_5, and $f_9(D = 30)$ after 1000, 2000, 1500, 1500 generations, respectively. (a)–(d): JADE without archive. (e, f): JADE with archive. Each data point is obtained based on 50 independent runs.

various problems (even for linearly transformable problems such as the spherical function f_1 and the ellipsoid function f_3) or at different evolution stages of a single problem (e.g., f_9).

We further investigate the effect of the initial values of μ_F and μ_{CR} on the parameter adaptation process. According to extensive experimental studies (results are not reported due to space limitation), the initial value of μ_F has little effect on the performance of the algorithm, while a moderate to large initial value of μ_{CR} is recommended especially for nonseparable functions. In general, an initial setting of $\mu_{CR} = \mu_F = 0.5$ works well for all the test functions and thus is considered as a standard setting of JADE.

D. Parameter Values of JADE

JADE introduces its own parameters c and p which determine the adaptation rates of μ_{CR} and μ_F and the greediness of the mutation strategy, respectively. It is believed that these two parameters are problem insensitive according to their roles in JADE; it is thus an advantage over the problem-dependent parameter selection (F and CR) in the classic DE. However, it is still interesting to find a range of these parameters which is appropriate for different problems. As shown in Fig. 5, the median best values are plotted for JADE with different parameter combinations: $1/c \in \{1, 2, 5, 10, 20, 50, 100\}$ and $p \cdot NP \in \{1, 3, 5, 10, 20, 30, 50\}$ with $NP = 100$. As expected, a small value of $1/c$ (e.g., $1/c = 1$) or p (e.g., $p \cdot NP = 1$) may lead to less satisfactory results in some cases. The former causes false convergence due to the lack of sufficient information to smoothly update CR and F, while the latter is too greedy to maintain the diversity of the population. However, it is clear that JADE is better than or competitive to other algorithms in a large range of c and p (as compared to the results of other algorithms in Figs. 2 and 3). Specifically, it shows that JADE (with or without archive) works best with values $1/c \in [5, 20]$ and $p \in [5\%, 20\%]$.

VI. Conclusion

Parameter adaptation is beneficial to improve the optimization performance of an evolutionary algorithm by automatically updating control parameters to appropriate values during the evolutionary search. It is natural to incorporate parameter adaptation and a greedy mutation strategy such as "DE/current-to-$p$best" to increase the convergence rate while maintaining the reliability of the algorithm at a high level. This motivated our consideration of JADE, a parameter adaptive "current-to-$p$best" DE algorithm. In "current-to-$p$best," we utilized the information of multiple best solutions to balance the greediness of the mutation and the diversity of the population. The parameter adaptation of JADE was implemented by evolving the mutation factors and crossover probabilities based on their historical record of success. We also introduced an external archive to store recently explored inferior solutions and their difference from the current population is utilized as promising directions toward the optimum.

JADE has been tested on a set of classic benchmark functions taken from the literature. It shows better or at least competitive optimization performance in terms of the convergence rate and the reliability, compared to other classic and adaptive DE algorithms and the canonical PSO algorithm. JADE also shows superiority over other evolutionary algorithms from the literature according to their reported results.

JADE with archive has shown promising results in optimizing high dimensional problems. Compared to other techniques such as cooperative coevolution [41], [42], JADE has a quite different emphasis on improving the convergence performance for high dimensional problems. It is expected that JADE can serve as an appropriate underlying scheme where cooperative coevolution is incorporated to achieve better performance [43], [44]. In addition, it is interesting to extend the current work to MOO, as parameter adaptive MOO has been commented as "a very interesting topic that very

few researchers have addressed in the specialized literature" [45]. While our initial studies have shown promising results [46], there are still many open questions in incorporating parameter adaptation schemes to multiobjective evolutionary optimization.

REFERENCES

[1] R. Storn and K. Price, "Differential evolution a simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[2] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, 1st ed. New York: Springer-Verlag, Dec. 2005.

[3] R. Joshi and A. C. Sanderson, "Minimal representation multisensor fusion using differential evolution," *IEEE Trans. Syst., Man Cybern. Part A*, vol. 29, no. 1, pp. 63–76, Jan. 1999.

[4] J. Zhang, V. Avasarala, and R. Subbu, "Evolutionary optimization of transition probability matrices for credit decision-making," *Eur. J. Oper. Res.*, to be published.

[5] J. Zhang, V. Avasarala, A. C. Sanderson, and T. Mullen, "Differential evolution for discrete optimization: An experimental study on combinatorial auction problems," in *Proc. IEEE World Congr. Comput. Intell.*, Hong Kong, China, Jun. 2008, pp. 2794–2800.

[6] R. Gamperle, S. D. Muller, and P. Koumoutsakos, "A parameter study for differential evolution," in *Proc. Advances Intell. Syst., Fuzzy Syst., Evol. Comput.*, Crete, Greece, 2002, pp. 293–298.

[7] J. Zhang and A. C. Sanderson, "An approximate Gaussian model of differential evolution with spherical fitness functions," in *Proc. IEEE Congr. Evol. Comput.*, Singapore, Sep. 2007, pp. 2220–2228.

[8] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello, "A comparative study of differential evolution variants for global optimization," in *Proc. Genetic Evol. Comput. Conf.*, Seattle, WA, Jul. 2006, pp. 485–492.

[9] H. A. Abbass, "The self-adaptive pareto differential evolution algorithm," in *Proc. IEEE Congr. Evol. Comput.*, vol. 1. Honolulu, HI, May 2002, pp. 831–836.

[10] J. Teo, "Exploring dynamic self-adaptive populations in differential evolution," *Soft Comput.: Fusion Found., Methodologies Applicat.*, vol. 10, no. 8, pp. 673–686, 2006.

[11] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Comput.: Fusion Found., Methodologies Applicat.*, vol. 9, no. 6, pp. 448–462, 2005.

[12] F. Xue, A. C. Sanderson, P. P. Bonissone, and R. J. Graves, "Fuzzy logic controlled multiobjective differential evolution," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Reno, NV, Jun. 2005, pp. 720–725.

[13] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Proc. IEEE Congr. Evol. Comput.*, vol. 2. Sep. 2005, pp. 1785–1791.

[14] V. L. Huang, A. K. Qin, and P. N. Suganthan, "Self-adaptive differential evolution algorithm for constrained real-parameter optimization," in *Proc. IEEE Congr. Evol. Comput.*, Jul. 2006, pp. 17–24.

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

[16] J. Brest, V. Zumer, and M. S. Maucec, "Self-adaptive differential evolution algorithm in constrained real-parameter optimization," in *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Jul. 2006, pp. 215–222.

[17] J. Brest, B. Boskovic, S. Greiner, V. Zumer, and M. S. Maucec, "Performance comparison of self-adaptive and adaptive differential evolution algorithms," *Soft Comput.: Fusion Found., Methodologies Applicat.*, vol. 11, no. 7, pp. 617–629, 2007.

[18] Z. Yang, K. Tang, and X. Yao, "Self-adaptive differential evolution with neighborhood search," in *Proc. IEEE Congr. Evol. Comput.*, Hong Kong, China, Jun. 2008, pp. 1110–1116.

[19] P. J. Angeline, "Adaptive and self-adaptive evolutionary computations," in *Computational Intelligence: A Dynamic Systems Perspective*. 1995, pp. 152–163.

[20] A. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 124–141, Jul. 1999.

[21] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing. Natural Computing*. New York: Springer, 2003.

[22] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: The University of Michigan Press, 1975.

[23] T. Back, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. New York: Oxford University Press, 1996.

[24] R. Mendes, I. Rocha, E. C. Ferreira, and M. Rocha, "A comparison of algorithms for the optimization of fermentation processes," in *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Jul. 2006, pp. 2018–2025.

[25] R. C. Eberhart, Y. Shi, and J. Kennedy, *Swarm Intelligence*. 1st ed., San Mateo, CA: Morgan Kaufmann, Mar. 2001.

[26] J. Zhang and A. C. Sanderson, "JADE: Self-adaptive differential evolution with fast and reliable convergence performance," in *Proc. IEEE Congr. Evol. Comput.*, Singapore, Sep. 2007, pp. 2251–2258.

[27] J. Tvrdik, I. Krivy, and L. Misik, "Evolutionary algorithm with competing heuristics," in *Proc. MENDEL 2001, Int. Conf. Soft Computing*, Brno, Czech, Jun. 2001, pp. 58–64.

[28] J. Tvrdík, L. Misik, and I. Krivy. "Competing heuristics in evolutionary algorithms," *Intell. Technol. Theory Applicat.*, pp. 159–165, 2002.

[29] 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.

[30] C. Y. Lee and X. Yao, "Evolutionary programming using mutations based on the Lévy probability distribution," *IEEE Trans. Evol. Comput.*, vol. 8, no. 1, pp. 1–13, Feb. 2004.

[31] B. V. Babu and M. M. L. Jehan, "Differential evolution for multiobjective optimization," in *Proc. IEEE Congr. Evol. Comput.*, Dec. 2003, pp. 2696–2703.

[32] U. Pahner and K. Hameyer, "Adaptive coupling of differential evolution and multiquadrics approximation for the tuning of the optimization process," *IEEE Trans. Magnetics*, vol. 36, no. 4, pp. 1047–1051, Jul. 2000.

[33] E. Mezura-Montes, J. Velazquez-Reyes, and C. A. Coello Coello, "Modified differential evolution for constrained optimization," in *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Jul. 2006, pp. 25–32.

[34] H. G. Beyer, *Theory of Evolution Strategies*. New York: Springer-Verlag, Apr. 2001.

[35] X. Yao, Y. Liu, K.-H. Liang, and G. Lin. "Fast evolutionary algorithms," in *Proc. Advances Evol. Computing: Theory Applicat.*, New York, 2003, pp. 45–94.

[36] L. C. W. Dixon and G. Szegö, "The global optimization problem: An introduction," in *Proc. Toward Global Optimization 2*, Amsterdam, Netherlands: North-Holland, 1978, pp. 1–15.

[37] I. C. Trelea, "The particle swarm optimization algorithm: Convergence analysis and parameter selection," *Inform. Process. Lett.*, vol. 85, no. 6, pp. 317–325, 2003.

[38] J. Vesterstroem and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2004, pp. 1980–1987.

[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] Z. Yang, J. He, and X. Yao, "Making a difference to differential evolution," in *Proc. Advances Metaheuristics Hard Optimization*, Dec. 2007, pp. 397–414.

[41] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Inform. Sci.*, vol. 178, no. 15, pp. 2985–2999, Feb. 2008.

[42] O. Olorunda and A. P. Engelbrecht, "Differential evolution in high-dimensional search spaces," in *Proc. 2007 IEEE Congr. Evol. Comput.*, Singapore, Sep. 2007, pp. 1934–1941.

[43] J. Zhang and A. C. Sanderson, "Adaptive differential evolution-A robust approach to multimodel problem optimization," in *Series of Adaptation, Learning, and Optimization*, New York: Springer-Verlag, Aug. 2009.

[44] Z. Yang, J. Zhang, K. Tang, X. Yao, and A. C. Sanderson, "An adaptive coevolutionary differential evolution algorithm for large-scale optimization," in *Proc. IEEE Congr. Evol. Comput.*, May 2009, pp. 102–109.

[45] C. A. Coello Coello, "20 years of evolutionary multiobjective optimization: What has been done and what remains to be done," in *Computational Intelligence: Principles and Practice*, Los Alamitos, CA: IEEE Computational Intelligence Society, 2006, pp. 73–88.

[46] J. Zhang and A. C. Sanderson, "Self-adaptive multiobjective differential evolution with direction information provided by archived inferior solutions," in *Proc. IEEE World Congr. Evol. Comput.*, Hong Kong, China, Jun. 2008, pp. 2801–2810.

Jingqiao Zhang received the B.S. degree from Beijing University of Posts and Telecommunications in 2000, the M.S. degree from Tsinghua University, China, in 2003, and the Ph.D. degree from Rensselaer Polytechnic Institute, Troy, New York, in 2008, all in electrical engineering.

From June 2007 to May 2008, he served as an Intern Researcher in the Industrial Artificial Intelligence Lab, General Electric Global Research Center, Niskayuna, New York. He is currently with the Department of Computer and System Engineering, Rensselaer Polytechnic Institute. His research interests include evolutionary optimization, mathematical programming, machine learning, artificial intelligence, and real-world applications in large-scale complex systems. He has published over 20 papers in journals and conference proceedings.

Dr. Zhang won the Competition Program Award of Scale-Invariant Optimization at the 2008 World Congress of Computational Intelligence.

Arthur C. Sanderson received the B.S. degree from Brown University, Providence, RI, in 1968 and the M.S. and Ph.D. degrees from Carnegie Mellon University, Pittsburg, PA, in 1970 and 1972, respectively.

He has held faculty positions at Carnegie Mellon University from 1973 to 1987, where he was Co-Director of the Robotics Institute, and has held a number of international visiting positions in Europe, Latin America, and Asia. In 1987, he joined Rensselaer Polytechnic Institute, Troy, New York, as Professor and served as Department Head of the Electrical, Computer and Systems Engineering Department. From 1998–2000, he was Division Director of Electrical and Communications Systems Research at the National Science Foundation in Arlington. From 2000 through July 2004, he served as Vice President for Research at Rensselaer Polytechnic Institute. He is the author of over 250 publications and proceedings and four books in the areas of biomedical signal processing, robotics and automation systems, sensor-based control, and computer vision. In 2005, he was on sabbatical leave conducting research on underwater robotics and distributed sensor networks and their application to environmental observation and monitoring. He has been a Visiting Distinguished Professor in the College of Engineering, University of South Florida, and a Visiting Research Fellow of the Autonomous Undersea Systems Institute.

Dr. Sanderson was instrumental in establishing the IEEE Robotics and Automation Society in 1989, and served as its first President. He is a Fellow of the AAAS.