Hindawi Publishing Corporation
Computational Intelligence and Neuroscience
Volume 2015, Article ID 285730, 15 pages
http://dx.doi.org/10.1155/2015/285730

Research Article

An Enhanced Differential Evolution Algorithm Based on Multiple Mutation Strategies

Wan-li Xiang, Xue-lei Meng, Mei-qing An, Yin-zhen Li, and Ming-xia Gao

School of Traffic & Transportation, Lanzhou Jiaotong University, Lanzhou, Gansu 730070, China

Correspondence should be addressed to Wan-li Xiang; xiangwl@tju.edu.cn

Received 12 May 2015; Accepted 5 July 2015

Academic Editor: Yufeng Zheng

Differential evolution algorithm is a simple yet efficient metaheuristic for global optimization over continuous spaces. However, there is a shortcoming of premature convergence in standard DE, especially in DE/best/1/bin. In order to take advantage of direction guidance information of the best individual of DE/best/1/bin and avoid getting into local trap, based on multiple mutation strategies, an enhanced differential evolution algorithm, named EDE, is proposed in this paper. In the EDE algorithm, an initialization technique, opposition-based learning initialization for improving the initial solution quality, and a new combined mutation strategy composed of DE/current/1/bin together with DE/*p*best/bin/1 for the sake of accelerating standard DE and preventing DE from clustering around the global best individual, as well as a perturbation scheme for further avoiding premature convergence, are integrated. In addition, we also introduce two linear time-varying functions, which are used to decide which solution search equation is chosen at the phases of mutation and perturbation, respectively. Experimental results tested on twenty-five benchmark functions show that EDE is far better than the standard DE. In further comparisons, EDE is compared with other five state-of-the-art approaches and related results show that EDE is still superior to or at least equal to these methods on most of benchmark functions.

1. Introduction

Optimization problems are ubiquitous in the various areas including production, life, and scientific community. These optimization problems are usually nonlinear and nondifferentiable. Particularly, the number of their local optima may increase exponentially with the problem size. Thus, evolutionary algorithms (EAs) only needing the value information of objective functions have many more advantages and have drawn more and more attention of many researchers all over the world. In this way, a lot of researchers have developed a great number of evolutionary algorithms, such as genetic algorithms (GAs), particle swarm optimization (PSO), ant colony optimization (ACO), and differential evolution (DE) algorithm. Among them, differential evolution is one of the most powerful stochastic real-parameter optimization algorithms [1]. It was originally developed by Storn and Price [2, 3] in 1995.

Due to its simple implementation, few control parameters, and fast convergence, DE has been widely and successfully applied in function optimization problems [2–26], constrained optimization problems [27–29], multiobjective optimization problems [30], scheduling [31–33], and others [34–39].

According to the aforementioned statements, it can be seen that DE has been very successful in solving various optimization problems. As far as the type of optimization problems is concerned, more researches mainly focus on continuous function optimization. However, the convergence precision and convergence speed over function optimization are still to be improved. That is, the exploration ability and exploitation ability of DE cannot be well balanced. To overcome the shortage of imbalance of the two abilities, more and more researchers have developed a large number of DE variants. For example, Noman and Iba [11] proposed a kind of accelerated differential evolution by incorporating an adaptive local search technique. Rahnamayan et al. [13] proposed an opposition-based differential evolution (ODE for short), in which a novel opposition-based learning (OBL) technique and a generation-jumping scheme are employed.

Qin et al. [14] proposed a self-adaptive differential evolution algorithm, called SaDE, in which both trial vector generation strategies and their associated parameter values are dynamically self-adapted during the process of producing promising solutions. Zhang and Sanderson [15] proposed a novel differential evolution referred to as JADE, in which a novel self-adaptive parameters scheme and a new mutation strategy with optional archive are proposed. And these improvements made JADE achieve a very fast convergence speed and high-quality solutions. Subsequently, Gong et al. [22, 23] proposed a few enhanced DE versions based on JADE by introducing adaptive strategy selection schemes or control parameters adaption mechanisms. In summary, all these state-of-the-art DE variants have achieved better convergence performance than the traditional DE.

Unfortunately, up to now, there exists no specific DE version to substantially achieve the best solution for all optimization problems because the exploration and the exploitation often mutually contradict in reality. Hence, searching for better approaches is very necessary. In order to solve continuous optimization problems more efficiently, an enhanced differential evolution algorithm based on multiple mutation strategies, called EDE for short, is presented in this paper.

The structure of the paper is organized as follows. The standard differential evolution algorithm is described briefly in Section 2. In Section 3, an enhanced differential evolution algorithm is presented and described in detail. Subsequently, Section 4 employs a set of benchmark functions to comprehensively investigate the performance of the proposed algorithm through experimental results of these functions and comparisons with other well-known evolutionary algorithms. Finally, conclusions and further study directions are given in Section 5.

2. Differential Evolution Algorithm

Differential evolution algorithm was first proposed by Storn and Price [2, 3]. Like other evolutionary algorithms, an initialization phase is its first task. In addition, it also consists of three major operations: mutation, crossover, and selection. Meanwhile, there exist a few mutation strategies proposed in the work [3]. In order to distinguish the different DE versions with various mutation strategies or different crossover schemes, the famous notation DE/x/y/z was introduced in the literature [3], where x represents the vector to be mutated, y is the number of differential vectors used, and z denotes the crossover scheme employed. DE/rand/1/bin was applied most commonly and it was also usually considered as the canonical DE version. To be specific, the canonical DE version can be described as follows.

2.1. Initialization. At the first step, a population of NP individuals is generated randomly by the following form:

$$x_{ij} = x_j^{\min} + \left(x_j^{\max} - x_j^{\min}\right) \cdot \text{rand}\,(0,1)\,, \tag{1}$$

where $i = 1, 2, \ldots, \text{NP}$, $j = 1, 2, \ldots, D$; $x_j^{\min}$ and $x_j^{\max}$ are the lower bound and upper bounds of the parameter j, respectively. Then, the cost function of each solution is evaluated.

2.2. Mutation. Mutation strategy is very important in DE. At the step, a mutant vector v_i is generated by the following formula for each D-dimensional target vector x_i:

$$v_i = x_a + F \cdot \left(x_b - x_c\right), \tag{2}$$

where $i = 1, 2, \ldots, \text{NP}$, $a, b, c \in \{1, 2, \ldots, \text{NP}\}$ are mutually different random integer number, and they are such that $a \neq b \neq c \neq i$. The mutation scale factor F is a real and constant factor $\in [0, 2]$ which controls the amplification of the differential variation $(x_b - x_c)$ [3].

2.3. Crossover. In order to exchange information between a mutant vector v_i and the current target vector x_i, crossover operation is introduced. At this time, a trial vector $u_i = (u_{i1}, u_{i2}, \ldots, u_{iD})$ is produced by the following form:

$$u_{ij} = \begin{cases} v_{ij}, & \text{if rand}\,[0,1]_j \leq \text{Cr} \vee j == j_{\text{rand}}, \\ x_{ij}, & \text{otherwise}, \end{cases} \tag{3}$$

where $j = 1, 2, \ldots, D$, rand$[0, 1]_j$ is a random real number between $[0, 1]$, and $j_{\text{rand}} \in \{1, 2, \ldots, D\}$ is a randomly chosen index, which ensures that the trial vector u_i obtains at least one parameter from the mutant vector v_i. Crossover rate Cr is a predefined constant within the range $[0, 1]$ and it controls the fraction of parameter values copied from the mutant vector.

2.4. Selection. After crossover operation, the trial vector u_i is compared to the target vector x_i through a greedy selection mechanism. The winner is retained and it will become a member of next generation. For a minimization problem, the selection process can be described according to the following equation:

$$x_i^{\star} = \begin{cases} u_i, & \text{if } f\left(u_i\right) < f\left(x_i\right), \\ x_i, & \text{otherwise}, \end{cases} \tag{4}$$

where $f(x)$ denotes the objective of solution x and $x_i^{\star}$ is an offspring corresponding to the target vector x_i.

In a word, except for the initialization phase, the aforementioned steps will be repeated in turn until a stopping criterion is reached.

3. An Enhanced Differential Evolution Algorithm

3.1. Initialization Based on Opposition-Based Learning. Recently, Rahnamayan et al. [12, 13] proposed a new scheme for generating random numbers, called opposition-based learning (OBL), which can effectively make use of random numbers and their opposites. Moreover, the ability of OBL accelerating the optimization, search, or learning process in many soft computing techniques has been reported in

(1) **for** $i = 1$ to NP **do**
(2) **for** $j = 1$ to D **do**
(3) $x_{i,j} = x_j^{\min} + \text{rand}(0,1) \cdot \left(x_j^{\max} - x_j^{\min}\right)$
(4) $ox_{i,j} = x_j^{\min} + x_j^{\max} - x_{i,j}$ //opposition-based learning
(5) **end for**
(6) **end for**

Algorithm 1: Initialization based on opposition-based learning.

the literatures [12, 13]. At first, a state-of-the-art algorithm, named ODE, was proposed by applying the OBL scheme to accelerate DE [13]. After that, the OBL scheme has been successfully used in other evolutionary algorithms such as artificial bee colony algorithm [40], harmony search algorithm [41], particle swarm optimization [42, 43], and teaching learning based algorithm [44]. A comprehensive survey about the OBL scheme can be found in [45].

In order to improve the solution quality of initial population, the OBL scheme is employed to initialize the population individuals of EDE in the work. The initial process can be described as shown in Algorithm 1. In Algorithm 1, two sets, that is, sets X and OX, are generated, where $X = \{x_1, x_2, \ldots, x_{\text{NP}}\}$ and $OX = \{ox_1, ox_2, \ldots, ox_{\text{NP}}\}$. The initial population consists of the top NP individuals chosen from the set $X \cup OX$ according to their fitness values.

3.2. Multiple Mutation Strategies. A mutation strategy DE/current/1/bin is employed. Namely, the target vector x_i is employed as the base vector in this DE version. That is, a mutant vector v_i will be generated by the following equation:

$$v_i = x_i + F \cdot \left(x_a - x_b\right), \tag{5}$$

where $i \in \{1, 2, \ldots, \text{NP}\}$ represents the index of current individual, $a \in \{1, 2, \ldots, \text{NP}\}$ and $b \in \{1, 2, \ldots, \text{NP}\}$ are random integers, and $a \neq b \neq i$.

In order to better take advantage of the guiding information of best individual, a new version of DE/best/1/bin, DE/$p$best/1/bin, proposed by Zhang and Sanderson [15], is further employed in the work to speed up the convergence speed of the proposed approach EDE. That is, a mutant vector v_i is produced as follows:

$$v_i = x_{\text{best}}^p + F \cdot \left(x_a - x_b\right), \tag{6}$$

where $p \in \{1, 2, \ldots, M\} \subseteq \{1, 2, \ldots, \text{NP}\}$ is a random number and it denotes the top p individuals according to the fitness values of individuals. It should be noted that p of DE/$p$best/1/bin in JADE [15] is a proportional number between $[0, 1]$.

More specifically, according to the first mutation strategy, it can be seen that new generated mutant vectors will be scattered around the respective target vectors, which can not only keep good population diversity but also avoid the overrandomness of classic mutation strategy DE/rand/1/bin. According to the second mutation strategy DE/$p$best/1/bin, owing to the guidance of one of several better individuals (x_{best}^p) rather than the only best individual x_{best}, the used mutation strategy can drive population towards better individuals so as to enhance the convergence speed. In addition, it can also prevent EDE from congregating the vicinity of global best individual to some extent.

In the meantime, a probabilistic parameter r_1 is time varying and designed to control which of the two mutation strategies is to be executed at the mutation step. The parameter r_1 can be described as follows:

$$r_1 = r_{\max} - \frac{\text{FEs}}{\max \text{FEs}} \cdot \left(r_{\max} - r_{\min}\right), \tag{7}$$

where $r_{\max}$ and $r_{\min}$ denote the maximum probability value and the minimum probability value, respectively. FEs is an iterative variable. max FEs represents the maximum number of fitness function evaluations.

As a matter of fact, the probability parameter r_1 plays an important role in balancing the exploration ability and the exploitation ability. That is, it is hoped that good population diversity is kept at the beginning of evolution and fast convergence speed is achieved at the end of search.

3.3. Perturbation. After repeating all operations (mutation, crossover, and select operations) of differential evolution, a perturbed scheme is conducted over the best individual in order to further trade off the searching ability of the aforementioned solution search equations. During the process, two perturbed equations are introduced and the best individual is perturbed dimension by dimension according to them, which are described by (8) and (9), respectively. One has

$$\mu_j = x_{\text{best},n} + (2 \cdot \text{rand}\,(0,1) - 1) \cdot \left(x_{\text{best},n} - x_{k,n}\right), \tag{8}$$

where $j = 1, 2, \ldots, D$, $\mu = (\mu_1, \mu_2, \ldots, \mu_D)$ is a temporary copy of the best individual, best represents the index of best individual in current population, $k \in \{1, 2, \ldots, \text{NP}\} \wedge k \neq \text{best}$ is a uniform random number, and $n \in \{1, 2, \ldots, D\}$ is also a random number. One has

$$\mu_j = x_{\text{best},j} + (2 \cdot \text{rand}\,(0,1) - 1) \cdot \left(x_{\text{best},n} - x_{k,n}\right), \tag{9}$$

where all the notations are the same as those in (8).

From (9), it can be observed that perturbation operation occurs on the current component j of best individual, and the differential variation $(x_{\text{best},n} - x_{k,n})$ acts as perturbed scales.

Notice that dimension n may be different from j, which is helpful to enrich perturbation scales to some extent. That is, it may increase the probability of getting out of local minima trap.

What is more, the term $x_{\text{best},j}$ of (9) is different from the first term on the right hand side of formulation (8). The reason for (8) introduced is that information between different dimensions of best individual could be shared. Thus, the EDE algorithm could get out of local optimal trap with a larger probability.

Like the aforementioned tradeoff scheme, a probability parameter r_2 is employed. The parameter r_2 is linear time-varying during the evolution process as follows:

$$r_2 = w_{\min} + \frac{\text{FEs}}{\max \text{FEs}} \cdot (w_{\max} - w_{\min}), \tag{10}$$

where $w_{\max}$ and $w_{\min}$ denote the maximum probability value and the minimum probability value, respectively. The rest of these parameters are the same as those in (7).

Concretely speaking, (8) is executed with a probability value r_2, but (9) is executed with a probability value $(1 - r_2)$.

3.4. Boundary Constraints Handling Technique. In order to keep solutions subject to boundary constraints, some components of a solution violating the predefined boundary constraints should be repaired. That is, if a parameter value produced by solution search equations exceeds its predefined boundaries, the parameter should be set to an acceptable value. The following repair rule used in the literature [17] is employed in this work:

$$x_{ij} = \begin{cases} x_j^{\min} + \text{rand}(0,1) \cdot \left(x_j^{\max} - x_j^{\min}\right), & \text{if } x_{ij} < x_j^{\min}, \\ x_j^{\max} - \text{rand}(0,1) \cdot \left(x_j^{\max} - x_j^{\min}\right), & \text{if } x_{ij} > x_j^{\max}. \end{cases} \tag{11}$$

3.5. The Proposed Approach. In order to effectively take use of the guidance information of best individual, mutation strategy DE/best/1/bin is considered. In order to prevent a large number of individuals from clustering around the global best individual, inspired by JADE [15], mutation strategy DE/$p$best/1/bin is actually used. In addition, another mutation strategy DE/current/1/bin is employed to further trade off the exploitation ability of DE/$p$best/1/bin. At the same time, a selective probability r_1 with linear time-varying nature is introduced to decide which mutation strategy works at the mutation phase of DE. Subsequently, a perturbation scheme for the best individual is incorporated into the modified DE version. In short, the pseudocode of EDE can be given in Algorithm 2 based on the above explanation.

4. Experimental Study and Discussion

4.1. Benchmark Functions and Parameter Settings. To verify the optimization effectiveness of EDE, twenty-five benchmark functions with different characteristics taken from Yao et al. [46], Gong et al. [23], and Gao and Liu [40] are employed here.

These benchmark functions are listed briefly in Table 1, in which D designates the dimensionality of test functions. All the functions are scalable and high-dimensional problems. Functions f_{01}–f_{05}, f_{14}, and f_{15} are unimodal. Function f_{06}, that is, the step function, has one minimum and is discontinuous. Function f_{07} is a quartic function with noise. Functions f_{08}–f_{13} and f_{16}–f_{19} are difficult multimodal functions where the number of local minima increases exponentially as the dimension of test function increases. In addition, six shifted functions are chosen to evaluate the performance of EDE. Namely, functions f_{20}–f_{25} are shifted functions and $o = (o_1, o_2, \ldots, o_D)$ representing a shifted vector is generated randomly in the corresponding search range.

In our experimental study, all benchmark functions are tested in 30 dimensions and 100 dimensions. The corresponding maximum number of fitness function evaluations (max FEs) is $15e4$ and $50e4$, respectively. Moreover, the other specific parameters of DE and EDE are set as follows.

DE Settings. In canonical DE/rand/1/bin, the scale factor F is set to 0.5, the parameter of crossover rate Cr is set to 0.9, and the population size SN is 100. It should be noted that the values of three parameters are the same as those of the state-of-the-art algorithm ODE [13].

EDE Settings. In our proposed algorithm, the scale factor F is set to 0.5. The parameter of crossover rate Cr is set to 0.9. And the population size SN is 20. A few other parameters are set as follows: $r_{\max} = 1$, $r_{\min} = 0.1$, $w_{\max} = 0.2$, $w_{\min} = 0$, and $M = 4$.

For the set of experiments tested on 25 benchmark functions, we use the aforementioned parameter settings unless a change is mentioned. Furthermore, each test case is optimized thirty runs independently. Then, experimental results for these well-known problems as well as some comparisons with other famous methods are reported as follows.

4.2. Comparison between DE and EDE. For the purpose of validating the enhancing effectiveness of EDE, EDE is first compared with canonical DE in terms of best, worst, median, mean, and standard deviation (Std.) values of solutions achieved by each algorithm in 30 independent runs. The corresponding results are listed in Table 2. Furthermore, the Wilcoxon rank sum test is conducted to compare the significant difference between DE and EDE at $\alpha = 0.05$ significance level. The related test results are also reported in Table 2. And then, some representatives of convergence curves of DE and EDE are shown in Figure 1 in order to show the convergence speed of EDE more clearly.

From Table 2, it can be seen that EDE is significantly superior to DE in most cases. To be specific, EDE is significantly better than DE on 20 functions, that is, f_{01}, f_{02}, f_{03}, f_{04}, f_{05}, f_{06}, f_{08}, f_{09}, f_{10}, f_{12}, f_{13}, f_{14}, f_{15}, f_{16}, f_{19}, f_{20}, f_{21}, f_{22}, f_{24}, and f_{25}, in terms of related Wilcoxon rank sum test results. In addition, for function f_{07} with $D = 30$, EDE is still better than DE. For function f_{07} with $D = 100$, EDE is equal to DE; actually, the mean result achieved by EDE is slightly better than that of DE. For function f_{11} with $D = 30$, EDE is similar

(1) Initialize a population of NP individuals based on the opposition-based learning technique
(2) Set FEs = 2 ∗ NP // FEs represents the iteration counter
// max FEs represents the maximum number of fitness function evaluations
(3) **while** FEs ⩽ max FEs **do**
(4) **for** i = 1 to NP **do**
(5) Sort the population from best to worst
(6) Randomly choose a relatively better individual from the top $p \in \{1, 2, \ldots, M\}$ individuals, and let $p$best represent the index of chosen individual
(7) Select uniform randomly $a \neq b \neq i$
(8) **if** rand ⩽ r_1 **then**
(9) Generate a mutant individual v according to (5)
(10) **else**
(11) Generate a mutant individual v according to (6)
$v = x_{p\text{best}} + F \cdot (x_a - x_b)$
(12) **end if**
(13) Let $u = x_i$
//rand is a function for generating a random number in the range of [0, 1]
(14) Let $j$rand = $\lfloor D * \text{rand} \rfloor + 1$
(15) **for** j = 1 to D **do**
(16) **if** rand ⩽ Cr ‖ $j ==$ $j$rand **then**
(17) $u_j = v_j$
(18) **end if**
(19) **end for**
(20) Evaluate the new produced individual u
(21) Set FEs = FEs + 1
(22) **if** $f(u)$ is better than $f(x_i)$ **then**
(23) Replace x_i with u
(24) **end if**
(25) **end for**
(26) //Perturb the best individual dimension by dimension
(27) **for** j = 1 to D **do**
(28) Set $\mu = x_{\text{best}}$ // best denotes the index of best individual
(29) $k = \lfloor \text{rand} * \text{NP} \rfloor + 1 \wedge k \neq \text{best}$
(30) $n = \lfloor \text{rand} * D \rfloor + 1$
(31) **if** rand < r_2 **then**
(32) Modify μ_j according to (8)
(33) **else**
(34) Modify μ_j according to (9)
(35) **end if**
(36) Evaluate the modified individual μ
(37) Set FEs = FEs + 1
(38) Choose a better individual from the set $\{x_{\text{best}}, \mu\}$ to represent x_{best}
(39) **end for**
(40) Record the best solution found so far
(41) **end while**

Algorithm 2: The EDE algorithm.

to DE. For the function f_{11} with $D = 100$, DE outperforms EDE. Nevertheless, the results obtained by EDE are very close to those found by DE. For the functions f_{17}, f_{18} and f_{23}, DE is better than EDE. And yet, the results obtained by EDE are very close to those found by DE on the functions f_{17}, f_{18} at $D = 30$ and f_{23} at $D = 100$.

From Figure 1, it can also be observed that EDE is far better than DE in terms of solutions accuracy and convergence speed on the representative cases.

According to the aforementioned analyses, it can be concluded that EDE is better than or approximately equal to DE on almost all the functions. In other words, multiple

Table 1: Benchmark functions used in experiments.

Test functions	Search space	Optimum
$f_{01}(x) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$	0
$f_{02}(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	$[-10, 10]^D$	0
$f_{03}(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2$	$[-100, 100]^D$	0
$f_{04}(x) = \max_i \{ \lvert x_i \rvert, 1 \leqslant i \leqslant D \}$	$[-100, 100]^D$	0
$f_{05}(x) = \sum_{i=1}^{D-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + (x_i - 1)^2 \right]$	$[-30, 30]^D$	0
$f_{06}(x) = \sum_{i=1}^{D} \left(\lfloor x_i + 0.5 \rfloor \right)^2$	$[-100, 100]^D$	0
$f_{07}(x) = \sum_{i=1}^{D} i x_i^4 + \text{random}[0, 1)$	$[-1.28, 1.28]^D$	0
$f_{08}(x) = -418.98288727243369 \times D + \sum_{i=1}^{D} \left[-x_i \sin \left(\sqrt{\lvert x_i \rvert} \right) \right]$	$[-500, 500]^D$	0
$f_{09}(x) = \sum_{i=1}^{D} \left(x_i^2 - 10 \cos (2\pi x_i) + 10 \right)$	$[-5.12, 5.12]^D$	0
$f_{10}(x) = -20 \exp \left(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^{D} x_i^2} \right) - \exp \left(\frac{1}{D} \sum_{i=1}^{D} \cos (2\pi x_i) \right) + 20 + e$	$[-32, 32]^D$	0
$f_{11}(x) = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i}^{D} \cos \left(\frac{x_i}{\sqrt{i}} \right) + 1$	$[-600, 600]^D$	0
$f_{12}(x) = \frac{\pi}{D} \left\{ 10 \sin^2 (\pi y_1) + \sum_{i=1}^{D-1} (y_i - 1)^2 \left[1 + 10 \sin^2 (\pi y_{i+1}) \right] (y_n + 1)^2 \right\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4),$ where $y_i = 1 + \frac{1}{4}(x_i + 1)$, $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a; \\ 0, & -a \leqslant x_i \leqslant a; \\ k(-x_i - a)^m, & x_i < -a. \end{cases}$	$[-50, 50]^D$	0
$f_{13}(x) = 0.1 \left\{ \sin^2 (3\pi x_1) + \sum_{i=1}^{D} (x_i - 1)^2 \left[1 + \sin^2 (3\pi x_i + 1) \right] + (x_n - 1)^2 \left[1 + \sin^2 (2\pi x_n) \right] \right\} + \sum_{i=1}^{D} u(x_i, 5, 100, 4)$	$[-50, 50]^D$	0
$f_{14}(x) = \sum_{i=1}^{D} i x_i^2$	$[-10, 10]^D$	0
$f_{15}(x) = \sum_{i=1}^{D} i x_i^4$	$[-1.28, 1.28]^D$	0
$f_{16}(x) = \sum_{i=1}^{D} \left[y_i^2 - 10 \cos (2\pi y_i) + 10 \right]$, where $y_i = \begin{cases} x_i & \text{if } \lvert x_i \rvert < 0.5, \\ \frac{\text{round}(2x_i)}{2} & \text{else } \lvert x_i \rvert \geqslant 0.5. \end{cases}$	$[-5.12, 5.12]^D$	0
$f_{17}(x) = 0.5 + \frac{\sin^2 \sqrt{\sum_{i=1}^{D} x_i^2} - 0.5}{\left(1 + 0.001 \sum_{i=1}^{D} x_i^2 \right)^2}$	$[-100, 100]^D$	0
$f_{18}(x) = -\cos (2\pi \lVert x \rVert) + 0.1 \lVert x \rVert + 1$, where $\lVert x \rVert = \sqrt{\sum_{i=1}^{D} x_i^2}$	$[-100, 100]^D$	0
$f_{19}(x) = \sum_{i=1}^{D} \lvert x_i \sin (x_i) + 0.1 x_i \rvert$	$[-10, 10]^D$	0
$f_{20}(x) = \sum_{i=1}^{D} z_i^2$, $z = x - o$	$[-100, 100]^D$	0

Table 1: Continued.

Test functions	Search space	Optimum
$f_{21}(x) = \sum_{i=1}^{D} \left(z_i^2 - 10\cos(2\pi z_i) + 10\right), z = x - o$	$[-5, 5]^D$	0
$f_{22}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)\right) + 20 + e, z = x - o$	$[-32, 32]^D$	0
$f_{23}(x) = \frac{1}{4000}\sum_{i=1}^{D} z_i^2 - \prod_{i}^{D}\cos\left(\frac{z_i}{\sqrt{i}}\right) + 1, z = x - o$	$[-600, 600]^D$	0
$f_{24}(x) = \sum_{i=1}^{D}\left(\sum_{j=1}^{i} z_j\right)^2, z = x - o$	$[-100, 100]^D$	0
$f_{25}(x) = \sum_{i=1}^{D-1}\left(100\left(z_{i+1} - z_i^2\right)^2 + (z_i - 1)^2\right), z = x - o + 1$	$[-100, 100]^D$	0

mutation strategies and perturbation schemes are beneficial to the performance of EDE.

4.3. Comparison between EDE and Other Three DE Variants. In this subsection, EDE is further compared with some representatives of state-of-the-art DE variants, such as SaDE [14], JADE [15], and SaJADE [23]. Here sixteen test functions are used for the comparison. The related comparison results are listed in Table 3. For a fair comparison, except for the proposed algorithm EDE, the rest of the results reported in Table 3 are directly taken from Gong et al. [23].

From Table 3, it can be seen that EDE is obviously better than JADE on twelve functions, that is, f_{01}, f_{02}, f_{04}, f_{05}, f_{06}, f_{08}, f_{09}, f_{10}, f_{12}, f_{13}, f_{19}, and f_{21}. JADE works better than EDE on four functions. Notice that EDE is just slightly inferior to JADE on the three functions f_{03}, f_{07}, and f_{18}. When compared with SaDE, EDE performs better than it does on thirteen functions. And the results found by EDE are very close to those found by SaDE on other two functions f_{07} and f_{18}. When compared with SaJADE, SaJADE is better than EDE on four functions, but the superiority of SaJADE is not obvious on the three functions f_{05}, f_{07}, and f_{18} except for function f_{11}. Yet EDE is better than or equal to SaJADE on other twelve functions.

It should be pointed out that the results are summarized as $w/t/l$ in the last line of Table 3, which means that EDE wins in w function cases, ties in t cases, and loses in l cases when compared with its competitor. For JADE, SaDE, and SaJADE, they are 12/0/4, 13/0/3, and 11/1/4, respectively. The results show that EDE is superior to or similar to other three approaches on the majority of benchmark functions.

4.4. Comparison among EDE and Two Artificial Bee Colony Algorithms. Artificial bee colony algorithm introduced by Karaboga and Basturk is a relatively new swarm-based optimization algorithm [47]. And it has become a promising technique [48]. Particularly, a modified artificial bee colony algorithm, named MABC, proposed by Gao and Liu [40], is an outstanding representative of many enhanced ABC versions. In order to further demonstrate the superiority of EDE, EDE is compared with standard ABC and MABC on twenty-one functions again. In the experimental study, the maximum number of fitness function evaluations (max FEs) is set to 15*e*4 for all compared algorithms as recommended by Gao and Liu [40].

The further comparison results are given in Table 4. For convenience, besides the data achieved by the EDE algorithm, the rest of the results are gained by Gao and Liu [40] directly.

From Table 4, it is clear that EDE is better than or at least even with ABC on nineteen functions, but ABC only works better than EDE on two functions. EDE is better than or equal to MABC on eighteen functions. MABC also only surpasses EDE on three functions. In addition, the accuracy of solution obtained by EDE is far better than that obtained by ABC on many benchmark functions such as f_{01}, f_{02}, f_{08}, f_{14}, f_{15}, and f_{19}. Meanwhile, the accuracy of solution obtained by EDE is far better than that obtained by MABC on some test functions including f_{01}, f_{02}, and f_{14}. In summary, EDE is superior to both ABC and MABC.

5. Conclusion

In order to achieve a better compromise between the exploration ability and the exploitation ability of DE, in this work, an enhanced differential evolution algorithm, called EDE, is presented. In EDE, first, an initialization technique, opposition-based learning initialization, is employed. Next, inspired by JADE [15], a mutation strategy DE/*p*best/1/bin is introduced in EDE. At the same time, a new mutation strategy DE/current/bin/1 is also introduced. That is, there are multiple mutation strategies composed of the two mutation strategies in EDE to better balance the exploration and the exploitation of DE. When performing the EDE algorithm, one of the two mutation strategies is chosen randomly with a linear time-varying scheme. Last, a perturbation scheme for the best individual is presented in order to get out of local minima, where the perturbation scheme is also composed of two solution search equations. Specifically, the best individual is perturbed dimension by dimension in two modes. All these modifications make up the proposed algorithm EDE.

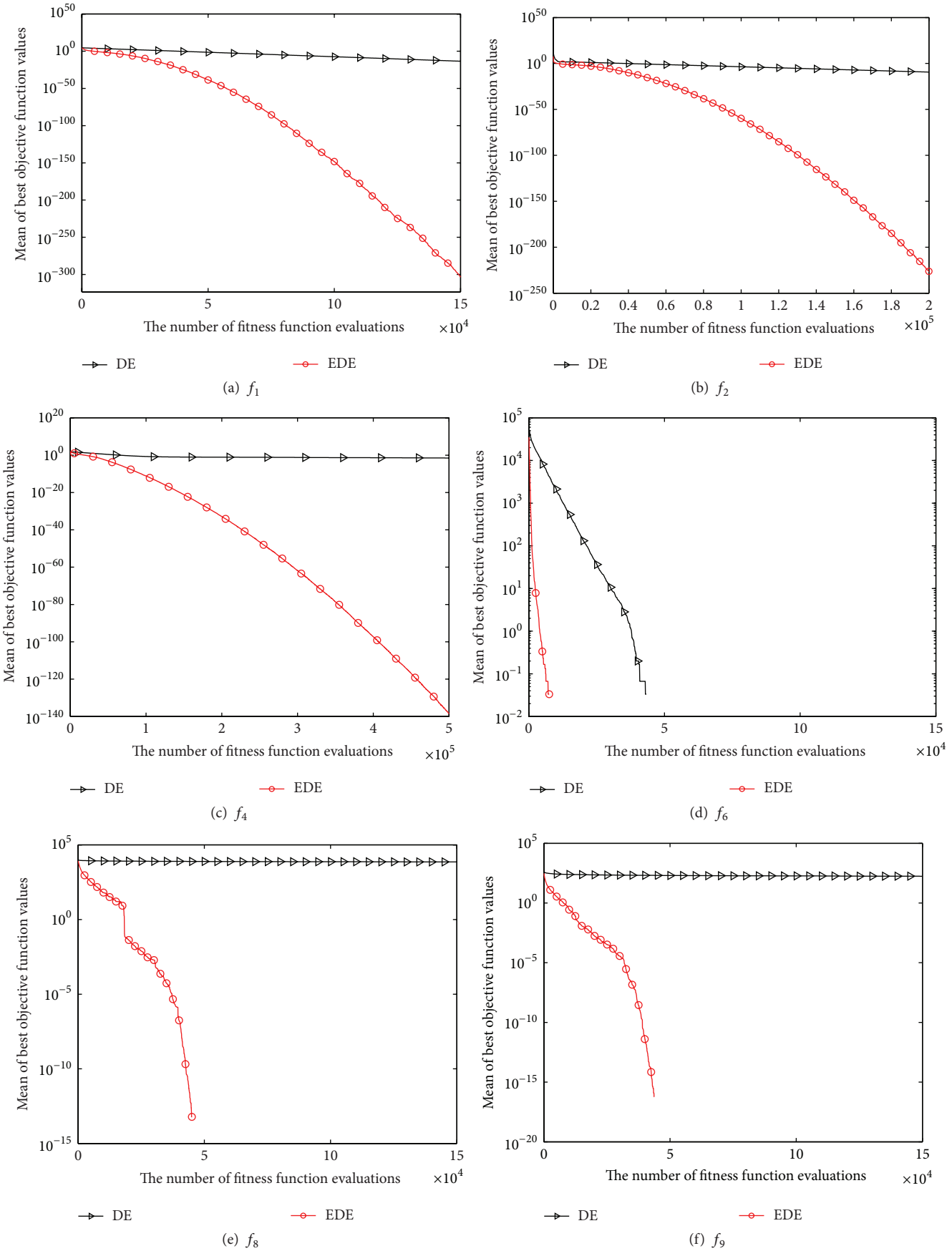

Figure 1: Continued.

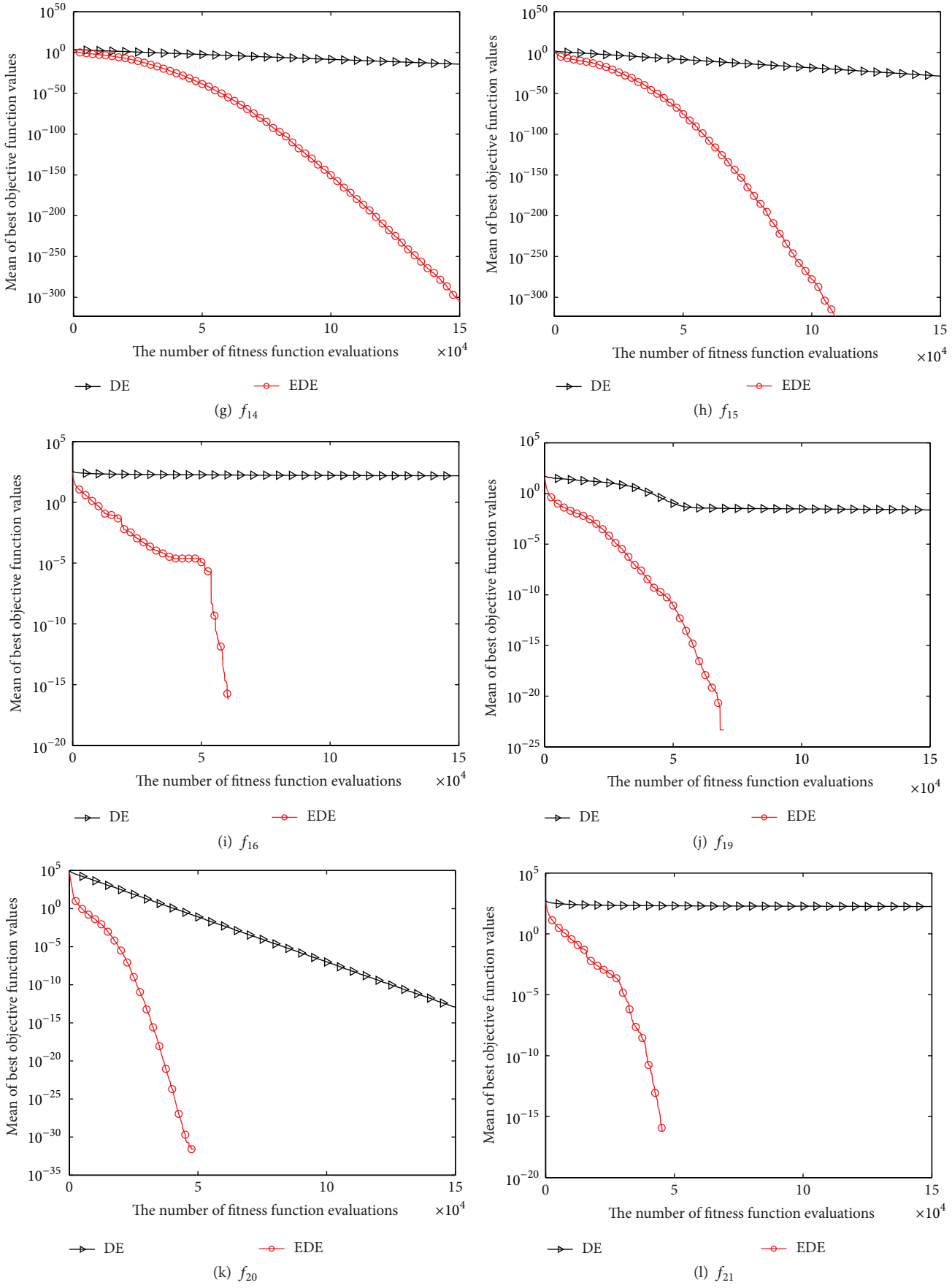

Figure 1: Convergence performance of DE and EDE on the twelve test functions at $D = 30$.

Table 2: Best, worst, median, mean, and standard deviation values achieved by DE and EDE through 30 independent runs.

Number	Dim.	max FEs	Methods	Best	Worst	Median	Mean	Std.	Sig.
f_{01}	30	15e4	DE	1.48e − 014	1.00e − 013	3.38e − 014	3.81e − 014	1.87e − 014	†
			EDE	0.00e − 000	1.13e − 302	1.12e − 315	4.19e − 304	0.00e − 000	
	100	50e4	DE	1.50e − 018	9.93e − 017	7.79e − 018	1.29e − 017	1.86e − 017	†
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
f_{02}	30	20e4	DE	1.40e − 010	9.17e − 010	3.57e − 010	3.95e − 010	1.93e − 010	†
			EDE	6.18e − 234	1.00e − 225	1.05e − 229	4.65e − 227	0.00e − 000	
	100	50e4	DE	1.39e − 010	6.55e − 010	3.61e − 010	3.71e − 010	1.30e − 010	†
			EDE	8.41e − 229	2.02e − 221	5.42e − 225	7.34e − 223	0.00e − 000	
f_{03}	30	50e4	DE	8.22e − 013	1.93e − 010	2.21e − 011	4.23e − 011	4.84e − 011	†
			EDE	7.63e − 085	7.80e − 079	3.78e − 081	8.01e − 080	1.71e − 079	
	100	50e4	DE	5.27e + 003	2.23e + 004	9.65e + 003	1.01e + 004	3.62e + 003	†
			EDE	3.64e − 004	1.18e − 002	2.50e − 003	3.70e − 003	3.20e − 003	
f_{04}	30	50e4	DE	4.40e − 012	2.50e − 001	1.00e − 003	3.15e − 002	6.32e − 002	†
			EDE	1.43e − 144	2.81e − 138	2.95e − 140	2.79e − 139	6.39e − 139	
	100	50e4	DE	1.01e + 001	2.67e + 001	1.97e + 001	1.97e + 001	3.30e − 000	†
			EDE	3.61e − 027	7.99e − 025	3.92e − 026	1.02e − 025	1.59e − 025	
f_{05}	30	15e4	DE	1.41e + 001	1.84e + 001	1.70e + 001	1.68e + 001	1.06e − 000	†
			EDE	1.52e − 024	2.43e − 001	4.03e − 015	8.50e − 003	4.43e − 002	
		50e4	DE	3.47e − 016	3.98e − 000	1.99e − 013	1.32e − 001	7.27e − 001	†
			EDE	0.00e − 000	7.28e − 027	2.17e − 029	4.42e − 028	1.38e − 027	
	100	50e4	DE	7.79e + 001	1.96e + 002	1.41e + 002	1.33e + 002	3.63e + 001	†
			EDE	1.19e − 004	1.61e + 002	7.44e + 001	5.31e + 001	4.70e + 001	
f_{06}	30	8e3	DE	1.79e + 003	5.38e + 003	4.07e + 003	3.90e + 003	8.52e + 002	†
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
		15e4	DE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
	100	5e4	DE	4.27e + 002	1.06e + 003	6.83e + 002	6.81e + 002	1.65e + 002	
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	†
		50e4	DE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
f_{07}	30	30e4	DE	2.50e − 003	8.10e − 003	4.40e − 003	4.70e − 003	1.40e − 003	†
			EDE	7.02e − 004	3.80e − 003	2.30e − 003	2.30e − 003	8.87e − 004	
	100	50e4	DE	1.80e − 002	9.09e − 002	2.87e − 002	3.25e − 002	1.34e − 003	≈
			EDE	2.29e − 002	4.25e − 002	3.03e − 002	3.05e − 002	4.90e − 003	
f_{08}	30	15e4	DE	6.56e + 003	7.72e + 003	7.26e + 003	7.26e + 003	2.91e + 002	†
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
	100	50e4	DE	2.56e + 004	3.30e + 004	2.89e + 004	2.93e + 004	1.84e + 003	†
			EDE	9.45e − 011	9.45e − 011	9.45e − 011	9.45e − 011	0.00e − 000	
f_{09}	30	15e4	DE	1.46e + 002	1.94e + 002	1.77e + 002	1.74e + 002	1.34e + 001	†
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
	100	50e4	DE	1.91e + 002	6.65e + 002	5.70e + 002	5.49e + 002	1.03e + 002	†
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
f_{10}	30	15e4	DE	2.82e − 008	2.07e − 007	5.86e − 008	7.10e − 008	3.55e − 008	†
			EDE	4.44e − 015	4.44e − 015	4.44e − 015	4.44e − 015	0.00e − 000	
	100	50e4	DE	2.06e − 010	2.79e − 009	5.90e − 010	6.86e − 010	4.79e − 010	†
			EDE	4.44e − 015	7.99e − 015	7.99e − 015	7.40e − 015	1.34e − 015	
f_{11}	30	15e4	DE	1.16e − 014	7.40e − 003	1.11e − 013	4.93e − 004	1.90e − 003	≈
			EDE	0.00e − 000	7.34e − 002	1.60e − 002	2.09e − 002	2.19e − 002	
	100	50e4	DE	0.00e − 000	5.37e − 002	0.00e − 000	3.20e − 003	1.01e − 002	-
			EDE	0.00e − 000	6.11e − 002	0.00e − 000	1.05e − 002	1.76e − 002	

Table 2: Continued.

Number	Dim.	max FEs	Methods	Best	Worst	Median	Mean	Std.	Sig.
f_{12}	30	15e4	DE	8.61e − 016	2.17e − 014	3.90e − 015	5.28e − 015	4.65e − 015	†
			EDE	1.57e − 032	1.57e − 032	1.57e − 032	1.57e − 032	5.56e − 048	
	100	50e4	DE	5.48e − 019	1.55e − 001	2.46e − 016	8.30e − 003	2.94e − 002	†
			EDE	4.71e − 033	4.71e − 033	4.71e − 033	4.71e − 033	1.39e − 048	
f_{13}	30	15e4	DE	8.68e − 015	1.06e − 013	2.61e − 014	3.55e − 014	2.46e − 014	†
			EDE	1.34e − 032	1.34e − 032	1.34e − 032	1.34e − 032	5.56e − 048	
	100	50e4	DE	2.74e − 017	2.75e + 001	9.82e − 000	1.02e + 001	6.88e − 000	†
			EDE	1.34e − 032	1.34e − 032	1.34e − 032	1.34e − 032	5.56e − 048	
f_{14}	30	15e4	DE	7.58e − 016	1.86e − 014	5.13e − 015	5.71e − 015	4.17e − 015	†
			EDE	0.00e − 000	6.48e − 304	1.00e − 313	2.19e − 305	0.00e − 000	
	100	50e4	DE	2.21e − 019	2.62e − 017	3.41e − 018	4.70e − 018	4.93e − 018	†
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
f_{15}	30	15e4	DE	5.46e − 031	1.13e − 028	5.45e − 030	1.53e − 029	2.50e − 029	†
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
	100	50e4	DE	1.58e − 027	3.78e − 024	3.00e − 026	1.97e − 025	6.84e − 025	†
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
f_{16}	30	15e4	DE	1.18e + 002	1.70e + 002	1.52e + 002	1.51e + 002	1.37e + 001	†
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
	100	50e4	DE	3.64e + 002	7.47e + 002	6.07e + 002	6.05e + 002	9.01e + 001	†
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
f_{17}	30	15e4	DE	3.72e − 002	3.72e − 002	3.72e − 002	3.72e − 002	5.43e − 007	-
			EDE	7.82e − 002	2.72e − 001	1.78e − 001	1.63e − 001	5.62e − 002	
	100	50e4	DE	7.82e − 002	1.41e − 001	7.82e − 002	9.29e − 002	2.29e − 002	-
			EDE	4.79e − 001	4.94e − 001	4.90e − 001	4.89e − 001	4.50e − 003	
f_{18}	30	15e4	DE	1.12e − 001	2.00e − 001	1.99e − 001	1.97e − 001	1.60e − 002	-
			EDE	2.99e − 001	6.99e − 001	4.99e − 001	5.03e − 001	1.09e − 001	
	100	50e4	DE	2.99e − 001	3.99e − 001	3.07e − 001	3.39e − 001	4.65e − 002	-
			EDE	2.09e − 000	3.39e − 000	2.79e − 000	2.77e − 000	3.31e − 001	
f_{19}	30	15e4	DE	1.85e − 002	2.87e − 002	2.39e − 002	2.38e − 002	2.70e − 002	†
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
	100	50e4	DE	2.74e − 010	3.27e − 008	2.84e − 009	4.88e − 009	6.52e − 009	†
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
f_{20}	30	15e4	DE	1.55e − 014	5.02e − 013	7.96e − 014	1.07e − 013	1.07e − 013	†
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
	100	50e4	DE	1.51e − 017	1.97e − 016	4.43e − 017	5.78e − 017	3.86e − 017	†
			EDE	0.00e − 000	4.93e − 032	0.00e − 000	6.57e − 033	1.70e − 032	
f_{21}	30	15e4	DE	1.49e + 002	1.95e + 002	1.79e + 002	1.77e + 002	1.18e + 001	†
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
	100	50e4	DE	2.56e + 002	7.10e + 002	6.09e + 002	5.90e + 002	1.07e + 002	†
			EDE	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	0.00e − 000	
f_{22}	30	15e4	DE	3.20e − 008	2.01e − 007	6.56e − 008	7.75e − 008	3.80e − 008	†
			EDE	4.44e − 015	7.99e − 015	7.99e − 015	7.40e − 015	1.34e − 015	
	100	50e4	DE	3.52e − 010	1.42e − 009	7.04e − 010	7.54e − 010	2.85e − 010	†
			EDE	7.99e − 015	7.99e − 015	7.99e − 015	7.99e − 015	0.00e − 000	
f_{23}	30	15e4	DE	1.17e − 013	2.35e − 012	5.77e − 013	7.79e − 013	6.66e − 013	-
			EDE	0.00e − 000	1.29e − 001	9.90e − 003	2.10e − 002	2.70e − 002	
	100	50e4	DE	0.00e − 000	9.90e − 003	0.00e − 000	9.03e − 004	2.80e − 003	-
			EDE	0.00e − 000	3.94e − 002	7.40e − 003	8.50e − 003	1.05e − 002	

Table 2: Continued.

Number	Dim.	max FEs	Methods	Best	Worst	Median	Mean	Std.	Sig.
f_{24}	30	15e4	DE	8.85e − 001	5.87e − 000	1.77e − 000	2.29e − 000	1.30e − 000	†
			EDE	1.62e − 023	2.05e − 019	5.25e − 021	2.05e − 020	4.02e − 020	
	100	50e4	DE	1.93e + 004	5.73e + 004	2.76e + 004	2.90e + 004	6.85e + 003	†
			EDE	6.45e − 004	2.96e − 003	6.20e − 003	7.40e − 003	6.60e − 003	
f_{25}	30	15e4	DE	1.64e + 001	7.61e + 001	1.84e + 001	2.03e + 001	1.05e + 001	†
			EDE	4.23e − 015	7.36e − 001	2.72e − 007	6.93e − 002	1.79e − 001	
	100	50e4	DE	7.78e + 001	3.01e + 002	1.43e + 002	1.54e + 002	5.62e + 001	†
			EDE	2.62e − 005	1.72e + 002	7.59e + 001	6.08e + 001	4.64e + 001	

† indicates that EDE is better than its competitor by the Wilcoxon rank sum test at $\alpha = 0.05$.
- means that EDE is worse than its competitor.
≈ means that there is no significant difference between DE and EDE.

Table 3: Performance comparison between EDE and other three DEs over 30 independent runs for the 16 test functions at $D = 30$, where "$w/t/l$" means that EDE wins in w functions, ties in t functions, and loses in l functions, compared with its competitors.

Number	max FEs	JADE-w	SaDE	SaJADE	EDE
f_{01}	15e4	2.69e − 56 (1.41e − 55)†	3.42e − 37 (3.63e − 37)†	1.10e − 79 (7.52e − 79)†	**4.19e − 304 (0.00e − 000)**
f_{02}	20e4	3.18e − 25 (2.05e − 24)†	3.51e − 25 (2.74e − 25)†	1.35e − 47 (7.53e − 47)†	**4.65e − 227 (0.00e − 000)**
f_{03}	50e4	**6.11e − 81 (1.62e − 80)**‡	1.54e − 14 (4.56e − 14)†	1.77e − 77 (3.39e − 77)†	8.01e − 080 (1.71e − 079)
f_{04}	50e4	5.29e − 14 (2.05e − 14)†	6.39e − 27 (8.27e − 27)†	1.26e − 19 (1.35e − 19)†	**2.79e − 139 (6.39e − 139)**
f_{05}	50e4	1.59e − 01 (7.89e − 01)†	7.98e − 02 (5.64e − 01)†	**1.60e − 30 (6.32e − 30)**‡	4.42e − 028 (1.38e − 027)
f_{06}	1e4	5.62e − 00 (1.87e − 00)†	5.07e + 01 (1.34e + 01)†	**0.00e − 00 (0.00e − 00)**≈	**0.00e − 000 (0.00e − 000)**
f_{07}	30e4	6.14e − 04 (2.55e − 04)‡	2.06e − 03 (5.21e − 04)‡	**4.10e − 04 (1.48e − 04)**‡	2.30e − 003 (8.87e − 004)
f_{08}	10e4	2.62e − 04 (3.59e − 04)†	1.13e − 08 (1.08e − 08)†	6.83e − 07 (2.70e − 06)†	**0.00e − 000 (0.00e − 000)**
f_{09}	10e4	1.33e − 01 (9.74e − 02)†	2.43e − 00 (1.60e − 00)†	1.54e − 01 (2.25e − 01)†	**0.00e − 000 (0.00e − 000)**
f_{10}	5e4	3.35e − 09 (2.84e − 09)†	3.81e − 06 (8.26e − 07)†	1.12e − 12 (1.07e − 12)†	**5.03e − 015 (1.34e − 015)**
f_{11}	5e4	1.57e − 08 (1.09e − 07)‡	2.52e − 09 (1.24e − 08)‡	**0.00e − 00 (0.00e − 00)**‡	2.31e − 002 (2.44e − 002)
f_{12}	5e4	1.67e − 15 (1.02e − 14)†	8.25e − 12 (5.12e − 12)†	2.10e − 23 (6.89e − 23)†	**1.57e − 032 (5.56e − 048)**
f_{13}	5e4	1.87e − 10 (1.09e − 09)†	1.93e − 09 (1.53e − 09)†	3.83e − 21 (1.56e − 20)†	**1.35e − 032 (5.56e − 048)**
f_{18}	30e4	2.00e − 01 (1.63e − 02)‡	**1.56e − 01 (5.01e − 02)**‡	1.76e − 01 (4.28e − 02)‡	4.33e − 001 (8.02e − 002)
f_{19}	30e4	1.87e − 10 (2.09e − 09)†	1.93e − 09 (1.53e − 09)†	3.83e − 21 (1.56e − 20)†	**0.00e − 000** (0.00e − 000)
f_{21}	10e4	1.35e − 00 (6.08e − 01)†	1.46e − 00 (1.02e − 00)†	1.13e − 01 (1.60e − 01)†	**0.00e − 000 (0.00e − 000)**
$w/t/l$		12/0/4	13/0/3	11/1/4	—

† indicates that EDE is better than its competitor.
‡ means that EDE is worse than its competitor.
≈ means that the performance of the corresponding algorithm is even with that of EDE.
Bold entities mean the best results.

To testify the convergence performance of EDE, twenty-five benchmark functions with different characteristics from literatures are employed. The first experimental results demonstrate that EDE significantly enhances the performance of standard DE in terms of the best, worst, median, mean, and standard deviation (Std.) values of final solutions in most cases. Moreover, other two comparisons also show that EDE performs significantly better than or at least highly competitive with other five well-known algorithms, that is, JADE, SaDE, SaJADE, ABC, and MABC, on the majority of the corresponding benchmark functions. Therefore, it can be concluded that EDE is an efficient method and it may be a good alternative for solving complex numerical optimization problems.

Last but not least, it is desirable to further apply the EDE algorithm to deal with other optimization problems such as the training of neural networks, system parameter identification, and data clustering.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Table 4: Comparison between EDE and other two ABCs over 30 independent runs on the 21 test functions with $D = 30$ in terms of mean and standard deviation.

Number	max FEs	ABC	MABC	EDE
f_{01}	15e4	5.21e − 010 (2.46e − 010)	9.43e − 032 (6.67e − 032)	**4.19e − 304 (0.00e − 000)**
f_{02}	15e4	1.83e − 006 (4.80e − 007)	2.40e − 017 (9.02e − 018)	**1.24e − 169 (0.00e − 000)**
f_{04}	15e4	1.80e + 001 (2.25e − 000)	1.02e + 001 (1.49e − 000)	**1.65e − 041 (2.79e − 041)**
f_{05}	15e4	4.23e − 001 (4.34e − 001)	6.11e − 001 (4.55e − 001)	**8.50e − 003 (4.43e − 002)**
f_{06}	15e4	**0.00e − 000 (0.00e − 000)**	**0.00e − 000 (0.00e − 000)**	**0.00e − 000 (0.00e − 000)**
f_{07}	15e4	8.74e − 002 (1.77e − 002)	3.71e − 002 (8.53e − 003)	**4.90e − 003 (2.40e − 003)**
f_{08}	15e4	8.86e + 001 (8.62e + 001)	**−1.21e − 013 (4.53e − 013)**	0.00e − 000 (0.00e − 000)[a]
f_{09}	15e4	4.81e − 003 (2.57e − 002)	**0.00e − 000 (0.00e − 000)**	**0.00e − 000 (0.00e − 000)**
f_{10}	15e4	4.83e − 006 (2.12e − 006)	4.13e − 014 (2.17e − 015)	**4.44e − 015 (0.00e − 000)**
f_{11}	15e4	1.61e − 008 (3.99e − 008)	**0.00e − 000 (0.00e − 000)**	2.09e − 002 (2.19e − 002)
f_{12}	15e4	1.39e − 011 (3.82e − 012)	1.90e − 032 (3.70e − 033)	**1.57e − 032(5.56e − 048)**
f_{13}	15e4	1.06e − 009 (4.24e − 010)	2.23e − 031 (1.46e − 031)	**1.34e − 032(5.56e − 048)**
f_{14}	15e4	2.22e − 011 (1.14e − 011)	2.10e − 032 (1.56e − 032)	**2.19e − 305 (0.00e − 000)**
f_{15}	15e4	5.51e − 029 (6.70e − 029)	1.45e − 067 (2.28e − 067)	**0.00e − 000 (0.00e − 000)**
f_{16}	15e4	1.12e − 001 (2.97e − 001)	**0.00e − 000 (0.00e − 000)**	**0.00e − 000 (0.00e − 000)**
f_{17}	15e4	4.41e − 001 (1.81e − 002)	2.95e − 001 (3.17e − 002)	**1.63e − 001 (5.62e − 002)**
f_{19}	15e4	7.66e − 005 (2.76e − 005)	1.58e − 016 (2.48e − 016)	**0.00e − 000 (0.00e − 000)**
f_{20}	15e4	1.55e − 009 (5.54e − 010)	**0.00e − 000 (0.00e − 000)**	**0.00e − 000 (0.00e − 000)**
f_{21}	15e4	1.49e − 001 (3.55e − 001)	**0.00e − 000 (0.00e − 000)**	**0.00e − 000 (0.00e − 000)**
f_{22}	15e4	9.73e − 005 (5.69e − 005)	4.92e − 014 (5.31e − 015)	**7.40e − 015 (1.34e − 015)**
f_{23}	15e4	4.93e − 004 (2.25e − 003)	**0.00e − 000 (0.00e − 000)**	2.10e − 002 (2.70e − 002)
$w/t/l$		18/1/2	13/5/3	—

Bold entities mean the best results.
Here "a" means that the results obtained by EDE are set to zero on the function f_8 when the results are less than $1e-308$. This is the reason that the coefficient −418.98288727243369 with low precision in function f_8 may result in the negative results. As a matter of fact, the results should be zero.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant nos. 61064012, 61164003, 61263027, 61364026, and 61563028), the Natural Science Foundation of Gansu Province (Grant no. 148RJZA030), New Teacher Project of Research Fund for the Doctoral Program of Higher Education of China (Grant no. 20126204120002), and the Science and Technology Foundation of Lanzhou Jiaotong University (Grant no. ZC2014010).

References

[1] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[2] R. Storn and K. Price, "Differential evolution—a simple and efficient adaptive scheme for global opitmization over continuous spaces," Report TR-95-012, 1995.

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

[4] M. M. Ali and A. Törn, "Population set-based global optimization algorithms: some modifications and numerical studies," *Computers & Operations Research*, vol. 31, no. 10, pp. 1703–1725, 2004.

[5] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Computing*, vol. 9, no. 6, pp. 448–462, 2005.

[6] J. Sun, Q. Zhang, and E. P. Tsang, "DE/EDA: a new evolutionary algorithm for global optimization," *Information Sciences*, vol. 169, no. 3-4, pp. 249–262, 2005.

[7] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[8] P. Kaelo and M. M. Ali, "A numerical study of some modified differential evolution algorithms," *European Journal of Operational Research*, vol. 169, no. 3, pp. 1176–1184, 2006.

[9] M. M. Ali, "Differential evolution with preferential crossover," *European Journal of Operational Research*, vol. 181, no. 3, pp. 1137–1147, 2007.

[10] Y.-J. Wang and J.-S. Zhang, "Global optimization by an improved differential evolutionary algorithm," *Applied Mathematics and Computation*, vol. 188, no. 1, pp. 669–680, 2007.

[11] N. Noman and H. Iba, "Accelerating differential evolution using an adaptive local search," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 107–125, 2008.

[12] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition versus randomness in soft computing techniques," *Applied Soft Computing Journal*, vol. 8, no. 2, pp. 906–918, 2008.

[13] R. S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 64–79, 2008.

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

[15] J. Zhang and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–928, 2009.

[16] Z. Yang, K. Tang, and X. Yao, "Scalability of generalized adaptive differential evolution for large-scale continuous optimization," *Soft Computing*, vol. 15, no. 11, pp. 2141–2155, 2011.

[17] W. Gong, Z. Cai, and C. X. Ling, "DE/BBO: a hybrid differential evolution with biogeography-based optimization for global numerical optimization," *Soft Computing*, vol. 15, no. 4, pp. 645–665, 2011.

[18] M. Weber, F. Neri, and V. Tirronen, "A study on scale factor in distributed differential evolution," *Information Sciences*, vol. 181, no. 12, pp. 2488–2511, 2011.

[19] Y. Wang, Z. X. Cai, and Q. F. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 55–66, 2011.

[20] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing*, vol. 11, no. 2, pp. 1679–1696, 2011.

[21] A. Ghosh, S. Das, A. Chowdhury, and R. Giri, "An improved differential evolution algorithm with fitness-based adaptation of the control parameters," *Information Sciences*, vol. 181, no. 18, pp. 3749–3765, 2011.

[22] W. Gong, Á. Fialho, Z. Cai, and H. Li, "Adaptive strategy selection in differential evolution for numerical optimization: an empirical study," *Information Sciences. An International Journal*, vol. 181, no. 24, pp. 5364–5386, 2011.

[23] W. Y. Gong, Z. H. Cai, C. X. Ling, and H. Li, "Enhanced differential evolution with adaptive strategies for numerical optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 41, no. 2, pp. 397–413, 2011.

[24] M. Weber, F. Neri, and V. Tirronen, "Shuffle or update parallel differential evolution for large-scale optimization," *Soft Computing*, vol. 15, no. 11, pp. 2089–2107, 2011.

[25] A. P. Piotrowski, J. J. Napiorkowski, and A. Kiczko, "Differential evolution algorithm with separated groups for multi-dimensional optimization problems," *European Journal of Operational Research*, vol. 216, no. 1, pp. 33–46, 2012.

[26] S. Ghosh, S. Das, A. V. Vasilakos, and K. Suresh, "On convergence of differential evolution over a class of continuous functions with unique global optimum," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 1, pp. 107–124, 2012.

[27] E. Mezura-Montes, M. E. Miranda-Varela, and R. del Carmen Gómez-Ramón, "Differential evolution in constrained numerical optimization: an empirical study," *Information Sciences*, vol. 180, no. 22, pp. 4223–4262, 2010.

[28] H. Liu, Z. Cai, and Y. Wang, "Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization," *Applied Soft Computing Journal*, vol. 10, no. 2, pp. 629–640, 2010.

[29] D. Zou, H. Liu, L. Gao, and S. Li, "A novel modified differential evolution algorithm for constrained optimization problems," *Computers and Mathematics with Applications*, vol. 61, no. 6, pp. 1608–1623, 2011.

[30] W. Gong and Z. Cai, "An improved multiobjective differential evolution based on Pareto-adaptive ϵ-dominance and orthogonal design," *European Journal of Operational Research*, vol. 198, no. 2, pp. 576–601, 2009.

[31] X. Li and M. Yin, "An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure," *Advances in Engineering Software*, vol. 55, pp. 10–31, 2013.

[32] M. Fatih Tasgetiren, Q.-K. Pan, P. N. Suganthan, and O. Buyukdagli, "A variable iterated greedy algorithm with differential evolution for the no-idle permutation flowshop scheduling problem," *Computers and Operations Research*, vol. 40, no. 7, pp. 1729–1743, 2013.

[33] R. Zhang, S. Song, and C. Wu, "A hybrid differential evolution algorithm for job shop scheduling problems with expected total tardiness criterion," *Applied Soft Computing Journal*, vol. 13, no. 3, pp. 1448–1458, 2013.

[34] A. Nobakhti and H. Wang, "A simple self-adaptive differential Evolution algorithm with application on the ALSTOM gasifier," *Applied Soft Computing Journal*, vol. 8, no. 1, pp. 350–370, 2008.

[35] Y. Liu and F. Sun, "A fast differential evolution algorithm using k-Nearest Neighbour predictor," *Expert Systems with Applications*, vol. 38, no. 4, pp. 4254–4258, 2011.

[36] L. Wang, X. Fu, Y. Mao, M. Ilyas Menhas, and M. Fei, "A novel modified binary differential evolution algorithm and its applications," *Neurocomputing*, vol. 98, pp. 55–75, 2012.

[37] Y. Tang, X. Zhang, C. Hua, L. Li, and Y. Yang, "Parameter identification of commensurate fractional-order chaotic system via differential evolution," *Physics Letters A*, vol. 376, no. 4, pp. 457–464, 2012.

[38] H.-C. Lu, M.-H. Chang, and C.-H. Tsai, "Parameter estimation of fuzzy neural network controller based on a modified differential evolution," *Neurocomputing*, vol. 89, pp. 178–192, 2012.

[39] F. Fabris and R. A. Krohling, "A co-evolutionary differential evolution algorithm for solving min-max optimization problems implemented on GPU using C-CUDA," *Expert Systems with Applications*, vol. 39, no. 12, pp. 10324–10333, 2012.

[40] W.-F. Gao and S.-Y. Liu, "A modified artificial bee colony algorithm," *Computers & Operations Research*, vol. 39, no. 3, pp. 687–697, 2012.

[41] A. K. Qin and F. Forbes, "Dynamic regional harmony search with opposition and local learning," in *Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference (GECCO '11)*, pp. 53–54, Dublin, Ireland, July 2011.

[42] W.-f. Gao, S.-Y. Liu, and L.-l. Huang, "Particle swarm optimization with chaotic opposition-based population initialization and stochastic search technique," *Communications in Nonlinear Science and Numerical Simulation*, vol. 17, no. 11, pp. 4316–4327, 2012.

[43] N. Dong, C.-H. Wu, W.-H. Ip, Z.-Q. Chen, C.-Y. Chan, and K.-L. Yung, "An opposition-based chaotic GA/PSO hybrid algorithm and its application in circle detection," *Computers and*

Mathematics with Applications, vol. 64, no. 6, pp. 1886–1902, 2012.

[44] P. K. Roy, C. Paul, and S. Sultana, "Oppositional teaching learning based optimization approach for combined heat and power dispatch," *International Journal of Electrical Power & Energy Systems*, vol. 57, pp. 392–403, 2014.

[45] Q. Xu, L. Wang, N. Wang, X. Hei, and L. Zhao, "A review of opposition-based learning from 2005 to 2012," *Engineering Applications of Artificial Intelligence*, vol. 29, pp. 1–12, 2014.

[46] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[47] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, 2007.

[48] D. Karaboga and B. Akay, "A comparative study of Artificial Bee Colony algorithm," *Applied Mathematics and Computation*, vol. 214, no. 1, pp. 108–132, 2009.