Applied Soft Computing 18 (2014) 232–247

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Differential evolution based on covariance matrix learning and bimodal distribution parameter setting

Yong Wang[a,b], Han-Xiong Li[b,c,*], Tingwen Huang[d], Long Li[a]

[a] School of Information Science and Engineering, Central South University, Changsha 410083, PR China
[b] Department of Systems Engineering and Engineering Management, City University of Hong Kong, Hong Kong, PR China
[c] State Key Laboratory of High Performance Complex Manufacturing, Central South University, Changsha 410083, PR China
[d] Texas A&M University at Qatar, Doha 5825, Qatar

ARTICLE INFO

Article history:
Received 19 April 2013
Received in revised form
21 December 2013
Accepted 28 January 2014
Available online 3 February 2014

Keywords:
Differential evolution
Global numerical and engineering optimization
Covariance matrix learning
Bimodal distribution parameter setting

ABSTRACT

Differential evolution (DE) is an efficient and robust evolutionary algorithm, which has been widely applied to solve global optimization problems. As we know, crossover operator plays a very important role on the performance of DE. However, the commonly used crossover operators of DE are dependent mainly on the coordinate system and are not rotation-invariant processes. In this paper, covariance matrix learning is presented to establish an appropriate coordinate system for the crossover operator. By doing this, the dependence of DE on the coordinate system has been relieved to a certain extent, and the capability of DE to solve problems with high variable correlation has been enhanced. Moreover, bimodal distribution parameter setting is proposed for the control parameters of the mutation and crossover operators in this paper, with the aim of balancing the exploration and exploitation abilities of DE. By incorporating the covariance matrix learning and the bimodal distribution parameter setting into DE, this paper presents a novel DE variant, called CoBiDE. CoBiDE has been tested on 25 benchmark test functions, as well as a variety of real-world optimization problems taken from diverse fields including radar system, power systems, hydrothermal scheduling, spacecraft trajectory optimization, etc. The experimental results demonstrate the effectiveness of CoBiDE for global numerical and engineering optimization. Compared with other DE variants and other state-of-the-art evolutionary algorithms, CoBiDE shows overall better performance.

1. Introduction

Differential evolution (DE), proposed by Storn and Price [1,2] in 1995, has become a hotspot in the community of evolutionary computation. Similar to other evolutionary algorithms (EAs), DE is a population-based optimization algorithm. In DE, each individual in the population is called a target vector. DE produces a mutant vector by making use of the mutation operator, which perturbs a target vector using the difference vector of other individuals in the population. Afterward, the crossover operator is applied to the target vector and the mutant vector to generate a trial vector. Finally, the trial vector competes with its target vector for survival according to their objective function values. Due to some advantages, e.g., simple structure, ease of implementation, and fast convergence speed, DE has been widely applied to some fields of science and engineering, such as cluster analysis [3], robot control [4], controller design [5], and graph theory [6].

It is noteworthy that in DE, the crossover operator depends mainly on the coordinate system and the distribution information of the population is usually unreasonably ignored. Moreover, the crossover operator of DE can be considered as a discrete recombination [7], and thus, the interactions among variables have not been systematically studied. As a result, DE often loses its effectiveness and advantages when solving problems with high variable correlation.

In addition, DE is sensitive to its two main control parameters: the scaling factor F and the crossover control parameter CR. These two control parameters have a significant impact on the performance of DE. Moreover, different control parameter settings show different characteristics [8]. For example, a larger F is effective for global search; however, a smaller F can accelerate the convergence. On the other hand, a larger CR results in higher diversity of the population, since the trial vector will inherit more information from the mutant vector. However, a smaller CR focuses on local exploitation since the target vector will contribute more information to the trial vector. Indeed, it is still an open issue to choose suitable settings of

* Corresponding author.
E-mail addresses: ywang@csu.edu.cn (Y. Wang), mehxli@cityu.edu.hk (H.-X. Li), tingwen.huang@qatar.tamu.edu (T. Huang).

1568-4946/$ – see front matter © 2014 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.asoc.2014.01.038

F and CR to balance the exploration and exploitation of DE during the evolution.

Based on the above considerations, in this paper, we present a novel DE, referred as CoBiDE, including two main components: covariance matrix learning and bimodal distribution parameter setting. In CoBiDE, the covariance matrix learning establishes a coordinate system according to the current population distribution, and then the crossover operator is applied according to the coordinate system thus built to generate the trial vector. Furthermore, in CoBiDE, both F and CR are produced according to a bimodal distribution composed of two Cauchy distributions, the aim of which is to balance the global exploration and the local exploitation during the evolution. CoBiDE has been tested on 25 benchmark test functions developed for the 2005 IEEE Congress on Evolutionary Computation (IEEE CEC2005) [9], as well as a variety of real-world application problems [10]. The experimental results suggest that the performance of CoBiDE is better than that of four other DE variants and three other state-of-the-art EAs.

The remainder of this paper is organized as follows. Section 2 briefly introduces DE and its operators. Section 3 reviews the related work and four main research directions of DE. Then, CoBiDE is presented in Section 4. The experimental results are given in Section 5. Section 6 concludes this paper.

2. Differential evolution (DE)

DE is a population-based heuristic search algorithm. Similar to other EAs, DE contains three basic operators: mutation, crossover, and selection. Firstly, DE produces an initial population by randomly sampling several points (each point is called a target vector) from the search space:

$$P_0 = \{\vec{x}_{i,0} = (x_{i,1,0}, x_{i,2,0}, \ldots, x_{i,D,0}), \quad i = 1, 2, \ldots, NP\} \tag{1}$$

where NP denotes the population size and D denotes the number of variables.

At each generation G, a mutant vector $\vec{v}_{i,G} = (v_{i,1,G}, v_{i,2,G}, \ldots, v_{i,D,G})$ $(i \in 1, 2, \ldots, NP)$ is produced by the mutation operator for each target vector $\vec{x}_{i,G}$. Afterward, the crossover operator is implemented on the mutant vector and the target vector to generate a trial vector $\vec{u}_{i,G} = (u_{i,1,G}, u_{i,2,G}, \ldots, u_{i,D,G})$ $(i \in 1, 2, \ldots, NP)$. The crossover operator and the mutation operator together are called trial vector generation strategy. The selection operator of DE is based on a one-to-one competition between the target vector and the trial vector.

Next, the mutation, crossover, and selection operators are introduced.

2.1. Mutation operator

The commonly used mutation operator can be formulated as follows:

$$\vec{v}_{i,G} = \vec{x}_{r1,G} + F \cdot (\vec{x}_{r2,G} - \vec{x}_{r3,G}) \tag{2}$$

where $r1$, $r2$, and $r3$ are mutually different integers randomly chosen from $[1, NP]$ and also different from i, and F is the scaling factor.

2.2. Crossover operator

The crossover operator combines the mutant vector $\vec{v}_{i,G}$ with the target vector $\vec{x}_{i,G}$ to generate a trial vector $\vec{u}_{i,G}$:

$$u_{i,j,G} = \begin{cases} v_{i,j,G}, & \text{if} \quad rand_j(0,1) \le CR \quad \text{or} \quad j = j_{rand} \\ x_{i,j,G}, & \text{otherwise} \end{cases} \tag{3}$$

where j_{rand} is a random integer between 1 and D, resulting in the trial vector being different from the target vector by at least one dimension, $rand_j(0, 1)$ is a uniformly distributed random number between 0 and 1, and CR is the crossover control parameter.

Based on Eq. (3), it is clear that the trial vector is a vertex of the hyper-rectangle defined by the mutant and target vectors [11]. Moreover, since the information of the trial vector is provided by the mutant vector or the target vector, the crossover operator is dependent on the coordinate system.

2.3. Selection operator

The selection operator of DE adopts a one-to-one competition between the target vector $\vec{x}_{i,G}$ and the trial vector $\vec{u}_{i,G}$. If the objective function value of the trial vector is less than or equal to that of the target vector, then the trial vector will survive into the next generation, otherwise, the target vector will enter the next generation:

$$\vec{x}_{i,G+1} = \begin{cases} \vec{u}_{i,G}, & \text{if} \quad f(\vec{u}_{i,G}) \le f(\vec{x}_{i,G}) \\ \vec{x}_{i,G}, & \text{otherwise} \end{cases} \tag{4}$$

3. The related work

During the past fifteen years, DE has attracted much attention by the researchers [12]. The current studies of DE mainly focus on the following four aspects: (1) improving the trial vector generation strategy, (2) adapting the control parameter setting, (3) hybridizing with other techniques, and (4) integrating multiple trial vector generation strategies with multiple control parameter settings.

3.1. Improving the trial vector generation strategy

Fan and Lampinen [13] proposed a trigonometric mutation operator and embedded it into DE to design a new method called TDE. In TDE, a probability parameter M_t is utilized to balance the trigonometric mutation operator and the original mutation operator of DE. The trigonometric mutation can be considered as a local search operator, which is able to enhance the convergence velocity of DE. The performance of TDE has been evaluated on two test functions and two practical problems.

Zhang and Sanderson [14] presented an improved current-to-best/1 operator, called current-to-$p$best/1, which can be formulated as follows:

$$\vec{v}_{i,G} = \vec{x}_{i,G} + F_i \cdot (\vec{x}^{\,p}_{best,G} - \vec{x}_{i,G}) + F_i \cdot (\vec{x}_{r1,G} - \vec{x}'_{r2,G}), \quad i \in \{1, 2, \ldots, NP\} \tag{5}$$

where $\vec{x}^{\,p}_{best,G}$ is randomly chosen from the best $100p\%$ individuals in the current population, and p is chosen from $(0, 1]$. Moreover, the previously generated offspring, which cannot survive into the next population, have been stored into a predefined archive. The individual $\vec{x}'_{r2,G}$ in Eq. (5) is randomly chosen from the union of the archive and the current population. As analyzed in [14], the advantages of the current-to-$p$best/1 operator are twofold: (1) the information of multiple best individuals can balance the greediness of the mutation and the diversity of the population, and (2) the difference between the recently explored inferior individuals and the current population may represent promising directions toward the global optimum.

Das et al. [15] proposed a neighborhood-based mutation operator, which contains two parts: global neighborhood-based mutation and local neighborhood-based mutation. In the method proposed by Das et al. [15], two trial vectors are produced by the global and local neighborhood-based mutation. Moreover, these two trial vectors are combined to form the actual trial vector by using a weight factor. Clearly, the main aim of the

neighborhood-based mutation operator is to balance the exploration and exploitation abilities of DE. This mutation operator has been tested on 24 benchmark test functions and two real-world problems and shown very competitive results.

After recognizing that the trial vector generated by the crossover operator is just a vertex of the hyper-rectangle defined by the mutant and target vectors, Wang et al. [11] employed orthogonal crossover [16] to make a systematic and rational search in the hyper-rectangle defined by the mutant and target vectors, and proposed a generic framework to enhance the search ability of DE. The experimental results have demonstrated that this framework can be used to improve the performance of different variants of DE.

3.2. Adapting the control parameter setting

Liu and Lampinen [17] designed a fuzzy adaptive DE (FADE) based on fuzzy logic controller. In FADE, the mean square roots of differences of the objective function values and the population members during the successive generations are treated as the inputs of the fuzzy logic controller, and the outputs are the values of F and CR. The experimental results have shown that FADE outperforms the classic DE on problems with high dimensionality. The main weakness of FADE lies in its complicated implementation due to fuzzy adapting.

Brest [18] proposed a DE with self-adaptive parameter control (jDE). In jDE, the control parameters F and CR are encoded into the chromosome and participate in the evolution. Each individual in the population is assigned an initial control parameter setting: $F_i = 0.5$ and $CR_i = 0.9$ ($i = 1, 2, \ldots, NP$). During the evolution, jDE regenerates F_i and CR_i according to the uniform random distributions $U(0.1, 0.9)$ and $U(0, 1)$ with probabilities τ_1 and τ_2, respectively. One of the main advantages of jDE is that its implementation is very simple. In [18], 21 test functions have been used to assess the performance of jDE.

In JADE proposed by Zhang and Sanderson [14], for each target vector, the scaling factor F is generated by the Cauchy distribution $C(\mu_F, 0.1)$, and the crossover control parameter CR obeys the normal distribution $N(\mu_{CR}, 0.1)$. In addition, JADE uses the following equations to update μ_F and μ_{CR}:

$$\mu_F = (1 - c) \cdot \mu_F + c \cdot mean_L(S_F) \tag{6}$$

$$\mu_{CR} = (1 - c) \cdot \mu_{CR} + c \cdot mean_A(S_{CR}) \tag{7}$$

where c controls the rate of parameter adaptation, S_F and S_{CR} are the sets of all successful scaling factor F and crossover control parameter CR at each generation, respectively, and $mean_A(\cdot)$ and $mean_L(\cdot)$ are the usual arithmetic mean and the Lehmer mean, respectively. The above parameter adaptation has the capability to adapt parameters to appropriate values, and thus, improves the robustness of DE.

3.3. Hybridizing with other techniques

Noman and Iba [7] proposed a crossover-based adaptive local search operator to enhance the convergence rate of DE. In this method, simplex crossover [19] is applied to the best individual and two other individuals of the population at each generation before implementing DE. This method does not add any additional complexity or any additional parameter. Moreover, it exhibits a higher convergence velocity compared with the original DE.

Opposition-based DE (ODE) is proposed by Rahnamayan et al. [20], which employs opposition-based learning to generate the initial population and new solutions. The experimental results suggest that opposition-based learning is a very effective way to speed up the convergence of DE. Concretely, ODE is on average 44% faster than the original DE on 58 test functions.

Sun et al. [21] combined DE with estimation of distribution algorithm (EDA), and proposed DE/EDA. In DE/EDA, one part of the trial vector is generated in the DE way, and the other part of the trial vector is sampled from the constructed probability distribution model. As a result, DE/EDA can not only utilize the global statistical information derived from EDA, but also use the differential information provided by DE.

3.4. Integrating multiple trial vector generation strategies with multiple control parameter settings

Recently, some researchers investigated the idea of integrating multiple trial vector generation strategies with multiple control parameter settings in DE. The main motivation is that different strategies along with different parameter settings may be suitable to different problems [8].

Qing et al. [8] proposed a self-adaptive DE (SaDE), in which both trial vector generation strategies and control parameter settings are self-adapted according to the previous information. SaDE establishes a strategy candidate pool which contains four trial vector generation strategies. At each generation, one trial vector generation strategy is chosen for one individual. In addition, SaDE assigns different control parameter settings for different individuals. SaDE has been used to solve a suite of 26 test functions and the experimental results are very promising.

Mallipeddi et al. [22] proposed a DE with ensemble of control parameter settings and trial vector generation strategies (EPSDE). EPSDE involves a pool of distinct trial vector generation strategies and a pool of values for each control parameter. During the evolution, a trial vector generation strategy and a control parameter setting are chosen based on their success experience in the past generations to create a trial vector. As a result, the successful combination of strategy and parameter setting has a higher probability to produce the trial vector. Since the strategies and the parameter settings in a pool have distinct properties, EPSDE exhibit distinct performance characteristics during different stages of the evolution.

During the past fifteen years, DE researchers have obtained some important experiences about choosing trial vector generation strategies and control parameter settings, which will be very useful for designing more effective DE. Motivated by the above consideration, Wang et al. [23] investigated whether the performance of DE can be improved by combining several trial vector generation strategies with several different control parameter settings, which exhibit different characterizes, and proposed a composite DE, named CoDE. CoDE combines three trial vector generation strategies with three control parameter settings in a random way to produce the trial vectors. The performance of CoDE has been evaluated on 25 benchmark test functions developed for IEEE CEC2005 [9].

Gong et al. [24] used four trial vector generation strategies proposed in [14] to form the strategy candidate pool and designed two adaptive methods to choose a suitable trial vector generation strategy for a problem at hand. In addition, the parameter adaptation mechanism proposed by Gong et al. [24] is similar to that proposed in [14]. The experimental results on 20 test functions and two real-world problems have verified that the method proposed in [24] is able to adaptively determine a more suitable strategy for a specific problem.

4. Proposed approach

In this section, we propose a novel DE, named CoBiDE. CoBiDE contains two main components: covariance matrix learning and

bimodal distribution parameter setting. Next, the implementation of the above two main components will be introduced in detail.

4.1. Covariance matrix learning

As mentioned previously, the crossover operator of DE is dependent mainly on the coordinate system, and the distribution information of the population, which could reflect the landscape of the problem to a certain extent [12], is usually ignored during the evolution. Indeed, the statistical properties of the population (such as mean value, variance, and covariance) can be utilized to represent the distribution of the population. In particular, the covariance matrix composed of variance and covariance reflects the diversity of the population and the interactions among the variables. Hence, systemically utilizing the covariance matrix should be very useful for relaxing the dependence of DE on the coordinate system and loosing the interactions among the variables.

Based on the above analysis, covariance matrix learning is proposed in this paper, the aim of which is to establish an Eigen coordinate system with loose variable correlation for the crossover operator. Fig. 1 shows the differences between the crossover operator in the original coordinate system (Fig. 1(a)) and the crossover operator in the Eigen coordinate system (Fig. 1(b)) for a problem with variable correlation. Suppose that the Eigen coordinate system (i.e., $ox'_1x'_2$) is obtained after analyzing the distribution of the population. From Fig. 1, it is clear that crossover in the Eigen coordinate system is more promising to find the global optimum, since the trial vectors generated by the crossover in the Eigen coordinate system may be more close to the global optimum than the trial vectors created by the crossover in the original coordinate system.

In this paper, the covariance matrix learning includes two core techniques: Eigen decomposition of the covariance matrix and the coordinate transformation. The purpose of the former is to obtain Eigen vectors which can serve as the axial orientations of the Eigen coordinate system. In addition, the latter transforms the trial vectors into the original coordinate system, after implementing the crossover operator according to the Eigen coordinate system. The procedure of the covariance matrix learning is introduced as follows.

Step 1. Compute the covariance matrix C of the top $ps{\cdot}NP$ individuals in the current population, and apply Eigen decomposition to C as follows:

$$C = BD^2B^T \tag{8}$$

where B and B^T are orthogonal matrices and D is a diagonal matrix composed of Eigen values. Note that each column of B is an Eigen vector of the covariance matrix C.

Step 2. Update the target vector and the mutant vector in the Eigen coordinate system by making use of B^T:

$$\vec{x}'_{i,G} = B^{-1}\vec{x}_{i,G} = B^T\vec{x}_{i,G} \tag{9}$$

$$\vec{v}'_{i,G} = B^{-1}\vec{v}_{i,G} = B^T\vec{v}_{i,G} \tag{10}$$

Step 3. Apply the crossover operator to $\vec{x}'_{i,G}$ and $\vec{v}'_{i,G}$, and create a trial vector $\vec{u}'_{i,G}$ in the Eigen coordinate system:

$$u'_{i,j,G} = \begin{cases} v'_{i,j,G}, & \text{if } \ rand_j(0,1) \le CR \ \text{ or } \ j = j_{rand} \\ x'_{i,j,G}, & \text{otherwise} \end{cases} \tag{11}$$

Step 4. Transform $\vec{u}'_{i,G}$ into the original coordinate system by taking advantage of B:

$$\vec{u}_{i,G} = B\vec{u}'_{i,G} \tag{12}$$

where $\vec{u}_{i,G}$ is the trial vector in the original coordinate system.

During the evolution, due to the randomness of the distribution of the population, if all the individuals in the population are used to compute the covariance matrix, the covariance matrix will be disturbed by such randomness and, as a result, the Eigen coordinate system constructed may not be quite reasonable. Therefore, in Step 1, we use the $ps{\cdot}NP$ individuals with the minimum objective function values in the population to compute the covariance matrix, where ps is in the interval [0, 1].

Remark 1. CMA-ES [25], BLXPCA [26], and BLXICA [26] have a similar motivation to use statistical information based on the covariance matrix. However, there are some differences between CoBiDE and them. In CoBiDE, the parameter ps is introduced to compute the covariance matrix of the top $ps{\cdot}NP$ individuals in the current population, while CMA-ES adopts a weighted method to compute the covariance matrix. In addition, all the individuals in the population are used to compute the covariance matrix in BLXPCA and BLXICA. On the other hand, CMA-ES, BLXPCA, and BLXICA use Eigen values and Eigen vectors obtained by the Eigen decomposition of the covariance matrix simultaneously. However, because of the properties of crossover in DE, the proposed approach only uses Eigen vectors to construct an appropriate coordinate system, and Eigen values are not used.

Remark 2. Recently, several methods which hybridize DE with CMA-ES [25] have been proposed. For example, DE performs the global exploration and CMA-ES is used as a local search engine in [27]. In [28], CMA-ES and a hybrid DE are executed serially. Moreover, two populations are utilized, one for CMA-ES and the other for the hybrid DE. LaTorre et al. [29] presented a multiple offspring sampling framework to combine a restart CMA-ES [30] with DE. In this framework, the average fitness increment is adopted as a quality function to update the participation ratios of the restart CMA-ES and DE. There are two major differences between CoBiDE and the above three methods. Firstly, DE is not coupled with CMA-ES in CoBiDE. Indeed, CoBiDE only exploits the statistical information provided by the covariance matrix of the population. Secondly, in CoBiDE the statistical information provided by the covariance matrix is embedded into DE to strengthen the crossover operator. However, in the above three methods, CMA-ES is independent of DE.

4.2. Bimodal distribution parameter setting

In this subsection, bimodal distribution parameter setting is proposed for the scaling factor F and the crossover control parameter CR. It is necessary to emphasize that the proposed bimodal distribution parameter setting is inspired by [14] and [23]. In addition, like [18], the parameters F and CR are encoded into each target vector $\vec{x}_{i,G}$, i.e., $F_{i,G}$ and $CR_{i,G}$ correspond to each $\vec{x}_{i,G}$. Moreover, if the trial vector $\vec{u}_{i,G}$ can successfully enter the next population, then $F_{i,G+1} = F_{i,G}$ and $CR_{i,G+1} = CR_{i,G}$; otherwise, $F_{i,G+1}$ and $CR_{i,G+1}$ are generated for the next generation according to the bimodal distribution parameter setting.

The bimodal distribution for $F_{i,G}$ ($i \in \{1, \ldots, NP\}$) is composed of two Cauchy distributions as follows:

$$F_{i,G} = \begin{cases} randc_i(0.65, 0.1), & \text{if } \ rand(0,1) < 0.5 \\ randc_i(1.0, 0.1), & \text{otherwise} \end{cases} \tag{13}$$

where $rand(0, 1)$ is a uniformly distributed random number between 0 and 1, and $randc_i(a, b)$ is a random number obeying a Cauchy distribution with location parameter a and scale parameter b. If the value of $F_{i,G}$ is larger than 1.0, then is truncated to 1.0; and if the value of $F_{i,G}$ is less than 0.0, then is regenerated according to Eq. (13).

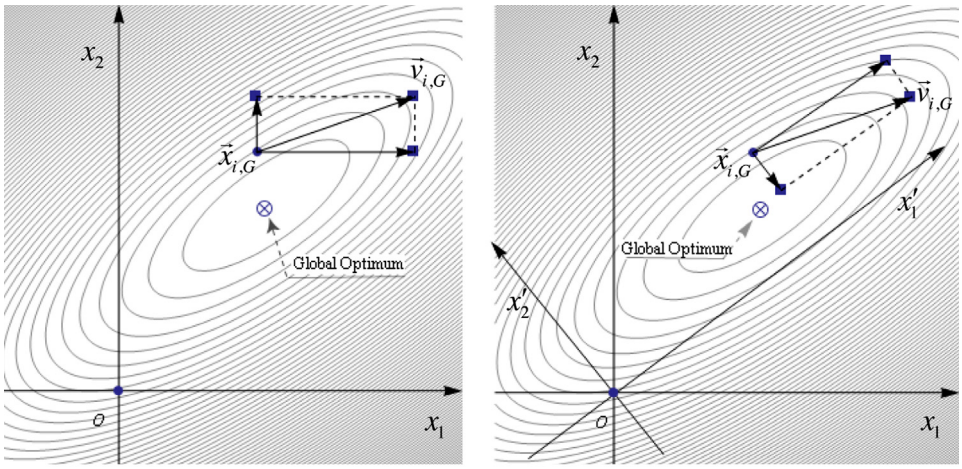


Fig. 1. Crossover in the original coordinate system (i.e., ox_1x_2) and in the Eigen coordinate system (i.e., $ox'_1x'_2$), where $\vec{x}_{i,G}$ is a target vector in the population, $\vec{v}_{i,G}$ is its mutant vector, and the square points denote the possible trial vectors.

The crossover control parameter $CR_{i,G}$ ($i \in \{1, \ldots, NP\}$) is generated using the bimodal distribution composed of two Cauchy distributions as follows:

$$CR_i = \begin{cases} randc_i(0.1, 0.1), & \text{if } rand(0, 1) < 0.5 \\ randc_i(0.95, 0.1), & \text{otherwise} \end{cases} \quad (14)$$

where $rand(0, 1)$ is a uniformly distributed random number between 0 and 1, and $randc_i(a,b)$ is a random number obeying a Cauchy distribution with location parameter a and scale parameter b. If the value of $CR_{i,G}$ is larger than 1.0, then is truncated to 1.0; and if the value of $CR_{i,G}$ is less than 0.0, then is truncated to 0.0.

The scaling factor F has the capability to control the search range of the mutation operator. In CoBiDE, two Cauchy distributions with the same probability (i.e., 0.5) are used for the setting of F. It is necessary to note that, Cauchy distribution with a higher location parameter (i.e., 1.0) tends to produce a bigger value for F which emphasizes the global exploration; however, Cauchy distribution with a relatively lower location parameter (i.e., 0.65) aims at producing a slightly smaller value for F, which focuses on the local exploitation.

On the other hand, two Cauchy distributions with the same probability (i.e., 0.5) are designed for the setting of CR. The Cauchy distribution with a bigger location parameter (i.e., 0.95) means that the trial vector may inherit more information from the mutant vector, which encourages the diversity of the population and the exploration. On the contrary, the Cauchy distribution with a smaller location parameter (i.e., 0.1) signifies that the trial vector may be quite similar to the target vector. In this case, the search will put emphasis on the neighbor of the parent population, which can accelerate the convergence.

Based on the above analysis, the use of Eqs. (13) and (14) is able to achieve an effective tradeoff between the exploration and exploitation. In addition, the scale parameter in both Eqs. (13) and (14) is set to 0.1, which results in the values of F and CR being located in the relatively small neighborhood of the location parameter with a higher probability.

4.3. Framework of CoBiDE

By combining the covariance matrix learning with the bimodal distribution parameter setting, CoBiDE is presented. The pseudocode of CoBiDE has been shown in Fig. 2.

At each generation, for each target vector $\vec{x}_{i,G}$, a mutant vector $\vec{v}_{i,G}$ is generated by making use of the mutation operator (i.e., Eq. (2)). Afterward, if the predefined parameter pb is larger than a random number between 0 and 1, the crossover operator according to the covariance matrix learning is utilized to produce a trial vector $\vec{u}_{i,G}$, otherwise, the crossover operator according to the original coordinate system is exploited to produce a trial vector $\vec{u}_{i,G}$. Moreover, during the evolution, each target vector has its control parameter setting and the control parameter setting is dynamically adapted based on Eqs. (13) and (14).

In the above procedure, we use the parameter pb to adjust the effect of the covariance matrix learning on the performance. The main reason is the following. Although the covariance matrix learning is an effective way to alleviate the dependence of DE on the coordinate system and the interactions among the variables, it is a relatively deterministic and greedy mechanism due to the use of some best individuals of the population to compute the covariance matrix, and as a result, the performance of the algorithm might degrade for some complex problems. Note that the crossover operator according to the original coordinate system has no bias to any special search directions. Consequently, the crossover operator is implemented in the original coordinate system with a probability (1-pb) to encourage the diversity of the population. With respect to CoBiDE, combining these two kinds of crossover can achieve a good tradeoff between diversity and convergence.

5. Experimental study

CoBiDE was tested on 25 benchmark test functions developed for IEEE CEC2005 [9]. These 25 benchmark test functions can be divided into four classes:

(1) Unimodal functions F_1–F_5.
(2) Basic multimodal functions F_6–F_{12}.
(3) Expanded multimodal functions F_{13}–F_{14}.
(4) Hybrid composition functions F_{15}–F_{25}.

Among the above test functions, F_1 and F_9 are separable functions and the others are non-separable functions. Some test functions are rotated using orthogonal matrices to make variables correlated with each other, and the global optima of some test functions are shifted so as to not at the center of the search space. Moreover, F_4 and F_{17} are used to test the robustness of the algorithm on noise. F_{15}–F_{25} are hybrid composition functions which are

Input: *NP*: the number of individuals contained by the population
MAX_FES: maximum number of function evaluations
pb: the probability to execute DE according to the covariance matrix learning
ps: the proportion of the individuals chosen from the current population to calculate the covariance matrix

(1) G=0;
(2) Generate an initial population $P_0 = \{\vec{x}_{1,0},\ldots,\vec{x}_{NP,0}\}$ by randomly sampling from the search space;
(3) Evaluate the objective function values of each individual (i.e., each target vector) in P_0;
(4) *FES*=*NP*; /* *FES* records the number of function evaluations */
(5) Generate the initial scaling factor $F_{i,0}$ and crossover control parameter $CR_{i,0}$ ($i \in \{1,\ldots,NP\}$) for each target vector $\vec{x}_{i,0}$ in the population according to Eq. (13) and Eq. (14), respectively;
(6) **While** *FES*<*MAX_FES*
(7) $P_{G+1} = \phi$;
(8) **For** *i*=1:*NP*
(9) Apply the mutation operator (i.e., Eq. (2)) to produce a mutant vector $\vec{v}_{i,G}$ for the target vector $\vec{x}_{i,G}$;
(10) **End For**
(11) **If** *rand*(0,1)<*pb* /* *rand*(0,1) denotes a uniformly distributed random number between 0 and 1 */
(12) **For** *i*=1:*NP*
(13) Implement the crossover operator according to the covariance matrix learning (i.e., Eqs. (8)-(12)), and produce a trial vector $\vec{u}_{i,G}$;
(14) **End For**
(15) **Else**
(16) **For** *i*=1:*NP*
(17) Implement the crossover operator according to the original coordinate system (i.e., Eq. (3)), and produce a trial vector $\vec{u}_{i,G}$;
(18) **End For**
(19) **End If**
(20) **For** *i*=1:*NP*
(21) Evaluate the objective function value of $\vec{u}_{i,G}$;
(22) **If** $f(\vec{u}_{i,G}) \le f(\vec{x}_{i,G})$
(23) $P_{G+1} = P_{G+1} \cup \vec{u}_{i,G}$;
(24) $F_{i,G+1}$=$F_{i,G}$ and $CR_{i,G+1}$=$CR_{i,G}$;
(25) **Else**
(26) $P_{G+1} = P_{G+1} \cup \vec{x}_{i,G}$;
(27) Generate $F_{i,G+1}$ and $CR_{i,G+1}$ according to Eq. (13) and Eq. (14) for the next generation;
(28) **End If**
(29) **End For**
(30) *FES*=*FES*+*NP*;
(31) G=G+1;
(32) **End While**

Output: the individual with the smallest objective function value in the population

Fig. 2. Pseudocode of CoBiDE.

composed of 10 sub- functions. The details of these 25 benchmark test functions have been given in [9].

In our experiments, the dimension (D) of each test function was set to 30 and each test function was independently run 25 times with 300,000 function evaluations (FES) as the termination criterion. All the experiments are performed on a computer with 2.4 GHz Dual-core Processor and 4.0 GB of RAM in Windows XP. The population size *NP* in CoBiDE was set to 60, $pb = 0.4$, and $ps = 0.5$.

In this section, the mean and standard deviation of the function error value ($f(\vec{x}) - f(\vec{x}^*)$) were calculated over 25 independent runs for each test function, where $\vec{x}$ is the best solution in the population when the algorithm terminates and $\vec{x}^*$ is the global optimal solution. Wilcoxon's rank sum test at a 0.05 significance level was performed to test the statistical significance of the experimental results between two algorithms.

5.1. Comparison with other DE variants

CoBiDE was compared with four other DE variants: JADE [14], jDE [18], SaDE [8], and CoDE [23]. These four algorithms have been briefly introduced in Section 3. JADE and jDE adopt self-adaptive parameter setting, and SaDE uses the normal distribution $N(0.5, 0.3)$ to produce the scaling factor F and adjusts the crossover control parameter CR in a self-adaptive way. For the above four algorithms, we used the same parameter settings as given in their original papers. The experimental results of CoBiDE and other four algorithms are summarized in Table 1. It is necessary to emphasize that the experimental results of JADE, jDE, SaDE, and CoDE were directly taken from [23] to ensure the comparison fair.

Table 1 also records the Cohen's d effect size [31] (within parentheses), which is a simple measure for quantifying the difference between two groups of data. The Cohen's d effect size is independent of the sample size. In general, we call a "small" effect if an effect size is between 0.2 and 0.3, a "medium" effect if an effect size is around 0.5, and a "large" effect if an effect size is from 0.8 to infinity [31]. Concretely, for F_3 the effect size is equal to 1.63 when comparing JADE with CoBiDE, which means that the performance difference between JADE and CoBiDE is large and that JADE exhibits performance improvement. In contrast, for F_{11} the effect size is equal to −10.36 when comparing JADE with CoBiDE, which means that the performance difference between JADE and CoBiDE is also large and that JADE shows performance deterioration. It is necessary to note that for some test functions, the differences between both the mean and the standard deviation are equal to 0 when comparing CoBiDE with another algorithm and, as a result, the corresponding effect size is denoted as NaN in Table 1.

Table 1
Experimental results of JADE, jDE, SaDE, CoDE, and CoBiDE over 25 independent runs on 25 test functions of 30 variables with 300,000 FES. "Mean Error" and "Std Dev" indicate the average and standard deviation of the function error values obtained in 25 runs, respectively. Wilcoxon's rank sum test at a 0.05 significance level is performed between CoBiDE and each of JADE, jDE, SaDE, and CoDE. The effect size is shown in the parentheses.

Function		JADE Mean Error ± Std Dev	jDE Mean Error ± Std Dev	SaDE Mean Error ± Std Dev	CoDE Mean Error ± Std Dev	CoBiDE Mean Error ± Std Dev
Unimodal functions	F_1	0.00E+00 ± 0.00E+00≈ (NaN)	0.00E+00 ± 0.00E+00≈ (NaN)	0.00E+00 ± 0.00E+00≈ (NaN)	0.00E+00 ± 0.00E+00≈ (NaN)	0.00E+00 ± 0.00E+00
	F_2	1.07E−28 ± 1.00E−28+ (0.80)	1.11E−06 ± 1.96E−06− (−0.82)	8.26E−06 ± 1.65E−05− (−0.72)	1.69E−15 ± 3.95E−15+ (0.80)	1.60E−12 ± 2.90E−12
	F_3	8.42E+03 ± 7.26E+03+ (1.63)	1.98E+05 ± 1.10E+05− (−1.46)	4.27E+05 ± 2.08E+05− (−2.37)	1.05E+05 ± 6.25E+04− (−0.56)	7.26E+04 ± 5.64E+04
	F_4	1.73E−16 ± 5.43E−16+ (0.61)	4.40E−02 ± 1.26E−01− (−0.49)	1.77E+02 ± 2.67E+02− (−0.96)	5.81E−03 ± 1.38E−02− (−0.48)	1.16E−03 ± 2.74E−03
	F_5	8.59E−08 ± 5.23E−07+ (0.77)	5.11E+02 ± 4.40E+02− (−1.33)	3.25E+03 ± 5.90E+02− (−7.51)	3.31E+02 ± 3.44E+02− (−0.96)	8.03E+01 ± 1.51E+02
Basic multimodal functions	F_6	1.02E+01 ± 2.96E+01− (−0.50)	2.35E+01 ± 2.50E+01− (−1.35)	5.31E+01 ± 3.25E+01− (−2.36)	1.60E−01 ± 7.85E−01− (−0.22)	4.13E−02 ± 9.21E−02
	F_7	8.07E−03 ± 7.42E−03− (−1.09)	1.18E−02 ± 7.78E−03− (−1.68)	1.57E−02 ± 1.38E−02− (−1.40)	7.46E−03 ± 8.55E−03− (−0.88)	1.77E−03 ± 3.73E−03
	F_8	2.09E+01 ± 1.68E−01− (−0.70)	2.09E+01 ± 4.86E−02− (−0.76)	2.09E+01 ± 4.95E−02− (−0.76)	2.01E+01 ± 1.41E−01+ (2.16)	2.07E+01 ± 3.75E−01
	F_9	0.00E+00 ± 0.00E+00≈ (NaN)	0.00E+00 ± 0.00E+00≈ (NaN)	2.39E−01 ± 4.33E−01− (−0.80)	0.00E+00 ± 0.00E+00≈ (NaN)	0.00E+00 ± 0.00E+00
	F_{10}	2.41E+01 ± 4.61E+00+ (1.69)	5.54E+01 ± 8.46E+00− (−1.43)	4.72E+01 ± 1.01E+01≈ (−0.62)	4.15E+01 ± 1.16E+01≈ (−0.12)	4.41E+01 ± 1.29E+01
	F_{11}	2.53E+01 ± 1.65E+00− (−10.36)	2.79E+01 ± 1.61E+00− (−11.83)	1.65E+01 ± 2.42E+00− (−4.81)	1.18E+01 ± 3.40E+00− (−2.21)	5.62E+00 ± 2.19E+00
	F_{12}	6.15E+03 ± 4.79E+03− (−0.75)	8.63E+03 ± 8.31E+03− (−0.89)	3.02E+03 ± 2.33E+03≈ (−0.02)	3.05E+03 ± 3.80E+03≈ (−0.03)	2.94E+03 ± 3.93E+03
Expanded multimodal functions	F_{13}	1.49E+00 ± 1.09E−01+ (1.46)	1.66E+00 ± 1.35E−01+ (1.24)	3.94E+00 ± 2.81E−01− (−1.61)	1.57E+00 ± 3.27E−01+ (1.31)	2.64E+00 ± 1.13E+00
	F_{14}	1.23E+01 ± 3.11E−01≈ (0)	1.30E+01 ± 2.00E−01− (−1.91)	1.26E+01 ± 2.83E−01− (−0.77)	1.23E+01 ± 4.81E−01≈ (0)	1.23E+01 ± 4.90E−01
Hybrid composition functions	F_{15}	3.51E+02 ± 1.28E+02+ (0.56)	3.77E+02 ± 8.02E+01+ (0.41)	3.76E+02 ± 7.83E+01≈ (0.43)	3.88E+02 ± 6.85E+01≈ (0.27)	4.04E+02 ± 5.03E+01
	F_{16}	1.01E+02 ± 1.24E+02− (−0.30)	7.94E+01 ± 2.96E+01− (−0.17)	8.57E+01 ± 6.94E+01− (−0.21)	7.37E+01 ± 5.13E+01≈ (0.00)	7.38E+01 ± 3.66E+01
	F_{17}	1.47E+02 ± 1.33E+02− (−0.80)	1.37E+02 ± 3.80E+01− (−2.16)	7.83E+01 ± 3.76E+01≈ (−0.20)	6.67E+01 ± 2.12E+01+ (0.29)	7.25E+01 ± 2.02E+01
	F_{18}	9.04E+02 ± 1.03E+00≈ (−0.14)	9.04E+02 ± 1.08E+01≈ (−0.10)	8.68E+02 ± 6.23E+01≈ (0.80)	9.04E+02 ± 1.04E+00− (−0.14)	9.03E+02 ± 1.05E+01
	F_{19}	9.04E+02 ± 8.40E−01≈ (−0.14)	9.04E+02 ± 1.11E+00≈ (−0.14)	8.74E+02 ± 6.22E+01≈ (0.66)	9.04E+02 ± 9.42E−01− (−0.14)	9.03E+02 ± 1.04E+01
	F_{20}	9.04E+02 ± 8.47E−01≈ (0)	9.04E+02 ± 1.10E+00≈ (0)	8.78E+02 ± 6.03E+01+ (0.62)	9.04E+02 ± 9.01E−01− (0)	9.04E+02 ± 5.95E−01
	F_{21}	5.00E+02 ± 4.67E−13≈ (0)	5.00E+02 ± 4.80E−13≈ (0)	5.52E+02 ± 1.82E+02− (−0.41)	5.00E+02 ± 4.88E−13≈ (0)	5.00E+02 ± 4.62E−13
	F_{22}	8.66E+02 ± 1.91E+01≈ (−0.17)	8.75E+02 ± 1.91E+01− (−0.55)	9.36E+02 ± 1.83E+01− (−3.19)	8.63E+02 ± 2.43E+01≈ (−0.04)	8.62E+02 ± 2.80E+01
	F_{23}	5.50E+02 ± 8.05E+01− (−0.29)	5.34E+02 ± 2.77E−04− (0)	5.34E+02 ± 3.57E−03− (0)	5.34E+02 ± 4.12E−04− (0)	5.34E+02 ± 1.30E−04
	F_{24}	2.00E+02 ± 2.85E−14≈ (0)	2.00E+02 ± 2.85E−14≈ (0)	2.00E+02 ± 6.20E−13≈ (0)	2.00E+02 ± 2.85E−14≈ (0)	2.00E+02 ± 2.85E−14
	F_{25}	2.11E+02 ± 7.92E−01− (−1.30)	2.11E+02 ± 7.32E−01− (−1.36)	2.14E+02 ± 2.00E+00− (−2.69)	2.11E+02 ± 9.02E−01− (−1.21)	2.10E+02 ± 7.73E−01
−		9	16	16	11	
+		7	2	1	4	
≈		9	7	8	10	

"−", "+", and "≈" denote that the performance of the corresponding algorithm is worse than, better than, and similar to that of CoBiDE, respectively.

Table 2
Results of the multiple-problem Wilcoxon's test for JADE, jDE, SaDE, CoDE, and CoBiDE at a 0.05 significance level and at a 0.1 significance level.

Algorithm	$R+$	$R-$	p-Value	$\alpha = 0.05$	$\alpha = 0.1$
CoBiDE vs JADE	209.0	116.0	0.206006	No	No
CoBiDE vs jDE	263.0	37.0	0.001183	Yes	Yes
CoBiDE vs SaDE	252.5	72.5	0.014889	Yes	Yes
CoBiDE vs CoDE	213.0	87.0	0.069634	No	Yes

Table 3
Ranking of JADE, jDE, SaDE, CoDE, and CoBiDE according to the statistical test of the Friedman test.

Algorithm	Ranking
CoBiDE	2.08
CoDE	2.64
JADE	2.84
SaDE	3.64
jDE	3.8

The last three lines of Table 1 summarize the experimental results:

(1) Unimodal functions F_1–F_5: JADE exhibits the best performance on five unimodal functions among the five algorithms. Evidently, the greedy mutation operator, i.e., current-to-$p$best/1, results in the fast convergence speed and high convergence precision of JADE under these conditions. CoBiDE is outperformed by JADE on four test functions and surpasses jDE, SaDE, and CoDE on four, four, and three test functions, respectively. jDE and SaDE cannot show better performance than CoBiDE on any test functions and CoDE performs better than CoBiDE on only one test function. Therefore, the performance of CoBiDE is the second best in terms of these five unimodal functions.
(2) Basic multimodal functions F_6–F_{12}: Clearly, CoBiDE has the best performance on this kind of test functions. CoBiDE has an edge over JADE, jDE, SaDE, and CoDE on five, six, five, and three test functions, respectively. JADE and CoDE are statistically better than CoBiDE on one test function, and jDE and SaDE cannot outperform CoBiDE on any test functions. The outstanding performance of CoBiDE can be attributed to its capability to balance the exploration and exploitation.
(3) Expanded multimodal functions F_{13}–F_{14}: The mean function error values of all the algorithms are of the same order of magnitude on F_{13} and F_{14}. JADE and CoDE are statistically better than CoBiDE. CoBiDE exhibits the similar performance with jDE. In addition, CoBiDE outperforms SaDE on these two test functions.
(4) Hybrid composition functions F_{15}–F_{25}: The solution of these 11 test functions is much more difficult than that of other test functions. For these 11 test functions, the results provided by the five algorithms are far way from the global optima. However, from Table 1, we can still observe that the performance of CoBiDE is superior to that of the other four algorithms according to the Wilcoxon's rank sum test.

According to the last three lines of Table 1, overall CoBiDE is the best among the five algorithms. For five unimodal functions, CoBiDE is ranked the second, and for basic multimodal functions and hybrid composition functions, CoBiDE is more reliable than others. The superior performance of CoBiDE stems from two aspects: (1) the bimodal distribution parameter setting is capable of motivating the population toward promising directions, and (2) the covariance matrix learning is able to accelerate the convergence by exploiting the information provided by some potential individuals.

In addition, we also performed the multiple-problem Wilcoxon's test [32] to check the behaviors of the above five algorithms. It is necessary to emphasize that the multiple-problem Wilcoxon's test was accomplished in this paper by using the KEEL software [33]. Table 2 summarizes the statistical analysis results. From Table 2, we can see that CoBiDE provides higher R+ values than R− values in all the cases. According to the Wilcoxon's test at $\alpha = 0.05$, the significant differences can be observed in two cases (i.e., CoBiDE vs jDE and CoBiDE vs SaDE). When $\alpha = 0.1$, the significant differences can be observed in three cases (i.e., CoBiDE vs jDE, CoBiDE vs SaDE, and CoBiDE vs CoDE), which means that CoBiDE is significantly better than jDE, SaDE, and CoDE on 25 test functions at $\alpha = 0.1$.

To further detect the significant differences between CoBiDE and the four competitors, the Friedman's test was carried out, in which Bonferroni–Dunn's procedure was used as a post hoc procedure. Again, the Friedman's test was implemented based on the KEEL software [33]. Table 3 summarizes the ranking of the five algorithms obtained by the Friedman's test. As shown in Table 3, CoBiDE has the best ranking among the five algorithms on 25 test functions.

Since all the five compared algorithms are the DE variants, one may be interested in the execution time of them on different test functions. To this end, we recorded the average runtime of each algorithm on each test function over 25 independent runs in Table 4. In order to compare the average runtime, we used the acceleration rate (AR). For each test function, AR is equal to the average runtime of CoBiDE divided by the average runtime of another algorithm. AR > 1 and AR < 1 mean that CoBiDE is faster and slower than another corresponding algorithm, respectively. The last row of Table 4 gives the average AR values. According to the average AR values, it is evident that JADE and jDE are faster than CoBiDE. In contrast, SaDE and CoDE are slower than CoBiDE. Moreover, based on our observation, the average AR values are 0.69 and 0.53 for 12 test functions (i.e., F_1–F_{10}, F_{13}, and F_{14}) when comparing CoBiDE with JADE and jDE, respectively. For these 12 test functions, the computational cost of the function evaluation is relatively cheap, and thus, the computing of the covariance matrix leads to the additional burden of the runtime of CoBiDE. However, for the other 13 test functions (i.e., F_{11}–F_{12} and F_{15}–F_{25}), the average AR values are 0.89 and 0.88 when comparing CoBiDE with JADE and jDE, respectively. For these 13 test functions, since the function evaluation is time-consuming, the overhead of computing the covariance matrix in CoBiDE seems to be trivial. Under these conditions, CoBiDE, JADE, and jDE have the similar average runtime. It is necessary to point out that we directly run the codes of the other four algorithms provided by the developers and the programming techniques of the developers also have a significant effect on the runtime.

The evolution of the mean function error values of the five algorithms in some typical test functions has been shown in Figs. 3 and 4.

5.2. Comparison with other state-of-the-art EAs

CoBiDE was also compared with three other EAs: CLPSO [34], CMA-ES [25], and GL-25 [35]. CLSPO, proposed by Liang et al., is an improved version of particle swarm optimization (PSO). In CLPSO, a novel learning strategy is proposed, in which all other particles' historical best information is used to update a particle's velocity. CMA-ES, proposed by Hansen and Ostermeier, is an evolution strategy (ES) based on completely derandomized self-adaptation. GL-25, proposed by Garcia-Martinez et al., is a global and local real-coded genetic algorithm (GA) based on parent-centric crossover operators. The reasons of the selection of these three algorithms in comparison are twofold: (1) CLPSO, CMA-ES, and GL-25 represent the state-of-the-art in PSO, ES, and GA, respectively. According to the Google Scholar Citation, as of December 20, 2013, the number of citations of CLPSO, CMA-ES, and GL-25 is 1011, 1332, and 81

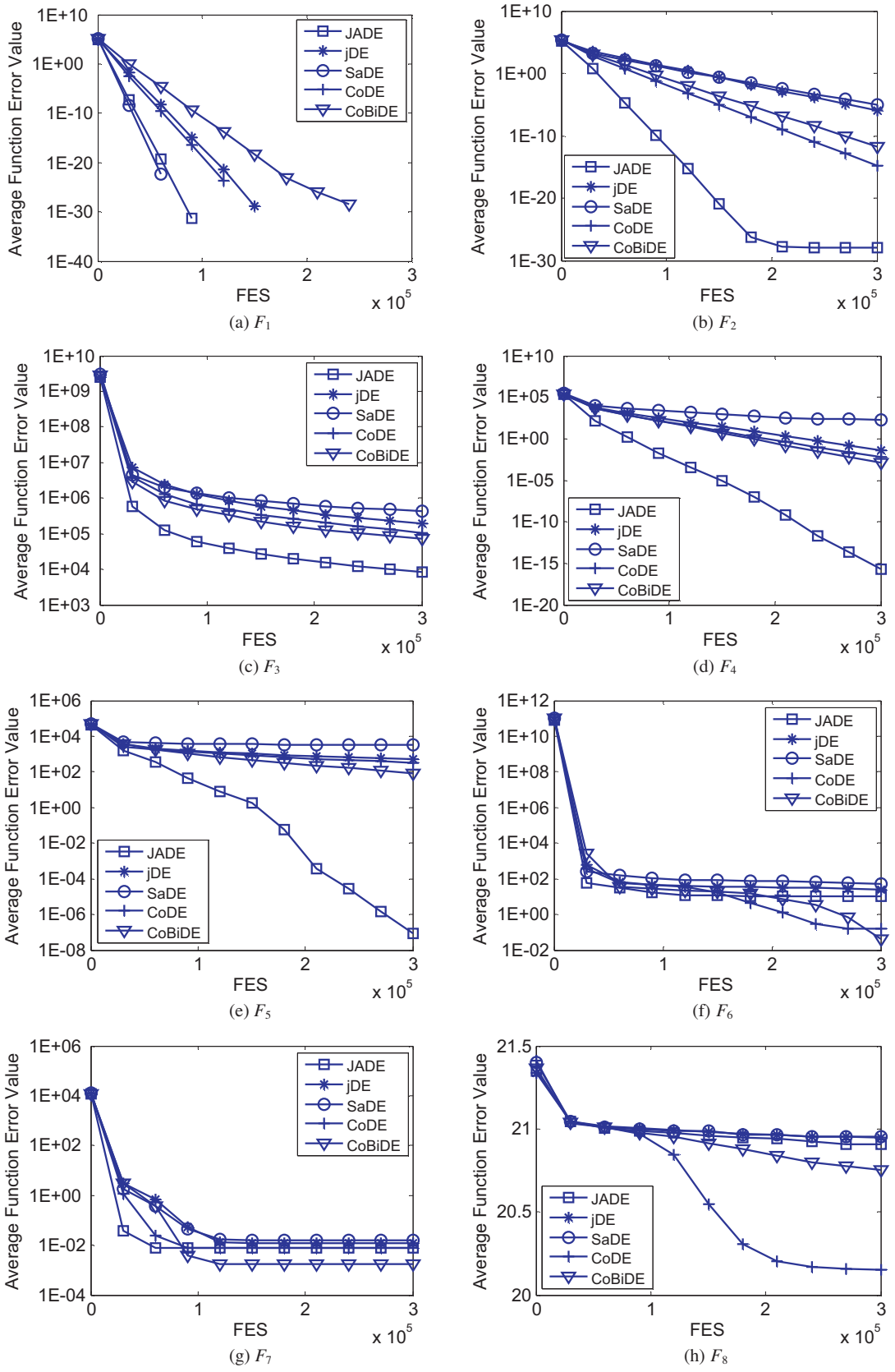


Fig. 3. Evolution of the mean function error values derived from JADE, jDE, SaDE, CoDE, and CoBiDE versus the number of FES on F_1, F_2, F_3, F_4, F_5, F_6, F_7, and F_8.

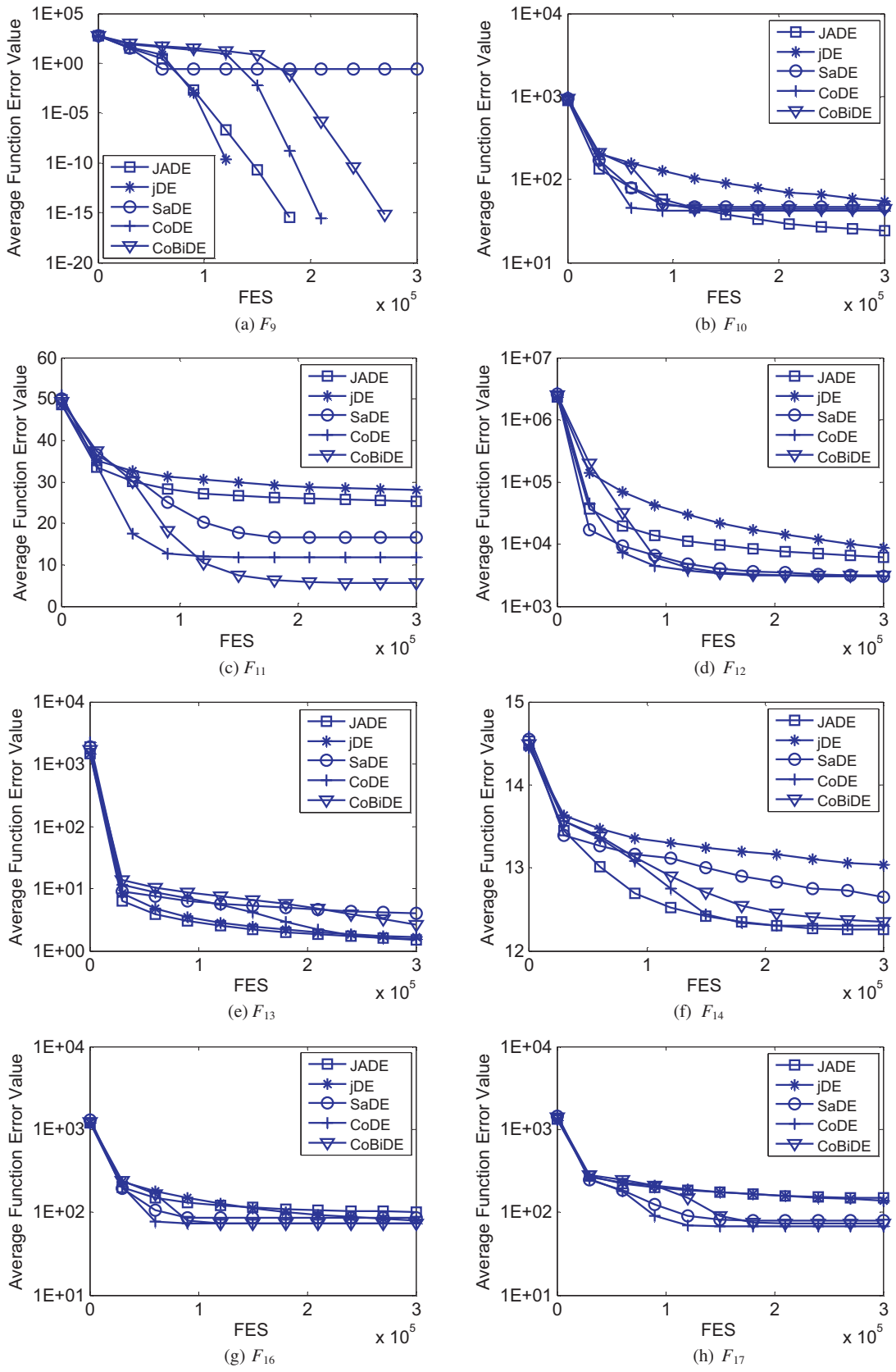


Fig. 4. Evolution of the mean function error values derived from JADE, jDE, SaDE, CoDE, and CoBiDE versus the number of FES on F_9, F_{10}, F_{11}, F_{12}, F_{13}, F_{14}, F_{16}, and F_{17}.

Table 4
Comparison of the average runtime (in s) of JADE, jDE, SaDE, CoDE, and CoBiDE for each test function. AR denotes the acceleration rate and the last row of the table represents the average AR.

Function		JADE	jDE	SaDE	CoDE	CoBiDE
Unimodal functions	F_1	3.74	2.43	39.14	8.93	4.91
	F_2	3.77	2.82	35.65	9.53	5.64
	F_3	3.82	2.83	36.51	12.49	5.26
	F_4	3.83	3.07	32.63	9.89	5.91
	F_5	4.59	3.74	36.21	11.12	6.28
Basic multimodal functions	F_6	3.36	2.37	34.31	9.29	5.07
	F_7	3.48	2.73	32.36	8.28	5.03
	F_8	4.14	3.52	36.36	12.33	5.94
	F_9	3.47	2.69	35.40	9.82	5.36
	F_{10}	4.07	3.21	35.50	9.98	5.31
	F_{11}	67.34	67.31	109.46	80.76	70.74
	F_{12}	19.56	19.23	52.92	28.66	24.40
Expanded multimodal functions	F_{13}	4.22	3.23	33.65	10.03	6.57
	F_{14}	5.00	4.28	39.34	13.74	7.21
Hybrid composition functions	F_{15}	152.24	155.58	245.17	179.93	171.61
	F_{16}	158.03	152.60	215.81	175.75	169.60
	F_{17}	149.77	153.18	222.27	209.18	165.98
	F_{18}	162.03	160.15	227.57	230.69	176.28
	F_{19}	163.59	155.65	212.93	193.63	180.76
	F_{20}	162.14	158.16	226.92	186.80	177.43
	F_{21}	159.13	155.38	227.15	178.51	183.83
	F_{22}	204.95	203.20	277.54	233.41	222.01
	F_{23}	158.61	157.33	242.07	183.79	180.15
	F_{24}	104.40	107.61	199.99	128.66	125.99
	F_{25}	120.10	115.51	215.87	136.15	131.38
Average AR		0.80	0.72	3.77	1.45	

respectively, and (2) their performance is very competitive. Table 5 summarizes the experimental results of CoBiDE and the above three algorithms. The parameter settings of CLPSO, CMA-ES, and CLPSO were the same as in their original papers and the experimental result of them were directly taken from [23] to make the comparison fair.

From Table 5, it is evident that, overall, CoBiDE is the best among the four compared algorithms in a statistically significant fashion.

Table 5
Experimental results of CLPSO, CMA-ES, GL-25, and CoBiDE over 25 independent runs on 25 test functions of 30 variables with 300,000 FES. "Mean Error" and "Std Dev" indicate the average and standard deviation of the function error values obtained in 25 runs, respectively. Wilcoxon's rank sum test at a 0.05 significance level is performed between CoBiDE and each of CLPSO, CMA-ES, and GL-25.

Function		CLPSO Mean Error ± Std Dev	CMA-ES Mean Error ± Std Dev	GL-25 Mean Error ± Std Dev	CoBiDE Mean Error ± Std Dev
Unimodal functions	F_1	0.00E+00 ± 0.00E+00≈	1.58E−25 ± 3.35E−26−	5.60E−27 ± 1.76E−26−	0.00E+00 ± 0.00E+00
	F_2	8.40E+02 ± 1.90E+02−	1.12E−24 ± 2.93E−25+	4.04E+01 ± 6.28E+01−	1.60E−12 ± 2.90E−12
	F_3	1.42E+07 ± 4.19E+06−	5.54E−21 ± 1.69E−21+	2.19E+06 ± 1.08E+06−	7.26E+04 ± 5.64E+04
	F_4	6.99E+03 ± 1.73E+03−	9.15E+05 ± 2.16E+06−	9.07E+02 ± 4.25E+02−	1.16E−03 ± 2.74E−03
	F_5	3.86E+03 ± 4.35E+02−	2.77E−10 ± 5.04E−11+	2.51E+03 ± 1.96E+02−	8.03E+01 ± 1.51E+02
Basic multimodal functions	F_6	4.16E+00 ± 3.48E+00−	4.78E−01 ± 1.32E+00−	2.15E+01 ± 1.17E+00−	4.13E−02 ± 9.21E−02
	F_7	4.51E−01 ± 8.47E−02−	1.82E−03 ± 4.33E−03≈	2.78E−02 ± 3.62E−02−	1.77E−03 ± 3.73E−03
	F_8	2.09E+01 ± 4.41E−02−	2.03E+01 ± 5.72E−01+	2.09E+01 ± 5.94E−02−	2.07E+01 ± 3.75E−01
	F_9	0.00E+00 ± 0.00E+00≈	4.45E+02 ± 7.12E+01−	2.45E+01 ± 7.35E+00−	0.00E+00 ± 0.00E+00
	F_{10}	1.04E+02 ± 1.53E+01−	4.63E+01 ± 1.16E+01≈	1.42E+02 ± 6.45E+01−	4.41E+01 ± 1.29E+01
	F_{11}	2.60E+01 ± 1.63E+00−	7.11E+00 ± 2.14E+00−	3.27E+01 ± 7.79E+00−	5.62E+00 ± 2.19E+00
	F_{12}	1.79E+04 ± 5.24E+03−	1.26E+04 ± 1.74E+04−	6.53E+04 ± 4.69E+04−	2.94E+03 ± 3.93E+03
Expanded multimodal functions	F_{13}	2.06E+00 ± 2.15E−01+	3.43E+00 ± 7.60E−01−	6.23E+00 ± 4.88E+00−	2.64E+00 ± 1.13E+00
	F_{14}	1.28E+01 ± 2.48E−01−	1.47E+01 ± 3.31E−01−	1.31E+01 ± 1.84E−01−	1.23E+01 ± 4.90E−01
Hybrid composition functions	F_{15}	5.77E+01 ± 2.76E+01+	5.55E+02 ± 3.32E+02−	3.04E+02 ± 1.99E+01+	4.04E+02 ± 5.03E+01
	F_{16}	1.74E+02 ± 2.82E+01−	2.98E+02 ± 2.08E+02−	1.32E+02 ± 7.60E+01−	7.38E+01 ± 3.66E+01
	F_{17}	2.46E+02 ± 4.81E+01−	4.43E+02 ± 3.34E+02−	1.61E+02 ± 6.80E+01−	7.25E+01 ± 2.02E+01
	F_{18}	9.13E+02 ± 1.42E+00−	9.04E+02 ± 3.01E−01−	9.07E+02 ± 1.48E+00−	9.03E+02 ± 1.05E+01
	F_{19}	9.14E+02 ± 1.45E+00−	9.16E+02 ± 6.03E+01−	9.06E+02 ± 1.24E+00−	9.03E+02 ± 1.04E+01
	F_{20}	9.14E+02 ± 3.62E+00−	9.04E+02 ± 2.71E−01+	9.07E+02 ± 1.35E+00−	9.04E+02 ± 5.95E−01
	F_{21}	5.00E+02 ± 3.39E−13≈	5.00E+02 ± 2.68E−12−	5.00E+02 ± 4.83E−13≈	5.00E+02 ± 4.62E−13
	F_{22}	9.72E+02 ± 1.20E+01−	8.26E+02 ± 1.46E+01+	9.28E+02 ± 7.04E+01−	8.62E+02 ± 2.80E+01
	F_{23}	5.34E+02 ± 2.19E−04−	5.36E+02 ± 5.44E+00−	5.34E+02 ± 4.66E−04−	5.34E+02 ± 1.30E−04
	F_{24}	2.00E+02 ± 1.49E−12−	2.12E+02 ± 6.00E+01−	2.00E+02 ± 5.52E−11−	2.00E+02 ± 2.85E−14
	F_{25}	2.00E+02 ± 1.96E+00+	2.07E+02 ± 6.07E+00≈	2.17E+02 ± 1.36E−01−	2.10E+02 ± 7.73E−01
−		19	16	23	
+		3	6	1	
≈		3	3	1	

"−", "+", and "≈" denote that the performance of the corresponding algorithm is worse than, better than, and similar to that of CoBiDE, respectively.

Specifically, CoBiDE outperforms CLPSO on 19 test functions and is worse than CLPSO on three test functions. CMA-ES surpasses CoBiDE on three unimodal functions; however, CoBiDE is significantly better than CMA-ES on three other types of test functions. Compared with GL-25, CoBiDE shows better and worse performance on 23 test functions and one test function, respectively.

In addition, some interesting phenomena can be observed according to the experimental results in Table 5. For separable functions (i.e., F_1 and F_9), the performance of CLPSO is significantly better than that of the other algorithms except for CoBiDE. Moreover, CLPSO also outperforms the other algorithms on F_{15} which is separable near the global optimum [9]. The superiority of CLPSO in separable functions is mainly due to the dimension-wise updating rules for velocity and position in PSO. CMA-ES performs quite well on some unimodal functions, which means the convergence speed of CMA-ES is very fast. Moreover, CMA-ES outperforms the other algorithms on test functions with high condition numbers, i.e., F_3 and F_{22}. It is because CMA-ES has the capability to adapt the population distribution according to the landscape of test functions. However, the performance of CMA-ES is not good when solving some multimodal functions, especially for test functions with noise, i.e., F_4 and F_{17}. Therefore, we can conclude that CMA-ES is sensitive to the noise. In contrast, CoBiDE shows the best performance on test functions with noise.

Tables 6 and 7 also present the statistical analysis results according to the multiple-problem Wilcoxon's test and the Friedman's test, respectively. It can be seen from Table 6 that CoBiDE obtains higher R+ values than R− values in all the cases. Furthermore, the p values of all the cases are less than 0.05. On the other hand, the experimental results in Table 7 indicate that CoBiDE has the best ranking among the four compared algorithms. In summary, the above comparison clearly demonstrates that CoBiDE is significantly better than the three competitors.

Table 6
Results of the multiple-problem Wilcoxon's test for CLPSO, CMA-ES, GL-25, and CoBiDE at a 0.05 significance level and at a 0.1 significance level.

Algorithm	$R+$	$R-$	p-Value	$\alpha = 0.05$	$\alpha = 0.1$
CoBiDE vs CLPSO	257.0	43.0	0.001834	Yes	Yes
CoBiDE vs CMA-ES	241.5	83.5	0.032428	Yes	Yes
CoBiDE vs GL-25	279.5	20.5	0.000193	Yes	Yes

Table 7
Ranking of CLPSO, CMA-ES, GL-25, and CoBiDE according to the statistical test of the Friedman test.

Algorithm	Ranking
CoBiDE	1.62
CMA-ES	2.6
CLPSO	2.84
GL-25	2.94

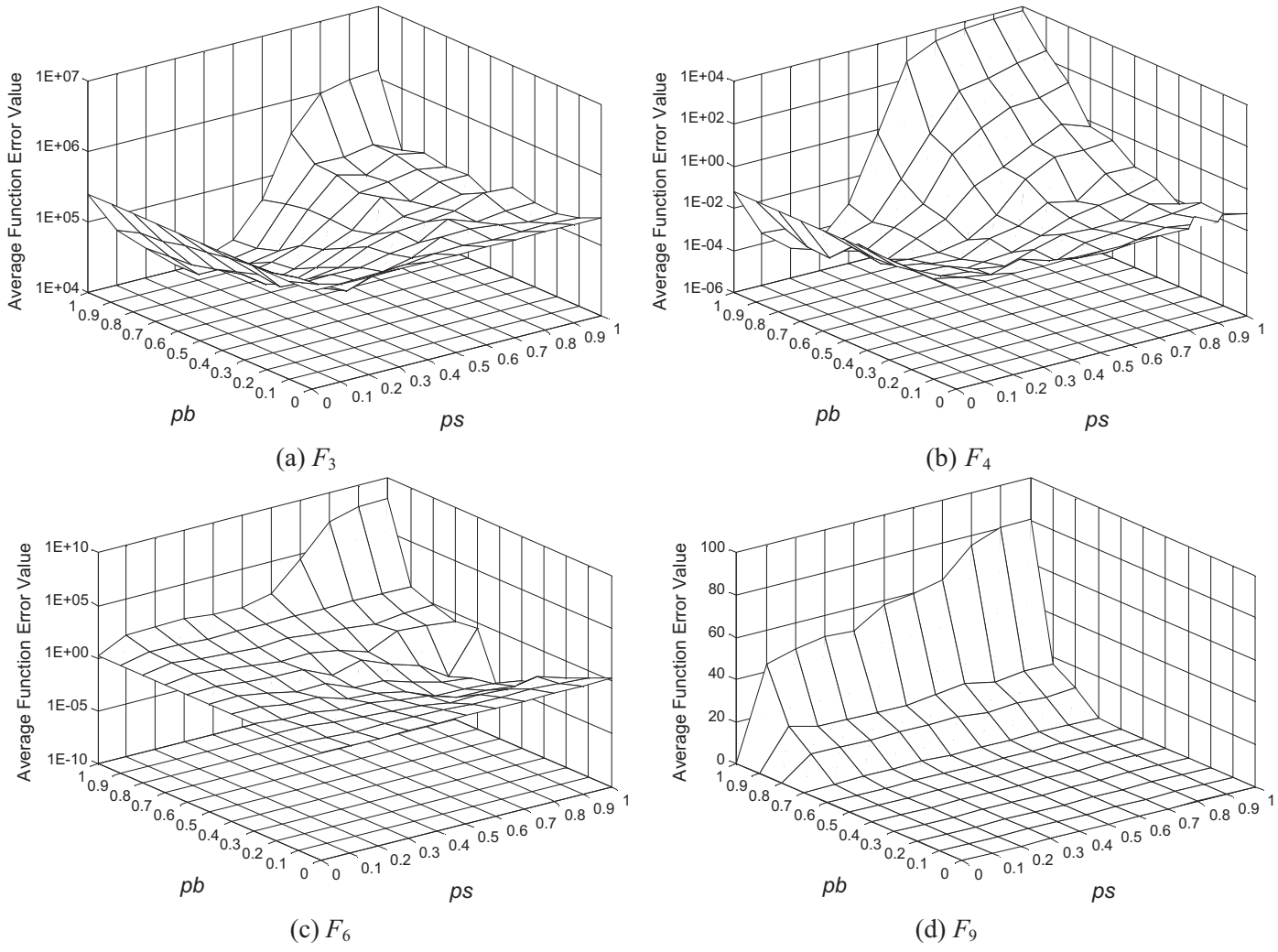
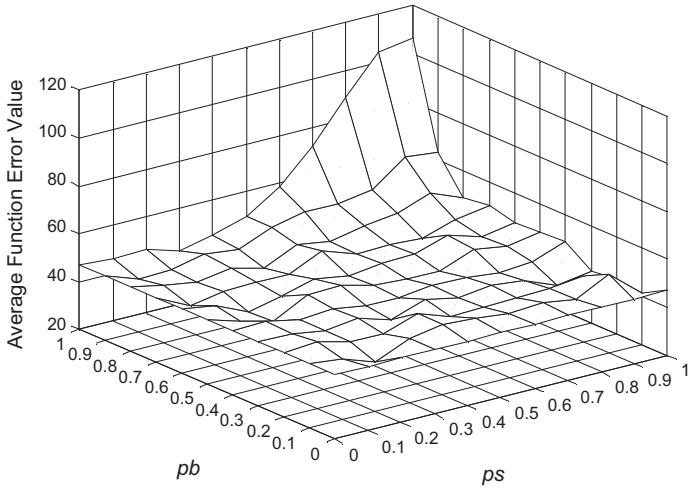
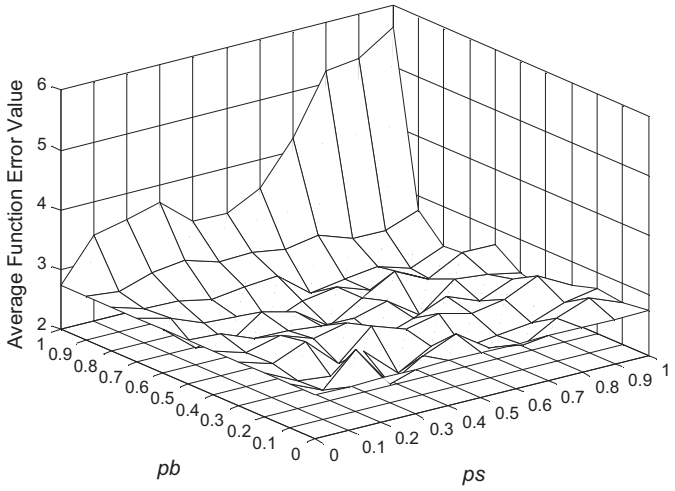


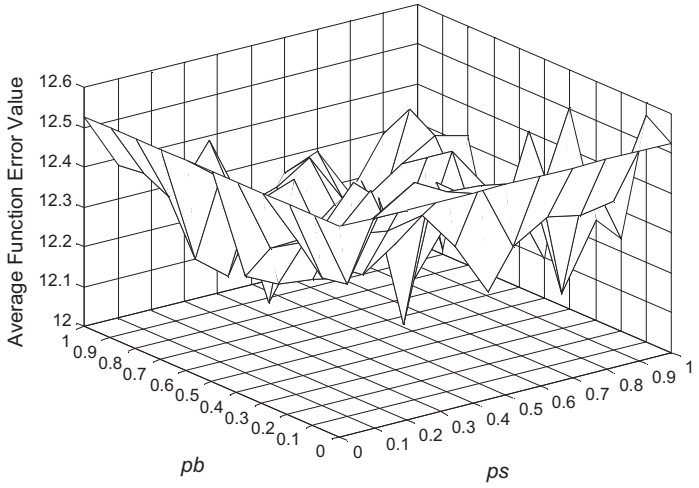
Fig. 5. The average function error values of CoBiDE with different combinations of *pb* and *ps*.



(e) F_{10}



(f) F_{13}



(g) F_{14}

Fig. 5. (*Continued*).

5.3. The effectiveness of the two components in CoBiDE

As mentioned previously, CoBiDE includes two main components: the covariance matrix learning and the bimodal distribution parameter setting. The aim of this subsection is to verify the effectiveness of the above two components. To this end, two additional experiments were executed for 25 benchmark test functions. In the first experiment, CoBiDE only adopts the covariance matrix learning and the bimodal distribution parameter setting is not used (denoted as CoBiDE-1). In this case, like [18,20], F and CR were fixed to 0.5 and 0.9 during the evolution, respectively. In addition, in the second experiment, CoBiDE only adopts the bimodal distribution parameter setting and the covariance matrix learning is ignored (denoted as CoBiDE-2). It is necessary to note that for CoBiDE-2, Steps 11–15 and Step 19 in Fig. 3 can be eliminated and only the crossover operator of the original DE (i.e., Eq. (3)) is employed.

For each test function, 25 independent runs were implemented and the maximum number of FES was set to 300,000. The experimental results of CoBiDE-1, CoBiDE-2, and CoBiDE have been shown in Table 8.

From Table 8, CoBiDE surpasses CoBiDE-1 on 18 test functions. We attribute the above phenomenon to the fact that the fixed parameter setting could not adjust the search behavior to suit different landscapes, and that the bimodal distribution parameter setting is more effective to balance the exploration and exploitation during the evolution. In addition, CoBiDE-1 outperforms CoBiDE on three test functions (i.e., F_4, F_{13}, and F_{19}). According to our further experiments, we found out that the parameter setting of $F = 0.5$ and $CR = 0.9$ provides best or near-best performance for these three test functions, which means that the above parameter setting happens to be very suitable for these three test functions.

In addition, compared with CoBiDE, CoBiDE-2 shows worse performance on 13 test functions, and cannot show better performance on any test functions. It is not difficult to understand, since the covariance matrix learning is not dependent on the coordinate system when implementing the crossover operator. As a result, it has the capability to adapt the search according to different landscapes. Moreover, once some potential regions have been located, it can accelerate the convergence speed and enhance the convergence accuracy of the population for different kinds of test functions, due to the use of the population information to construct more suitable coordinate system. It is also interesting to note that for 12 test functions (F_1, F_8–F_{10}, F_{12}–F_{13}, F_{15}, F_{20}–F_{23}, and F_{25}), the performance differences between CoBiDE and CoBiDE-2 are marginal. These 12 test functions can be divided into two categories: F_1 and F_9 belong to the first category, and the remaining 10 test functions belong to the second category. For F_1 and F_9, both CoBiDE and CoBiDE-2 can consistently reach the global optimum, and thus,

Table 8
Experimental results of CoBiDE-1, CoBiDE-2, and CoBiDE over 25 independent runs on 25 test functions of 30 variables with 300,000 FES. "Mean Error" and "Std Dev" indicate the average and standard deviation of the function error values obtained in 25 runs, respectively. Wilcoxon's rank sum test at a 0.05 significance level is performed between CoBiDE and each of CoBiDE-1 and CoBiDE-2.

Function		CoBiDE-1 Mean Error ± Std Dev	CoBiDE-2 Mean Error ± Std Dev	CoBiDE Mean Error ± Std Dev
Unimodal functions	F_1	1.54E−28 ± 1.35E−28−	0.00E+00 ± 0.00E+00≈	0.00E+00 ± 0.00E+00
	F_2	3.95E−12 ± 5.32E−12≈	2.07E−06 ± 3.69E−06−	1.60E−12 ± 2.90E−12
	F_3	3.25E+05 ± 1.84E+05−	2.46E+05 ± 1.45E+05−	7.26E+04 ± 5.64E+04
	F_4	5.43E−04 ± 1.28E−03+	6.39E−02 ± 6.77E−02−	1.16E−03 ± 2.74E−03
	F_5	7.58E+02 ± 5.56E+02−	1.29E+02 ± 2.67E+02−	8.03E+01 ± 1.51E+02
Basic multimodal functions	F_6	4.09E+01 ± 3.60E+01−	1.66E+00 ± 1.02E+00−	4.13E−02 ± 9.21E−02
	F_7	2.05E−02 ± 1.79E−02−	3.64E−03 ± 6.78E−03−	1.77E−03 ± 3.73E−03
	F_8	2.10E+01 ± 3.81E−02−	2.07E+01 ± 3.74E−01≈	2.07E+01 ± 3.75E−01
	F_9	1.56E+01 ± 7.56E+00−	0.00E+00 ± 0.00E+00≈	0.00E+00 ± 0.00E+00
	F_{10}	1.38E+02 ± 5.78E+01−	4.70E+01 ± 1.34E+01≈	4.41E+01 ± 1.29E+01
	F_{11}	1.62E+01 ± 1.24E+01−	7.82E+00 ± 2.95E+00−	5.62E+00 ± 2.19E+00
	F_{12}	4.35E+03 ± 4.16E+03−	3.45E+03 ± 3.14E+03≈	2.94E+03 ± 3.93E+03
Expanded multimodal functions	F_{13}	1.14E+01 ± 4.47E+00+	2.74E+00 ± 1.00E+00≈	2.64E+00 ± 1.13E+00
	F_{14}	1.31E+01 ± 2.46E−01−	1.25E+01 ± 5.39E−01−	1.23E+01 ± 4.90E−01
Hybrid composition functions	F_{15}	3.82E+02 ± 1.12E+02≈	3.76E+02 ± 8.31E+01≈	4.04E+02 ± 5.03E+01
	F_{16}	1.16E+02 ± 7.27E+01−	1.03E+02 ± 9.07E+01−	7.38E+01 ± 3.66E+01
	F_{17}	2.41E+02 ± 5.64E+01−	7.90E+01 ± 1.90E+01−	7.25E+01 ± 2.02E+01
	F_{18}	9.06E+02 ± 1.45E+00−	9.04E+02 ± 2.76E−01−	9.03E+02 ± 1.05E+01
	F_{19}	9.01E+02 ± 2.10E+01+	9.04E+02 ± 2.62E−01−	9.03E+02 ± 1.04E+01
	F_{20}	9.05E+02 ± 1.38E+00−	9.04E+02 ± 2.55E−01≈	9.04E+02 ± 5.95E−01
	F_{21}	5.24E+02 ± 8.31E+01−	5.00E+02 ± 8.84E−14≈	5.00E+02 ± 4.62E−13
	F_{22}	8.84E+02 ± 1.65E+01−	8.56E+02 ± 2.75E+01≈	8.62E+02 ± 2.80E+01
	F_{23}	5.50E+02 ± 8.05E+01−	5.34E+02 ± 3.53E−04≈	5.34E+02 ± 1.30E−04
	F_{24}	2.00E+02 ± 2.90E−14≈	2.00E+02 ± 1.03E−12−	2.00E+02 ± 2.85E−14
	F_{25}	2.10E+02 ± 5.84E−01≈	2.10E+02 ± 4.61E−01≈	2.10E+02 ± 7.73E−01
−		18	13	
+		3	0	
≈		4	12	

"−", "+", and "≈" denote that the performance of the corresponding algorithm is worse than, better than, and similar to that of CoBiDE, respectively.

the performance differences between CoBiDE and CoBiDE-2 are not significant. In addition, since CoBiDE might be easily trapped into a local optimum and the covariance matrix learning cannot help the population jump out of the local optimum, the insignificant performance differences occur for CoBiDE and CoBiDE-2 on the remaining 10 test functions.

From Table 8, we can conclude that the above two components can benefit each other to enhance the performance of DE. Indeed, the bimodal distribution parameter setting achieves high reliability and the covariance matrix learning results in fast convergence.

5.4. Sensitivity in relation to the parameters pb and ps

CoBiDE contains two parameters *pb* and *ps*. The former controls the computational resource assigned to the covariance matrix learning and the latter controls the number of individuals for computing the covariance matrix.

In order to investigate the sensitivity of the above two parameters, we tested CoBiDE with different *pb*: 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, and 1, and different *ps*: 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, and 1.0. Seven test functions (i.e., F_3, F_4, F_6, F_9, F_{10}, F_{13}, and F_{14}) were selected to test the performance of CoBiDE with different combinations of *pb* and *ps*. These test functions involve shifted problems, problems with noise, rotated problems, and high conditioned problems. The dimension was set to 30 for all the test functions and the maximum number of FES was set to 300,000. Fig. 5 shows the average function error values of CoBiDE with different combinations of *pb* and *ps*.

Generally speaking, a larger value of *pb* may discourage the diversity of the population, however, if the value of *pb* is too small, the covariance matrix learning cannot play its role in solving problems with high variable correlation. On the other hand, if *ps* is set to a larger value, the randomness of the population may cause side effect on the computation of the covariance matrix. However, if the value of *ps* is too small, the chosen individuals cannot reflect the statistical information of the population. Therefore, moderate values should be chosen for these two parameters in order to achieve competitive performance.

From Fig. 5, we can observe that, actually, CoBiDE is not sensitive to these two parameters, and that *pb* and *ps* can be chosen from a relatively large range to achieve competitive performance for CoBiDE. In general, the value of *pb* is recommended in the interval [0.2, 0.7] and the value of *ps* is recommended in the interval [0.3, 0.7].

5.5. Real-world application problems

Besides the above 25 benchmark test functions, eight real-world engineering optimization problems chosen from different fields including radar system, power systems, hydrothermal scheduling, spacecraft trajectory optimization, etc., were used to evaluate the performance of CoBiDE in this subsection. These eight real-world engineering optimization problems (denoted as P_1–P_8 in this paper) are problems T01, T06, T08, T10.1, T11.4, T12.1, T12.2, and T13 collected for the 2011 IEEE Congress on Evolutionary Computation (IEEE CEC2011) [10], respectively, which exhibit different complex characteristics and are very difficult to solve. For each problem, 25 independently runs were implemented with 150,000 FES as the termination criterion. The parameter settings of CoBiDE were the same with those for the 25 benchmark test functions, i.e., $NP = 60$, $pb = 0.4$, and $ps = 0.5$. In addition, the parameter settings of JADE, jDE, SaDE, and CoDE were the same as in their original papers.

Table 9 summarizes the mean and standard deviation of the objective function values over 25 independent runs for each problem. In order to have statistically sound conclusions, Wilcoxon's rank sum test at a 0.05 significance level was conducted on the

Table 9
Experimental results of JADE, jDE, SaDE, CoDE, and CoBiDE over 25 independent runs on eight real-world engineering optimization problems with 150,000 FES. "Mean Value" and "Std Dev" indicate the average and standard deviation of the objective function values obtained in 25 runs, respectively. Wilcoxon's rank sum test at a 0.05 significance level is performed between CoBiDE and each of JADE, jDE, SaDE, and CoDE.

Problem	JADE Mean Value ± Std Dev	jDE Mean Value ± Std Dev	SaDE Mean Value ± Std Dev	CoDE Mean Value ± Std Dev	CoBiDE Mean Value ± Std Dev
P_1	4.63E−01 ± 8.04E−01−	3.49E−01 ± 6.80E−01−	0.00E+00 ± 0.00E+00≈	4.06E−01 ± 2.03E+00−	0.00E+00 ± 0.00E+00
P_2	1.17E+00 ± 1.00E−01−	1.35E+00 ± 7.59E−02−	1.09E+00 ± 2.39E−01−	6.82E−01 ± 1.09E−01−	5.97E−01 ± 1.01E−01
P_3	2.04E+03 ± 4.89E+02−	2.04E+03 ± 4.50E+02−	6.48E+03 ± 7.69E+03−	1.95E+03 ± 4.97E+02≈	1.74E+03 ± 3.49E+02
P_4	5.24E+04 ± 4.92E+02≈	5.78E + 04 ± 3.74E + 03−	5.61E + 04 ± 1.51E + 04≈	5.22E + 04 ± 4.99E + 02≈	5.23E + 04 ± 6.27E + 02
P_5	1.32E+05 ± 5.09E+03≈	1.32E+05 ± 2.44E+03−	1.32E+05 ± 1.66E+03−	1.42E+05 ± 2.43E+03−	1.28E+05 ± 1.21E+03
P_6	9.40E+05 ± 3.47E+03≈	9.74E+05 ± 2.30E+04−	9.76E+05 ± 1.26E+05−	9.50E+05 ± 4.37E+04−	9.39E+05 ± 1.97E+03
P_7	1.01E+06 ± 1.74E+05≈	1.37E+06 ± 1.40+05−	1.37E+06 ± 2.02E+05−	1.11E+06 ± 6.68E+04−	9.52E+05 ± 2.37E+04
P_8	1.74E+01 ± 2.12E+00−	1.88E+01 ± 1.48E+00−	1.56E+01 ± 1.85E+00−	1.39E+01 ± 2.21E+00≈	1.43E+01 ± 1.75E+00
−	4	8	6	5	
+	0	0	0	0	
≈	4	0	2	3	

"−", "+", and "≈" denote that the performance of the corresponding algorithm is worse than, better than, and similar to that of CoBiDE, respectively.

Table 10
Results of the multiple-problem Wilcoxon's test for JADE, jDE, SaDE, CoDE, and CoBiDE at a 0.05 significance level and at a 0.1 significance level.

Algorithm	$R+$	$R-$	p-Value	$\alpha = 0.05$	$\alpha = 0.1$
CoBiDE vs JADE	36.0	0.0	0.007812	Yes	Yes
CoBiDE vs jDE	36.0	0.0	0.007812	Yes	Yes
CoBiDE vs SaDE	28.0	0.0	0.015626	Yes	Yes
CoBiDE vs CoDE	30.0	6.0	0.10938	No	No

Table 11
Ranking of JADE, jDE, SaDE, CoDE, and CoBiDE according to the statistical test of the Friedman test.

Algorithm	Ranking
CoBiDE	1.3125
CoDE	2.625
JADE	3.3125
SaDE	3.625
jDE	4.125

experimental results. From the experimental results shown in Table 9, we can see that JADE, jDE, SaDE, and CoDE cannot outperform CoBiDE on any problems, and that CoBiDE surpasses JADE, jDE, SaDE, and CoDE on four, eight, six, and five problems, respectively, which indicates that overall, CoBiDE performs significantly better than the four competitors on eight complex real-world engineering optimization problems.

By making use of the KEEL software [33], the multiple-problem Wilcoxon's test and the Friedman's test have been implemented. The experimental results have been summarized in Tables 10 and 11. As shown in Table 10, CoBiDE shows higher R+ values than R− values in all the cases. Moreover, the p values less than 0.05 and 0.1 in three cases (i.e., CoBiDE vs JADE, CoBiDE vs jDE, and CoBiDE vs SaDE). In addition, CoBiDE has the best ranking according to Table 11.

Therefore, the above experimental results verify the potential of CoBiDE in the real-world applications.

6. Conclusion

During the past fifteen years, differential evolution (DE) which is an efficient and robust evolutionary algorithm has become a hotspot in the community of evolutionary computation. In order to improve the performance of DE, CoBiDE, a DE variant based on covariance matrix learning and bimodal distribution parameter setting, is presented in this paper.

In CoBiDE, Eigen decomposition is applied to the covariance matrix computed according to the current population, the purpose of which is to establish an Eigen coordinate system for the crossover operator. The covariance matrix learning relaxes the dependence of DE on the coordinate system to a certain degree and improves the performance on problems with high variable correlation. Moreover, the bimodal distribution parameter setting is introduced for the scaling factor F and the crossover control parameter CR. The bimodal distribution for both F and CR is composed of two Cauchy distributions. CoBiDE has been tested on 25 benchmark test functions developed for IEEE CEC2005 and eight complex real-world engineering optimization problems collected for IEEE CEC2011. The experimental results suggest that the performance of CoBiDE is better than that of four other DE variants and three other state-of-the-art EAs. The experimental results also verify that both the covariance matrix learning and the bimodal distribution parameter setting are critical for CoBiDE. Finally, the parameter sensitivity of CoBiDE has been studied experimentally.

The Matlab source code of CoBiDE can be downloaded from Y. Wang's homepage: http://ist.csu.edu.cn/YongWang.htm

Acknowledgments

The authors sincerely thank the anonymous reviewers for their constructive and helpful comments and suggestions.

This research was supported in part by the National Natural Science Foundation of China under Grant 61273314, 51175519 and 61175064, in part by the Hong Kong Scholars Program, in part by the China Postdoctoral Science Foundation under Grant 2013M530359, in part by RGC of Hong Kong (CityU: 116212), and in part by the Program for New Century Excellent Talents in University under Grant NCET-13-0596. This research was made possible by NPRP grant # 4-1162-1-181 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the author[s].

References

[1] R. Storn, K. Price, Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces, Berkeley, CA, Tech. Rep. TR-95-012, 1995.
[2] R. Storn, K.V. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (4) (1997) 341–359.
[3] S. Das, A. Abraham, A. Konar, Automatic clustering using an improved differential evolution algorithm, IEEE Transactions on Systems, Man, and Cybernetics, Part A 38 (1) (2008) 218–236.
[4] F. Neri, E. Mininno, Memetic compact differential evolution for Cartesian robot control, IEEE Computational Intelligence Magazine 5 (2) (2010) 54–65.
[5] L. Wang, L.P. Li, Fixed-structure H∞ controller synthesis based on differential evolution with level comparison, IEEE Transactions on Evolutionary Computation 15 (1) (2011) 341–359.
[6] G.W. Greenwood, Using differential evolution for a subclass of graph theory problems, IEEE Transactions on Evolutionary Computation 13 (5) (2009) 1190–1192.
[7] N. Noman, H. Iba, Accelerating differential evolution using an adaptive local search, IEEE Transactions on Evolutionary Computation 12 (1) (2008) 107–125.

[8] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Transactions on Evolutionary Computation 13 (2) (2009) 398–417.
[9] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.-P. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, Nanyang Technol. Univ., Singapore, Tech. Rep. KanGAL #2005005, May, IIT Kanpur, India, 2005.
[10] S. Das, P.N. Suganthan, Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems, Technical Report, Jadavpur University and Nanyang Technological University, 2010.
[11] Y. Wang, Z. Cai, Q. Zhang, Enhancing the search ability of differential evolution through orthogonal crossover, Information Sciences 185 (1) (2012) 153–177.
[12] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Transactions on Evolutionary Computation 15 (1) (2011) 4–31.
[13] H.Y. Fan, J. Lampinen, A trigonometric mutation operator to differential evolution, Journal of Global Optimization 27 (1) (2003) 105–129.
[14] J. Zhang, A.C. Sanderson, JADE: adaptive differential evolution with optional external archive, IEEE Transactions on Evolutionary Computation 13 (5) (2009) 945–958.
[15] S. Das, A. Abraham, U.K. Chakraborty, A. Konar, Differential evolution using a neighborhood-based mutation operator, IEEE Transactions on Evolutionary Computation 13 (3) (2009) 526–553.
[16] Y.W. Leung, Y. Wang, An orthogonal genetic algorithm with quantization for global numerical optimization, IEEE Transactions on Evolutionary Computation 5 (1) (2001) 41–53.
[17] J. Liu, J. Lampinen, A fuzzy adaptive differential evolution algorithm, Soft Computing – A Fusion of Foundations, Methodologies and Applications 9 (6) (2005) 448–462.
[18] J. Brest, S. Greiner, B. Boskovic, M. Mernik, V. Zumer, Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems, IEEE Transactions on Evolutionary Computation 10 (6) (2006) 646–657.
[19] S. Tsutsui, M. Yamamura, T. Higuchi, Multi-parent recombination with simplex crossover in real coded genetic algorithms, in: Proceedings of the Genetic and Evolutionary Conference, 1999, pp. 657–664.
[20] S. Rahnamayan, H.R. Tizhoosh, M.M.A. Salama, Opposition-based differential evolution, IEEE Transactions on Evolutionary Computation 12 (1) (2008) 64–79.
[21] J. Sun, Q. Zhang, E.P.K. Tsang, DE/EDA: a new evolutionary algorithm for global optimization, Information Sciences 169 (3–4) (2005) 249–262.
[22] R. Mallipeddi, P.N. Suganthan, Q.K. Pan, M.F. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, Applied Soft Computing 11 (2) (2011) 1679–1696.
[23] Y. Wang, Z. Cai, Q. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, IEEE Transactions on Evolutionary Computation 15 (1) (2011) 55–66.
[24] W. Gong, Z. Cai, C.X. Ling, H. Li, Enhanced differential evolution with adaptive strategies for numerical optimization, IEEE Transactions on Systems, Man, and Cybernetics, Part B 41 (2) (2011) 397–413.
[25] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evolutionary Computation 9 (2) (2001) 159–195.
[26] M. Takahashi, H. Kita, A crossover operator using independent component analysis for real-coded genetic algorithms, in: Proceedings of IEEE Congress on Evolutionary Computation, 2001, pp. 643–649.
[27] K. Walczak, Hybrid differential evolution with covariance matrix adaptation for digital filter design, in: 2011 IEEE Symposium on Differential Evolution (SDE), 2011, pp. 1–7.
[28] J.H. Kämpf, D. Robinson, A hybrid CMA-ES and HDE optimisation algorithm with application to solar energy potential, Applied Soft Computing 9 (2) (2009) 738–745.
[29] A. LaTorre, S. Muelas, J. Peña, Evaluating the multiple offspring sampling framework on complex continuous optimization functions, Memetic Computing (2013), http://dx.doi.org/10.1007/s12293-013-0120-8 (in press).
[30] A. Auger, N. Hansen, A restart CMA evolution strategy with increasing population size, in: Proceedings of the IEEE Congress on Evolutionary Computation, 2005, pp. 1769–1776.
[31] J. Cohen, Statistical Power Analysis for the Behavioral Sciences, second ed., Lawrence Earlbaum Associates, Hillsdale, NJ, 1988.
[32] S. García, D. Molina, M. Lozano, F. Herrera, A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization, Journal of Heuristics 15 (6) (2009) 617–644.
[33] J. Alcalá-Fdez, L. Sánchez, S. García, M.J. del Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, J.C. Fernández, F. Herrera, KEEL: a software tool to assess evolutionary algorithms to data mining problems, Soft Computing 13 (3) (2009) 307–318.
[34] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Transactions on Evolutionary Computation 10 (3) (2006) 281–295.
[35] C. Garcia-Martinez, M. Lozano, F. Herrera, D. Molina, A.M. Sanchez, Global and local real-coded genetic algorithms based on parent-centric crossover operators, European Journal of Operational Research 185 (3) (2008) 1088–1113.