Soft Comput (2011) 15:2141–2155
DOI 10.1007/s00500-010-0643-6

FOCUS

Scalability of generalized adaptive differential evolution for large-scale continuous optimization

Zhenyu Yang · Ke Tang · Xin Yao

Published online: 11 September 2010

Abstract Differential evolution (DE) has become a very powerful tool for global continuous optimization problems. Parameter adaptations are the most commonly used techniques to improve its performance. The adoption of these techniques has assisted the success of many adaptive DE variants. However, most studies on these adaptive DEs are limited to some small-scale problems, e.g. with less than 100 decision variables, which may be quite small comparing to the requirements of real-world applications. The scalability performance of adaptive DE is still unclear. In this paper, based on the analyses of similarities and drawbacks of existing parameter adaptation schemes in DE, we propose a generalized parameter adaptation scheme. Applying the scheme to DE results in a new generalized adaptive DE (GaDE) algorithm. The scalability performance of GaDE is evaluated on 19 benchmark functions with problem scale from 50 to 1,000 decision variables. Based on the comparison with three other algorithms, GaDE is very competitive in both the performance and scalability aspects.

Keywords Differential evolution · Parameter adaptation · Large-scale optimization · Scalability

Z. Yang · K. Tang (✉) · X. Yao
Nature Inspired Computation and Applications Laboratory (NICAL), School of Computer Science and Technology, University of Science and Technology of China, Hefei, China
e-mail: ketang@ustc.edu.cn

Z. Yang
e-mail: zhyuyang@mail.ustc.edu.cn

X. Yao
e-mail: x.yao@cs.bham.ac.uk

X. Yao
School of Computer Science, University of Birmingham, Birmingham, UK

1 Introduction

The interest in large-scale evolutionary optimization has increased dramatically since the 2008 paper (Yang et al. 2008). Conference special sessions and new benchmark functions have been developed since then (Tang et al. 2009). Some algorithms that showed an excellent performance on low-dimensional problems do not perform well any more when the number of dimensions goes up. It is interesting to investigate the reason and how to develop new algorithms for large-scale optimization problems.

DE (Storn and Price 1997) has become a popular and effective algorithm for global continuous optimization problems. The crucial idea of DE is a novel mutation paradigm, which is executed by adding a weighted difference vector between two selected individuals to the third individual. It was found that this kind of mutation has desired self-adaptation feature based on current population diversity (Price et al. 2005). Thanks to these kinds of good features, DE has shown impressive performance on various optimization problems (Storn 1999; Vesterstrom and Thomsen 1980; Hansen 2005). However, most reported studies on DE are obtained using small-scale problems, e.g., with less than 100 decision variables, which are relatively small for many real-world applications. While it was already found that many optimization methods (e.g. evolutionary algorithms, EAs) suffer against large-scale problems (Yang et al. 2008; Liu et al. 2001), the influence on DE are still somewhat unknown. Scalability analysis on DE is useful to provide some insight into its computational complexity and reliability as the problem scale increases.

DE has several different mutation schemes, and three control parameters, i.e., population size NP, mutation scale factor F and crossover rate CR. Mutation schemes and control parameters adaptations are the most frequently used methods to improve DE's performance on small-scale problems. For analyzing the scalability of DE against large-scale problems, this could also be the first step. Apart from the common parameter NP for all population-based algorithms, mutation schemes and the other two parameters adaptations are the most important issues in proving DE's performance. Much related work has been done along these lines. For example, the relation between the control parameters and population diversity has been analyzed in Zaharie (2002) and Yang et al. (2008). Brest et al. (2006) presented the jDE, in which F and CR values are attached to all individuals of population, and updated in each generation according to some heuristic rules. Self-adaptive differential evolution (SaDE) (Qin and Suganthan 2005; Qin et al. 2009) and self-adaptive differential evolution with neighborhood search (SaNSDE) (Yang et al. 2008) adopt multiple mutation schemes spontaneously and introduce some probability parameters to control which one to use in practice. Parameters F and CR in the two algorithms are also adaptively controlled in different ways. JADE in Zhang and Sanderson (2009) proposed a novel mutation scheme that utilizes both the best solution and inferior solution information. Probability distribution-based adaptations are adopted to control F and CR. DEGL in Das et al. (2009) proposed a hybrid mutation scheme that utilizes both an explorative and an exploitive mutation operator. The two mutation operators were linearly combined through a new parameter, called the weight factor. Four different schemes were investigated to adaptively control the weight factor. Based on the success of these work, the behaviors of adaptive DE for large-scale optimization problems deserve more investigation.

In this paper, we first analyze the similarities and drawbacks of the adaptation methods used in several recently published adaptive DE variants. Then, we propose a generalized parameter adaptation scheme which is able to control any individual based parameters in EAs. According to the scheme, in each generation, a parameter value is generated for each individual according to a probability distribution. After the evolutionary operations and selection process, good parameter values, which are able to make the corresponding offspring enter the next generation, are recorded to update the previous probability distribution. Such a procedure is executed during the whole evolution process, and thus the parameter is adjusted adaptively. Based on the general parameter adaptation scheme, we propose the generalized adaptive differential evolution (GaDE), in which the parameters F and CR of DE are adapted with the above-mentioned method. Moreover, inspired by SaDE (Qin et al. 2009), a new adaptation strategy for mutation schemes is also incorporated in GaDE. The efficacy and scalability of the proposed GaDE are studied on a benchmark suite including 19 problems with scale from 50 to 1,000 decision variables.

The rest of this paper is organized as follows. Section 2 describes the basic operations of classical DE. Section 3 reviews several recently proposed adaptive DE variants. Section 4 proposes a new parameter adaptation scheme and also a new adaptive DE variant, GaDE. Section 5 presents the experimental studies. Finally, Section 6 concludes this paper briefly.

2 Classical DE

Individuals in DE for continuous optimization are represented by D-dimensional vectors $\mathbf{x}_i, \forall i \in \{1, \ldots, \mathrm{NP}\}$, where D is the number of decision variables and NP is the population size. According to Price et al. (2005), the basic operations in classical DE can be summarized as follows:

1. Mutation:

$$\mathbf{v}_i = \mathbf{x}_{r_1} + F(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}),$$

where $r_1, r_2, r_3 \in [1, \mathrm{NP}]$ are random and mutually different integers, and they are also different with the vector index i. Scale factor $F > 0$ is a real constant factor and it is often set to 0.5.2. Crossover:

$$\mathbf{u}_i(j) = \begin{cases} \mathbf{v}_i(j), & \text{if } U_j(0,1) \le \mathrm{CR} \text{ or } j = j_{\mathrm{rand}} \\ \mathbf{x}_i(j), & \text{otherwise} \end{cases}$$

with $U_j(0,1)$ stands for a uniform random number between 0 and 1, and j_{rand} is a randomly chosen index to ensure that the trial vector $\mathbf{u}_i$ does not duplicate $\mathbf{x}_i$. $\mathrm{CR} \in [0,1]$ is the crossover rate, which is often set to 0.9.3. Selection:

$$\mathbf{x}'_i = \begin{cases} \mathbf{u}_i, & \text{if } f(\mathbf{u}_i) \le f(\mathbf{x}_i) \\ \mathbf{x}_i, & \text{otherwise} \end{cases}$$

where $\mathbf{x}'_i$ is the offspring of $\mathbf{x}_i$ for the next generation.[1]

There are several schemes of DE based on different mutation strategies (Price et al. 2005):

$$\mathbf{v}_i = \mathbf{x}_{r_1} + F(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}) \quad (1)$$

$$\mathbf{v}_i = \mathbf{x}_{\mathrm{best}} + F(\mathbf{x}_{r_1} - \mathbf{x}_{r_2}) \quad (2)$$

$$\mathbf{v}_i = \mathbf{x}_i + F(\mathbf{x}_{\mathrm{best}} - \mathbf{x}_i) + F(\mathbf{x}_{r_1} - \mathbf{x}_{r_2}) \quad (3)$$

$$\mathbf{v}_i = \mathbf{x}_{\mathrm{best}} + F(\mathbf{x}_{r_1} - \mathbf{x}_{r_2}) + F(\mathbf{x}_{r_3} - \mathbf{x}_{r_4}) \quad (4)$$

$$\mathbf{v}_i = \mathbf{x}_{r_1} + F(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}) + F(\mathbf{x}_{r_4} - \mathbf{x}_{r_5}). \quad (5)$$

Schemes (1) and (3), with notations as DE/rand/1 and DE/current to best/2, are the most often used in practice

[1] Without loss of generality, we consider only minimization problem in this paper.

due to their superior performance (Price et al. 2005; Qin et al. 2009).

3 Review of adaptive DE

In this section, we introduce some recent adaptive DE variants, jDE, SaDE, SaNSDE, JADE, and DEGL. They all used parameter or mutation scheme adaptation techniques to improve the performance of classical DE. Key issues of these adaptive DE variants are summarized in the following subsections. Besides these representatives, we are aware of many other DE variants have been published towards various research aspects. A comprehensive survey on the recent advances in DE can be found in Neri and Tirronen (2010).

3.1 jDE

jDE was proposed by Brest et al. (2006) based on the parameter adaptation for the control of scale factor F, and crossover rate CR. It associates each individual with its own control parameter F_i and CR_i. Initially, F_i and CR_i are set to 0.5 and 0.9, respectively. Then, in each generation jDE generates new values for F_i and CR_i as follows:

$$F_{i,g+1} = \begin{cases} 0.1 + 0.9 \cdot \mathrm{rand}_1, & \text{if } \mathrm{rand}_2 < \tau_1 \\ F_{i,g}, & \text{otherwise} \end{cases} \tag{6}$$

and

$$\mathrm{CR}_{i,g+1} = \begin{cases} 0.1 + 0.9 \cdot \mathrm{rand}_3, & \text{if } \mathrm{rand}_4 < \tau_2 \\ \mathrm{CR}_{i,g}, & \text{otherwise,} \end{cases} \tag{7}$$

where $\mathrm{rand}_j, j = 1, 2, 3, 4$, are uniform random values in [0, 1]. Parameters $\tau_1 = \tau_2 = 0.1$ represent the probability to adjust previous control parameter values. The newly generated parameter values are used in the mutation and crossover operations to create corresponding offspring and will replace the previous parameter values if the offspring survive in the selection process.

3.2 SaDE

SaDE proposed by Qin et al. (2005, 2009) gave the first attempt to simultaneously adopt more than one mutation schemes. The adaptation techniques in SaDE are composed of two parts: (a) the adaptation of the probability, mp_i, of applying the i-th mutation scheme; and (b) the adaptation of DE's parameter F and CR.

For the mutation schemes adaptation, the probability of applying the k-th mutation scheme is represented using $mp_k, k = 1, 2, \ldots, K$, where K is the total number of candidate schemes. All mp_ks are initialized as $1/K$, which means all schemes have the same probability to be chosen. At the generation G, after evaluating all the generated trial individuals, the number of trial individuals generated by the k-th scheme that can successfully enter the next generation is recorded as $ns_{k,G}$, while the number of trial individuals generated by the k-th schemes that are discarded in the next generations is recored as $nf_{k,G}$. And then the probability of choosing the k-th scheme is updated by

$$mp_{k,G} = \frac{S_{k,G}}{\sum_{k=1}^{K} S_{k,G}}, \quad k = 1, \ldots, K \tag{8}$$

with

$$S_{k,G} = \frac{\sum_{g=G-\mathrm{LP}}^{G-1} ns_{k,g}}{\sum_{g=G-\mathrm{LP}}^{G-1} ns_{k,g} + \sum_{g=G-\mathrm{LP}}^{G-1} nf_{k,g}} + \epsilon, \tag{9}$$

where LP means *learning period*, which is a predefined number of generations (e.g., 50 in SaDE). The small constant value $\epsilon = 0.01$ is used to avoid the possible null success rates.

For the adaptation of scale factor F, individual-based F_i are generated at the start of each generation according to a Gaussian distribution with mean 0.5, and standard deviation 0.3, i.e.:

$$F_i = N_i(0.5, 0.3) \tag{10}$$

All F_i will be truncated to the interval $(0, 2]$ based on the empirical range of scale factor F.

For the adaptation of crossover rate CR, individual-based CR_i are generated every five generations according to a Gaussian distribution with mean CR_m and standard deviation 0.1, i.e.:

$$\mathrm{CR}_i = N_i(\mathrm{CR}_m, 0.1). \tag{11}$$

The center CR_m is initialized to be 0.5, and then updated every 25 generations according to

$$\mathrm{CR}_m = \frac{1}{|\mathrm{CR}_{\mathrm{suc}}|} \sum_{k=1}^{|\mathrm{CR}_{\mathrm{suc}}|} \mathrm{CR}_{\mathrm{suc}}(k), \tag{12}$$

where $\mathrm{CR}_{\mathrm{suc}}$ are the recorded successful CR values which are able to make the corresponding offspring enter the next generation in the last 25 generations.

3.3 SaNSDE

SaNSDE proposed in Yang et al. (2008) can be regarded as an improved version of SaDE. Its mutation is executed in the same way as SaDE except that the scale factor in the adopted mutation schemes are generated according to either a Gaussian distribution or a Cauchy distribution, i.e.

$$F_i = \begin{cases} N_i(0.5, 0.3) & \text{if } U_i(0, 1) < fp \\ C_i(0, 1), & \text{otherwise} \end{cases} \tag{13}$$

where $N_i(\mu, \delta^2)$ denotes a Gaussian random number with mean μ and standard deviation δ, and $C_i(\mu, \delta)$ denotes a random value from a Cauchy distribution with location and scale parameters μ, δ, respectively. The probability, *fp*, of applying either of the two distributions is adapted in a manner similar to the Eq. 8 in SaDE.

The adaptation of the crossover rate CR in SaNSDE follows the method in SaDE except that a weighting strategy is applied

$$\mathrm{CR}_m = \sum_{k=1}^{|\mathrm{CR}_{\mathrm{suc}}|} w_k \cdot \mathrm{CR}_{\mathrm{suc}}(k) \tag{14}$$

$$w_k = \delta_f(k) \Bigg/ \left(\sum_{k=1}^{|\delta_f|} \delta_f(k) \right), \tag{15}$$

where $\delta_f(k)$ is the corresponding fitness improvement related to each successful crossover rate $\mathrm{CR}_{\mathrm{suc}}(k)$.

3.4 JADE

JADE (Zhang and Sanderson 2009; Yang et al. 2009) is another DE variant, in which a new mutation scheme named DE/current-to-*p*best/2 is adopted. The mutation vectors are generated as follows:

$$\mathbf{v}_i = \mathbf{x}_i + F_i(\mathrm{x}_{p\_\mathrm{best}} - \mathbf{x}_i + \mathbf{x}_{r1} - \tilde{\mathbf{x}}_{r2}), \tag{16}$$

where x_{p_best} is randomly chosen as one of the best $100p\%$ individuals in the current population with $p \in (0, 1]$. $\mathbf{x}_{r1}(r1 \neq i)$ is an individual randomly selected from the population, and $\tilde{\mathbf{x}}_{r2}$ is an individual (distinct from $\mathbf{x}_i$ and $\mathbf{x}_{r1}$) randomly chosen from the union of the current population and an external archive of inferior solutions. The archive is initialized to be empty and then filled with the parent solutions that fail in the selection process of each generation. If the archive size exceeds a certain threshold, say NP, then some solutions are randomly removed from the archive to keep its size at NP.

In JADE, F_i and CR_i are randomly generated at each generation

$$F_i = C_i(F_m, 0.1) \tag{17}$$

$$\mathrm{CR}_i = N_i(\mathrm{CR}_m, 0.1), \tag{18}$$

where CR_m and F_m are updated in an adaptive manner

$$F_m = (1 - c)F_m + c \cdot \mathrm{mean}_L(F_{\mathrm{suc}}) \tag{19}$$

$$\mathrm{CR}_m = (1 - c) \cdot \mathrm{CR}_m + c \cdot \mathrm{mean}_A(\mathrm{CR}_{\mathrm{suc}}) \tag{20}$$

where $\mathrm{CR}_{\mathrm{suc}}$ and F_{suc} are the respective sets of all successful crossover probabilities and successful mutation factors obtained in the selection process at current generation. Parameter c is a positive constant between 0 and 1. $\mathrm{mean}_A(\cdot)$ is the usual arithmetic mean and $\mathrm{mean}_L(\cdot)$ is the Lehmer mean

$$\mathrm{mean}_L(F_{\mathrm{suc}}) = \frac{\sum_{F \in F_{\mathrm{suc}}} F^2}{\sum_{F \in F_{\mathrm{suc}}} F} \tag{21}$$

which pays more weight on larger mutation factor F to improve evolutionary progress.

3.5 DEGL

DEGL, proposed in Das et al. (2009), intends to balance the exploration and exploitation abilities of DE through a novel neighborhood-based mutation scheme. Specifically, the individuals of a population in DEGL are organized on a ring topology with respect to their indices. The neighborhood of the i-th individual $\mathbf{x}_i$ is given by those individuals $\mathbf{x}_{i-k}, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_{i+k}$, where the parameter k is defined as the radius of the neighborhood. The mutation of DEGL was performed through a combination of two contributions, the first contribution is given by the neighborhood individuals and the second by the entire population.

– The first one is called local contribution, which is calculated as:

$$\mathbf{L}_i = \mathbf{x}_i + \alpha(\mathbf{x}_{k\_\mathrm{best}_i} - \mathbf{x}_i) + \beta(\mathbf{x}_p + \mathbf{x}_q) \tag{22}$$

where k_best_i indicates the best individual in the neighborhood of $\mathbf{x}_i$, and p, q are chosen randomly in the interval $[i - k, i + k]$, with the condition $p \neq q \neq i$.

– The second one is called global contribution, which is given by:

$$\mathbf{G}_i = \mathbf{x}_i + \alpha(\mathbf{x}_{\mathrm{best}} - \mathbf{x}_i) + \beta(\mathbf{x}_r - \mathbf{x}_s), \tag{23}$$

where $\mathbf{x}_{\mathrm{best}}$ is the best individual in the current population, $\mathbf{x}_r$ and $\mathbf{x}_s$ are two individuals randomly selected with the condition $r \neq s \neq i$.

The parameters α and β in Eqs. 22 and 23 are two constants which have a similar role to that of the scale factor F in DE. The local and global contributions are then linearly combined by means of:

$$\mathbf{v}'_i = w\mathbf{G}_i + (1 - w)\mathbf{L}_i, \tag{24}$$

where $\mathbf{v}'_i$ is the mutated offspring of $\mathbf{x}_i$, and w is a weight factor to be set between 0 and 1.

Regarding the parameter setting in DEGL, as suggested in Das et al. (2009), the scale factors were set to $\alpha = \beta = 0.8$, the crossover rate was set to $\mathrm{CR} = 0.9$, and the neighborhood radius k was set to 10% of the population size. In addition, four different schemes were proposed and investigated to adaptively control the weight factor w. Let g denote the generation counter, $g_{\max}$ denote the maximum number of generations, and w_g denote the weight factor at the generation g, the four schemes are listed as follows:

1. Linear increment: w is linearly increased from 0 to 1:

$$w_g = \frac{g}{g_{\max}} \tag{25}$$

2. Exponential increment: w increases from 0 to 1 in an exponential fashion:

$$w_g = \exp\left(\frac{g}{g_{\max}}\ln(2)\right) - 1 \tag{26}$$

3. Random weight factor: The weight factor of each individual is generated from a uniform random number in (0, 1), i.e.:

$$w_{i,g} = U_i(0,1). \tag{27}$$

The subscript i mean that for each individual the scheme will generate a new w value.

4. Self-adaptive weight factor: In this scheme, a w_i was attached to the encoding of each candidate solution

$$\mathbf{x}_i = (x_{i,1},\ldots,x_{i,D},w_i). \tag{28}$$

During the initialization stage, each weight factor is initialized between 0 and 1. At each generation, before the combined mutation operation given in Eq. 24, the weight factor w_i is updated according to the following equation:

$$w_i = w_i + F(w_{\text{best}} - w_i) + F(w_r - w_s), \tag{29}$$

where w_{best} is the weight factor associated to the best individual of the current popualtion, and w_r and w_s are the weight factors of the randomly selected individuals $\mathbf{x}_r$ and $\mathbf{x}_s$ in Eq. 23.

In Das et al. (2009), the fourth scheme, i.e. the self-adaptive weight factor, was proved to be the most efficient one.

4 Generalized adaptive DE

4.1 Generalized parameter adaptation scheme

The parameter adaptation techniques used in adaptive DE variants listed in Sect. 3 can be classified into two classes: (a) heuristic rules based and (b) probability distribution based. jDE and DEGL are the representatives of heuristic rules based methods. Although this kind of methods have shown promising performance on some problems, the reason why they performed well is not fully understood, and thus it is difficult to analyze their advantages and weakness on different problem classes. Moreover, the used heuristic rules often introduce some new parameters (e.g. τ_1 and τ_2 in jDE), which may also be difficult to set in some situations. SaDE, SaNSDE and JADE belong to the other parameter adaptation class, i.e. probability distribution based. In this class, different parameter values are randomly generated according to a certain probability distribution, after evolutionary operations and selection process, the successful parameter values, which managed to generate better new solutions, are recorded to update the distribution for later evolution. The problem of SaDE and SaNSDE is that the parameters are only adapted after some *learning period*, e.g. 50 generations. If the time budget is limited, they may not have enough time to learn the suitable parameter settings.

And they always replaced previous learned parameter settings with the values calculated based on the learning experience of the current *learning period*. The risk is quite high if the current *learning period* is not reliable.

As for JADE, the update of probability distribution centers F_m and CR_m may not be reliable in some cases. For example, according to Eqs. 19 and 20 no matter how many parameter values are recorded in F_{suc} or CR_{suc}, F_m or CR_m will be changed in the same way. The probability distribution may be changed acutely by some occasional good parameter values.

Our proposed adaptive scheme falls into the second class (i.e., probability distribution based), with which we intend to develop a generalized version of the above-mentioned methods. For a given EA, we assume it has an individual-based and very sensitive control parameter $A \in [A_{\min}, A_{\max}]$, which need to be tuned during evolution. The generalized scheme is designed as follows.

1. During initialization, set generation counter $g = 0$, previous learning experience $M = 0$, and initial parameter center $A_m = A_0$, where A_0 is an empirical value in the range $[A_{\min}, A_{\max}]$.
2. Set $g = g + 1$, and generate A_i for the i-th individual $\mathbf{x}_i$ based on a probability distribution P with Am as the center:

$$A_i = P(A_m), i = 1,\ldots,\text{NP} \tag{30}$$

3. After evolutionary operations, the A_i that is able to make the offspring $\mathbf{x}'_i$ of $\mathbf{x}_i$ to successfully enter the next generation will be marked as a good parameter value. These good parameter values and the corresponding fitness improvements will be recorded in $A_{\text{suc}}(k)$ and $\delta_f(k)$, with $k = 1,\ldots,M_g$.
4. Update the parameter center according to

$$A_m = (1-w)A_m + wAm' \tag{31}$$

where the weight w is determined by

$$w = \frac{M_g}{M_g + M} \tag{32}$$

and A'_m is the weighted mean of values in A_{suc}:

$$A'_m = \sum_{k=1}^{M_g}\left(A_{\text{suc}}(k)\frac{\delta_f(k)}{\Delta_f}\right), \quad \Delta_f = \sum_{k=1}^{M_g}\delta_f(k). \tag{33}$$

5. Update the learning experience:

$$M = (1 - c)M + M_g \tag{34}$$

where the decreasing factor c is a user defined parameter, and $c = 0.1$ can be used as the default setting.
6. Go to Step 2 for the next generation until a stopping criterion is satisfied.

The proposed adaptation scheme does not involve the parameter *learning period*. Instead, a parameter c is introduced to make the algorithm balance between the experiences learned from previous generations and that of the current generation. With this difference, the proposed scheme is capable of taking into account the reliability of the recorded good parameter values. In the following section, we will apply it in DE to design an new adaptive DE variant, GaDE.

4.2 The proposed algorithm

As mentioned in Sect. 1, mutation scheme selection, scale factor F and crossover rate CR adaptations are the three crucial issues in designing adaptive DE. Based the parameter adaptation scheme presented in Sect. 4.1, we propose a new adaptive DE, GaDE, for large-scale optimization. The key points of GaDE are summarized as follows:

1. Mutation schemes adaptation: Mutation vectors are generated similarly as in SaDE and SaNSDE. The following two mutation schemes are used:

$$\mathbf{v}_i = \mathbf{x}_{r_1} + F_i(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}) \tag{35}$$

$$\mathbf{v}_i = \mathbf{x}_i + F_i(\mathbf{x}_{p\_\text{best}} - \mathbf{x}_i + \mathbf{x}_{r_1} - \mathbf{x}_{r_2}) \tag{36}$$

Probability parameter mp as in Eq. 8 is used to control which scheme to use in practice.
2. Scale factor F adaptation: The parameter adaptation scheme proposed in Sect. 4.1 is used to control the parameter F. The Cauchy distribution with location F_m, and scale parameter $t = 0.2$ is used to generate F values (Yao et al. 1999; Lee and Yao 2004).

   (a) For initialization, set $F_m = 0.5$ (Price et al. 2005).
   (b) At the beginning of each generation, an F_i is generated for each individual based on the Cauchy distribution:

$$F_i = C_i(F_m, 0.2), i = 1, \ldots, \text{NP} \tag{37}$$

   The F_i values will be regenerated if it is outside the interval $(0, 2]$ (Storn and Price 1997).
   (c) At the end of each generation, F_m is updated in the same way as A_m in Eqs. 31–33.
3. Crossover rate CR adaptation: The parameter adaptation scheme proposed in Sect. 4.1 is used to adjust CR. The Gaussian distribution with mean CR_m, and standard deviation 0.1 is used to generate CR values.

   (a) For initialization, set $\text{CR}_m = 0.9$ (Price et al. 2005).
   (b) At the beginning of each generation, a CR_i is generated for each individual based on the Gaussian distribution:

$$\text{CR}_i = N_i(\text{CR}_m, 0.1), i = 1, \ldots, \text{NP}. \tag{38}$$

   The CR_i values will be regenerated if it is outside the interval $[0, 1]$ (Storn and Price 1997).
   (c) At the end of each generation, CR_m is updated in the same way as A_m in Eqs. 31–33.

Note that different probability distributions are used to control the parameters F and CR, respectively. This is based on their different functions and characteristics in DE. For the scale factor F, because it affects the search step size of DE, sampling F values from the Cauchy distribution could increase the probability to generate large search step sizes, which is especially desirable when optimizing multimodal functions. As for the crossover rate CR, it was found that the proper choice of CR can lead to good optimization performance, while a wrong choice may deteriorate the performance significantly (Price et al. 2005). Good values of CR generally fall into a small range for a given problem, with which DE can perform consistently well (Qin et al. 2009). However, the appropriate small range may be different for different optimization problems. The Gaussian distribution with 0.1 as standard deviation is narrowly bounded, and thus is helpful to find out such a small range gradually, while the value range of Cauchy distribution is open-ended (Yao et al. 1999).

5 Experimental studies

5.1 Experimental setup

To analyze the performance and scalability of the proposed GaDE, experiments were conducted on a testing suite provided by the special issue on "Scalability of Evolutionary Algorithms and other Metaheuristics for Large Scale Continuous Optimization Problems" of *Soft Computing*. The testing suite includes 19 scalable functions with different features, such as unimodal or multimodal, separable or nonseparable, and whether they can be easily optimized dimension by dimension. Detailed formulas and descriptions of these functions can be found in http://sci2s.ugr.es/eamhco/updated-functions1-19.pdf. GaDE was tested on all the functions with dimensions $D = 50$, 100, 200, 500 and 1,000. The maximum number of fitness evaluations (FEs) was set to $5{,}000 \times D$ (http://sci2s.ugr.es/eamhco/updated-functions1-19.pdf). For parameter settings, the population size was set to $\mathrm{NP} = 60$, the probability of selecting $\mathbf{x}_{p\_\mathrm{best}}$ in Eq. 36 was set to $p = 0.2$, the decreasing factor c in Eq. 34 was set to 0.1, and all other parameters were adapted based on the description in Sect. 4.2.

5.2 Brief description of algorithms for comparison

The performance of GaDE was compared with three other algorithms, i.e. classical DE, G-CMA-ES (Auger and Hansen 2005) and real-coded CHC (Eshelman and Schaffer 1993). The results of them were provided by the organizers of the special issue in *Soft Computing*. A brief description of these algorithms are provided as follows (http://sci2s.ugr.es/eamhco/descriptions.pdf).

5.2.1 Classical DE

The basic components of classical DE have been given in Sect. 2. The used classical DE for experimental comparison is exactly the same except that the exponential crossover (Price et al. 2005) rather than the binomial crossover was used. The parameters F and CR were fixed to 0.5 and 0.9, respectively, and the population size NP was set to 60 (http://sci2s.ugr.es/eamhco/descriptions.pdf).

5.2.2 Real-coded CHC

The CHC algorithm (Crossgenerational elitist selection, Heterogeneous recombination, and Cataclysmic mutation) (Eshelman 1991) mainly concerns the combination of a selection strategy with a very high selective pressure and several other components. It was found that the CHC algorithm achieved better results on the comparison with different genetic algorithm approaches (Whitley et al. 1996).

Real-coded CHC was proposed in Eshelman and Schaffer (1993) by introducing real-coded representation. It maintains the basis of the original CHC as much as possible. The elitist selection is still used without any modification. In order to apply the heterogeneous recombination, the real values of the representation of the two selected individuals are encoded into bit strings using binary reflected Gray coding and the Hamming distance between the parents is measured. The mutation is only performed on those string pairs that differ from each other by some number of bits, which is denoted as mating threshold. The initial threshold is et to $L/4$, where L is the length of the bit string for representing an individual. The threshold will be recoded by 1 if no offspring is inserted into the new population. Finally, the real-parameter crossover operator BLX-α (Eshelman and Schaffer 1993) is considered to replace the original crossover operator. Restart technique may be applied if the population converges or the algorithm stops making progress during the search process.

For parameter settings, the α of the BLX-α crossover was set to 0.5, and the population size was set to 50 (http://sci2s.ugr.es/eamhco/descriptions.pdf).

5.2.3 G-CMA-ES

The CMA-ES (Covariance Matrix Adaptation Evolution Strategy) (Hansen and Ostermeier 2001) is one of the state-of-the-art EAs for continuous optimization. It has shown impressive performance on many small-scale benchmark functions (Hansen 2005). The G-CMA-ES is a improved variant of CMA-ES by introducing restarts with increasing population size technique (Auger and Hansen 2005). It detects premature convergence and launches a restart strategy which doubles the population size on each restart. It was claimed that by increasing the population size G-CMA-ES becomes more global after each restart, and could improve the performance of CMA-ES on mutlimodal functions.

The parameters of G-CMA-ES were set according to the suggestion in (Auger and Hansen 2005). The initial population is uniform randomly generated from the search space.

5.3 Experimental results and comparison

Table 1 showed the results of GaDE on functions F_1–F_{19} with dimensions $D = 50$, 100, 200, 500 and 1,000. All the results were summarized from 25 independent runs for each problem instance, which means one function with one given dimensionality. Four indicators, i.e. *Best, Median, Worst* and *Mean*, are used to evaluate the performance of GaDE on a certain problem instance. To comment the performance of GaDE against other algorithms, a comparison among GaDE, DE, G-CMA-ES and CHC on mean errors is provided in Table 2.

Generally speaking, GaDE achieved consistently good results on 12 (i.e. F_1, F_4–F_7, F_9–F_{12}, F_{14}–F_{16}, F_{18} and F_{19}) out of 19 functions over all tested dimensions. Although the results on F_2, F_3, F_8, F_{13} and F_{17} are still far away from their optima, GaDE is still better than classical DE. For other two functions F_{10} and F_{19}, it was found that GaDE showed very good performance on low-dimensional problems, but it may be trapped in some local optima with very small probability (about 1 in 25 runs) when these functions are scaled up to 500 and 1,000 dimensions. It is interesting to find that the results of classical DE on these two functions are consistently good. This might be because in GaDE a more greedy mutation scheme DE/current-to-p_best/2 was used.

Table 1 Experimental results of GaDE on F_1–F_{19} with dimensions $D = 50$, 100, 200, 500 and 1,000

D	F_i	Best	Median	Worst	Mean
50	F_1	0.00E+00	0.00E+00	0.00E+00	0.00E+00
100		0.00E+00	0.00E+00	0.00E+00	0.00E+00
200		0.00E+00	0.00E+00	0.00E+00	0.00E+00
500		0.00E+00	0.00E+00	0.00E+00	0.00E+00
1,000		0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	F_2	3.50E+00	1.27E+01	2.81E+01	1.46E+01
100		1.52E+01	3.92E+01	5.75E+01	3.88E+01
200		4.52E+01	5.91E+01	6.61E+01	5.76E+01
500		3.30E+01	7.49E+01	8.79E+01	7.42E+01
1,000		8.16E+01	8.92E+01	1.01E+02	8.93E+01
50	F_3	3.31E+00	1.28E+01	1.62E+01	1.18E+01
100		4.75E+01	5.35E+01	1.06E+02	5.89E+01
200		1.20E+02	1.44E+02	2.14E+02	1.61E+02
500		3.92E+02	4.35E+02	4.96E+02	4.40E+02
1,000		8.55E+02	9.45E+02	1.00E+03	9.45E+02
50	F_4	0.00E+00	0.00E+00	0.00E+00	0.00E+00
100		0.00E+00	0.00E+00	0.00E+00	0.00E+00
200		0.00E+00	0.00E+00	0.00E+00	0.00E+00
500		0.00E+00	0.00E+00	0.00E+00	0.00E+00
1,000		0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	F_5	0.00E+00	0.00E+00	0.00E+00	0.00E+00
100		0.00E+00	0.00E+00	0.00E+00	0.00E+00
200		0.00E+00	0.00E+00	0.00E+00	0.00E+00
500		0.00E+00	0.00E+00	0.00E+00	0.00E+00
1,000		0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	F_6	0.00E+00	0.00E+00	0.00E+00	0.00E+00
100		0.00E+00	0.00E+00	0.00E+00	0.00E+00
200		0.00E+00	0.00E+00	1.42E−14	0.00E+00
500		0.00E+00	1.42E−14	3.91E−14	1.46E−14
1,000		1.07E−14	1.42E−14	5.68E−14	1.66E−14
50	F_7	0.00E+00	0.00E+00	0.00E+00	0.00E+00
100		0.00E+00	0.00E+00	0.00E+00	0.00E+00
200		0.00E+00	0.00E+00	0.00E+00	0.00E+00
500		0.00E+00	0.00E+00	0.00E+00	0.00E+00
1,000		0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	F_8	4.75E−10	4.13E−09	6.09E−08	1.08E−08
100		4.67E−04	1.11E−03	3.75E−03	1.23E−03
200		1.01E+00	2.96E+00	8.33E+00	3.02E+00
500		8.22E+02	1.28E+03	2.03E+03	1.33E+03
1,000		1.35E+04	1.73E+04	2.26E+04	1.77E+04
50	F_9	0.00E+00	0.00E+00	7.53E−06	6.24E−07
100		0.00E+00	0.00E+00	3.55E−06	3.87E−07
200		0.00E+00	0.00E+00	6.05E−08	4.53E−09
500		0.00E+00	0.00E+00	0.00E+00	0.00E+00
1,000		0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	F_{10}	0.00E+00	0.00E+00	0.00E+00	0.00E+00
100		0.00E+00	0.00E+00	0.00E+00	0.00E+00
200		0.00E+00	0.00E+00	1.05E+00	4.20E−02
500		0.00E+00	0.00E+00	2.10E+00	3.78E−01
1,000		0.00E+00	0.00E+00	2.10E+00	4.62E−01
50	F_{11}	0.00E+00	2.90E−07	1.10E−05	1.31E−06
100		0.00E+00	0.00E+00	6.67E−06	4.34E−07
200		0.00E+00	0.00E+00	3.48E−06	1.85E−07
500		0.00E+00	0.00E+00	0.00E+00	0.00E+00
1,000		0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	F_{12}	0.00E+00	0.00E+00	0.00E+00	0.00E+00
100		0.00E+00	0.00E+00	0.00E+00	0.00E+00
200		1.14E−14	3.25E−14	1.21E−13	4.92E−14
500		2.30E−13	8.58E−13	4.26E-12	1.07E−12
1,000		1.89E−12	3.58E−12	7.78E−12	3.85E−12
50	F_{13}	8.00E+00	1.19E+01	1.45E+01	1.19E+01
100		4.14E+01	4.61E+01	9.54E+01	4.99E+01
200		1.10E+02	1.14E+02	1.71E+02	1.24E+02
500		3.19E+02	3.32E+02	3.50E+02	3.34E+02
1,000		6.84E+02	7.09E+02	7.60E+02	7.15E+02
50	F_{14}	1.75E−14	2.92E−13	8.85E−12	9.78E−13
100		1.13E−13	3.61E−13	4.88E−12	7.90E−13
200		5.87E−13	1.80E−12	2.22E−11	2.87E−12
500		1.04E−11	2.43E−11	6.62E−11	2.79E−11
1,000		6.84E−11	8.24E−11	1.20E−10	8.82E−11
50	F_{15}	0.00E+00	0.00E+00	0.00E+00	0.00E+00
100		0.00E+00	0.00E+00	0.00E+00	0.00E+00
200		0.00E+00	0.00E+00	0.00E+00	0.00E+00
500		0.00E+00	0.00E+00	0.00E+00	0.00E+00
1,000		0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	F_{16}	1.16E−12	3.08E−12	1.30E−11	4.78E−12
100		7.55E−13	1.88E−12	6.74E−12	2.45E−12
200		6.20E−13	1.49E−12	3.66E−12	1.58E−12
500		1.10E−12	1.64E−12	2.70E−12	1.67E−12
1,000		1.53E−12	2.32E−12	3.07E−12	2.35E−12
50	F_{17}	8.65E−03	1.39E−01	2.94E+00	4.97E−01
100		4.19E−01	2.41E+00	9.96E+00	3.28E+00
200		1.60E+01	1.97E+01	8.16E+01	2.45E+01
500		8.62E+01	8.86E+01	1.39E+02	9.26E+01
1,000		2.12E+02	2.18E+02	2.39E+02	2.19E+02
50	F_{18}	5.01E−09	2.47E−08	3.46E−07	4.82E−08
100		3.94E−09	1.70E−08	6.67E−08	1.96E−08
200		8.40E−09	1.98E−08	6.38E−08	2.53E−08
500		3.05E−08	5.23E−08	9.70E−08	5.59E−08
1,000		6.71E−08	1.18E−07	3.21E−07	1.30E−07
50	F_{19}	0.00E+00	0.00E+00	0.00E+00	0.00E+00
100		0.00E+00	0.00E+00	0.00E+00	0.00E+00
200		0.00E+00	0.00E+00	0.00E+00	0.00E+00
500		0.00E+00	0.00E+00	1.05E+00	4.20E−02
1,000		0.00E+00	0.00E+00	2.10E+00	3.78E−01

The results are summarized over 25 independent runs, and the values below 1.00E−14 are approximate to 0 as required

Comparing to classical DE, GaDE generally performed better, especially for functions $F_4, F_6, F_8, F_9, F_{11}, F_{12}, F_{14}, F_{16}$ and F_{18}. For functions F_1, F_5, F_7 and F_{15}, the two

Table 2 Comparison of mean errors among DE, CHC, G-CMA-ES and GaDE on F_1–F_{19} with dimensions $D = 50$, 100, 200, 500 and 1,000

D	F_i	DE	CHC	G-CMA-ES	GaDE
50	F_1	0.00E+00	1.67E−11	0.00E+00	0.00E+00
100		0.00E+00	3.56E−11	0.00E+00	0.00E+00
200		0.00E+00	8.34E−01	0.00E+00	0.00E+00
500		0.00E+00	2.84E−12	0.00E+00	0.00E+00
1,000		0.00E+00	1.36E−11	–	0.00E+00
50	F_2	3.60E−01	6.19E+01	2.75E−11	1.46E+01
100		4.45E+00	8.58E+01	1.51E−10	3.88E+01
200		1.92E+01	1.03E+02	1.16E−09	5.76E+01
500		5.35E+01	1.29E+02	3.48E−04	7.42E+01
1,000		8.46E+01	1.44E+02	–	8.93E+01
50	F_3	2.89E+01	1.25E+06	7.97E−01	1.18E+01
100		8.01E+01	4.19E+06	3.88E+00	5.89E+01
200		1.78E+02	2.01E+07	8.91E+01	1.61E+02
500		4.76E+02	1.14E+06	3.58E+02	4.40E+02
1,000		9.69E+02	8.75E+03	–	9.45E+02
50	F_4	3.98E−02	7.43E+01	1.05E+02	0.00E+00
100		7.96E−02	2.19E+02	2.50E+02	0.00E+00
200		1.27E−01	5.40E+02	6.48E+02	0.00E+00
500		3.20E−01	1.91E+03	2.10E+03	0.00E+00
1,000		1.44E+00	4.76E+03	–	0.00E+00
50	F_5	0.00E+00	1.67E−03	2.96E−04	0.00E+00
100		0.00E+00	3.83E−03	1.58E−03	0.00E+00
200		0.00E+00	8.76E−03	0.00E+00	0.00E+00
500		0.00E+00	6.98E−03	2.96E−04	0.00E+00
1,000		0.00E+00	7.02E−03	–	0.00E+00
50	F_6	1.43E−13	6.15E−07	2.09E+01	0.00E+00
100		3.10E−13	4.10E−07	2.12E+01	0.00E+00
200		6.54E−13	1.23E+00	2.14E+01	0.00E+00
500		1.65E−12	5.16E+00	2.15E+01	1.46E−14
1,000		3.29E−12	1.38E+01	–	1.66E−14
50	F_7	0.00E+00	2.66E−09	1.01E−10	0.00E+00
100		0.00E+00	1.40E−02	4.22E−04	0.00E+00
200		0.00E+00	2.59E−01	1.17E−01	0.00E+00
500		0.00E+00	1.27E−01	7.21E+153	0.00E+00
1,000		0.00E+00	3.52E−01	–	0.00E+00
50	F_8	3.44E+00	2.24E+02	0.00E+00	1.08E−08
100		3.69E+02	1.69E+03	0.00E+00	1.23E−03
200		5.53E+03	9.38E+03	0.00E+00	3.02E+00
500		6.09E+04	7.22E+04	2.36E−06	1.33E+03
1,000		2.46E+05	3.11E+05	–	1.77E+04
50	F_9	2.73E+02	3.10E+02	1.66E+01	6.24E−07
100		5.06E+02	5.86E+02	1.02E+02	3.87E−07
200		1.01E+03	1.19E+03	3.75E+02	4.53E−09
500		2.52E+03	3.00E+03	1.74E+03	0.00E+00
1,000		5.13E+03	6.11E+03	–	0.00E+00
50	F_{10}	0.00E+00	7.30E+00	6.81E+00	0.00E+00
100		0.00E+00	3.30E+01	1.66E+01	0.00E+00
200		0.00E+00	7.13E+01	4.43E+01	4.20E−02
500		0.00E+00	1.86E+02	1.27E+02	3.78E−01
1,000		0.00E+00	3.83E+02	–	4.62E−01
50	F_{11}	6.23E−05	2.16E+00	3.01E+01	1.31E−06
100		1.28E−04	7.32E+01	1.64E+02	4.34E−07
200		2.62E−04	3.85E+02	8.03E+02	1.85E−07
500		6.76E−04	1.81E+03	4.16E+03	0.00E+00
1,000		1.35E−03	4.82E+03	–	0.00E+00
50	F_{12}	5.35E−13	9.75E−01	1.88E+02	0.00E+00
100		5.99E−11	1.03E+01	4.17E+02	0.00E+00
200		9.76E−10	7.44E+01	9.06E+02	4.92E−14
500		7.07E−09	4.48E+02	2.58E+03	1.07E−12
1,000		1.68E−08	1.05E+03	–	3.85E−12
50	F_{13}	2.45E+01	2.08E+06	1.97E+02	1.19E+01
100		6.17E+01	2.70E+06	4.21E+02	4.99E+01
200		1.36E+02	5.75E+06	9.43E+02	1.24E+02
500		3.59E+02	3.22E+07	2.87E+03	3.34E+02
1,000		7.30E+02	6.66E+07	–	7.15E+02
50	F_{14}	4.16E−08	6.17E+01	1.09E+02	9.78E−13
100		4.79E−02	1.66E+02	2.55E+02	7.90E−13
200		1.38E−01	4.29E+02	6.09E+02	2.87E−12
500		1.35E−01	1.46E+03	1.95E+03	2.79E−11
1,000		6.90E−01	3.62E+03	–	8.82E−11
50	F_{15}	0.00E+00	3.98E−01	9.79E−04	0.00E+00
100		0.00E+00	8.13E+00	6.30E−01	0.00E+00
200		0.00E+00	2.14E+01	1.75E+00	0.00E+00
500		0.00E+00	6.01E+01	2.82E+262	0.00E+00
1,000		0.00E+00	8.37E+01	–	0.00E+00
50	F_{16}	1.56E−09	2.95E−09	4.27E+02	4.78E−12
100		3.58E−09	2.23E+01	8.59E+02	2.45E−12
200		7.46E−09	1.60E+02	1.92E+03	1.58E−12
500		2.04E−08	9.55E+02	5.45E+03	1.67E−12
1,000		4.18E−08	2.32E+03	–	2.35E−12
50	F_{17}	7.98E−01	2.26E+04	6.89E+02	4.97E−01
100		1.23E+01	1.47E+05	1.51E+03	3.28E+00
200		3.70E+01	1.75E+05	3.36E+03	2.45E+01
500		1.11E+02	8.40E+05	9.59E+03	9.26E+01
1,000		2.36E+02	2.04E+07	–	2.19E+02
50	F_{18}	1.22E−04	1.58E+01	1.31E+02	4.82E−08
100		2.98E−04	7.00E+01	3.07E+02	1.96E−08
200		4.73E−04	2.12E+02	6.89E+02	2.53E−08
500		1.22E−03	7.32E+02	2.05E+03	5.59E−08
1,000		2.37E−03	1.72E+03	–	1.30E-07
50	F_{19}	0.00E+00	3.59E+02	4.76E+00	0.00E+00
100		0.00E+00	5.45E+02	2.02E+01	0.00E+00
200		0.00E+00	2.06E+03	7.52E+02	0.00E+00
500		0.00E+00	1.76E+03	2.44E+06	4.20E-02
1,000		0.00E+00	4.20E+03	–	3.78E-01

The error values below 1.00E-14 are approximate to 0 as required

algorithms achieved exactly the same results. The classical DE has slightly advantages on only three functions F_2, F_{10} and F_{19}. This is consistent with the assumption that adaptive DE is able to handle more kinds of problems than classical DE with constant parameter settings.

Comparing to G-CMA-ES, GaDE also achieved better results on most of the functions. The superiority of GaDE is particularly significant on functions F_4–F_7 and F_9–F_{19}. Although the overall performance of G-CMA-ES is not good, it successfully solved some functions that both classical DE and GaDE failed to deal with. In particular, G-CMA-ES showed impressive results on functions F_2 and F_8. From Table 9 it can be found that both the two functions are unimodal, nonseparable and cannot be easily optimized dimension by dimension. In these cases, the use of covariance matrix adaptation in G-CMA-ES is able to learn the interactions between variables of nonseparable functions (Hansen and Kern 2004). However, this is often limited to unimodal nonseparable functions. G-CMA-ES still has difficult to deal with multimodal nonseparable functions, such as F_3, F_{13} and F_{17}. Moreover, learning and storing the covariance matrix during the evolution process is always very time consuming, which is one of the reason why the results of G-CMA-ES on 1,000-dimensional problems is not provided.

Real-coded CHC is the worst algorithms among all the tested ones. All its mean error values are much worse than that of other algorihtms. Hence, it might need further modification (or improvement) when applying to large-scale problems.

To compare the overall performance of all tested algorithms, we further provided Table 3 as a summary of the results with respect to the four performance indicators, i.e. *Best, Median, Worst* and *Mean*. For each algorithm, there are $19 \times 4 = 76$ indicators on the whole test suite with a given dimensionality. In Table 3, we collected the number of "wins" of each algorithm on all the 76 performance indicators. Here the term "win" means that an algorithm got the best result among all the compared algorithms with respect to an indicator. It can be found that GaDE obtained much more wins than others, which is consistent with the conclusions drawn based on the results in Tables 1 and 2.

The statistical analysis using Holm procedure[2] and Wilcoxon matched-pairs signed-rank test (García et al. 2009) on average results is provided in Tables 4 and 5. For the Holm procedure test with $\alpha = 0.05$, it can be found that GaDE is significantly better than G-CMA-ES and CHC, but the difference between GaDE and classical DE is not significant. For the Wilcoxon test with $p = 0.05$, it can be found that the differences between GaDE and G-CMA-ES, CHC are significant. In comparison with classical DE, the differences are also significant except on 500-dimensional problems. When increasing the p-value to the level 0.01, GaDE will be signifciantly better than classical DE on the testing suite with any dimensionality. The results of statistical tests confirm that GaDE is the best one among all the tested algorithms.

The scalability curves of the tested algorithms on some selected functions are listed in Fig. 1. For functions F_2 and F_8, it can found that although G-CMA-ES outperformed all other algorithms in terms of results, its performance deteriorate very quickly as problem scale increases. The other three algorithms have similar scalability performance. For function F_6, the performance of real-coded CHC dropped suddenly when the problem large than 200 dimension. Based on these scalability curves, GaDE has quite stable scalability performance, and it is clear better than others on four of the six functions.

[2] The Holm procedure test is not performed on the result of 1,000-dimensional functions because the results of G-CMA-ES are not available for such problems.

Table 3 The number of win's of all tested algorithms on functions F_1–F_{19}

D	GaDE	DE	G-CMA-ES	CHC	Sum (tie)
50	62	25	18	2	107 (31)
100	61	26	18	1	106 (30)
200	62	25	19	0	106 (30)
500	61	25	16	0	102 (26)
1,000	65	26	N/A	0	91 (15)

Table 4 The results of Holm procedure test of GaDE with other algorithms

D	i	Algs	z	p	α/i
50	1	CHC	5.780	0.000[†]	0.017
	2	G-CMA-ES	4.021	0.000[†]	0.025
	3	DE	1.508	0.132	0.050
100	1	CHC	5.780	0.000[†]	0.017
	2	G-CMA-ES	4.021	0.000[†]	0.025
	3	DE	1.508	0.132	0.050
200	1	CHC	5.906	0.000[†]	0.017
	2	G-CMA-ES	4.272	0.000[†]	0.025
	3	DE	1.634	0.102	0.050
500	1	CHC	5.152	0.000[†]	0.017
	2	G-CMA-ES	4.147	0.000[†]	0.025
	3	DE	1.257	0.208	0.050

[†] Result is significant with $\alpha = 0.05$

Table 5 The results of Wilcoxon matched-pairs signed-rank test of GaDE with other algorithms

D	v.s. DE	v.s. G-CMA-ES	v.s. CHC
50	1.34e−02[†]	2.90e−03[†]	1.32e−04[†]
100	1.34e−02[†]	3.80e−03[†]	1.32e−04[†]
200	2.45e−02[†]	1.40e−03[†]	1.32e−04[†]
500	5.54e−02	1.60e−03[†]	1.32e−04[†]
1,000	3.53e−02[†]	N/A	1.32e−04[†]

[†] Result is significant with $p < 0.05$

5.4 Efficacy of parameter adaptation and setting

Although the results in Tables 1 and 2 showed that the performance of GaDE is competitive, it is still not clear how the generalized parameter adaptation scheme works.

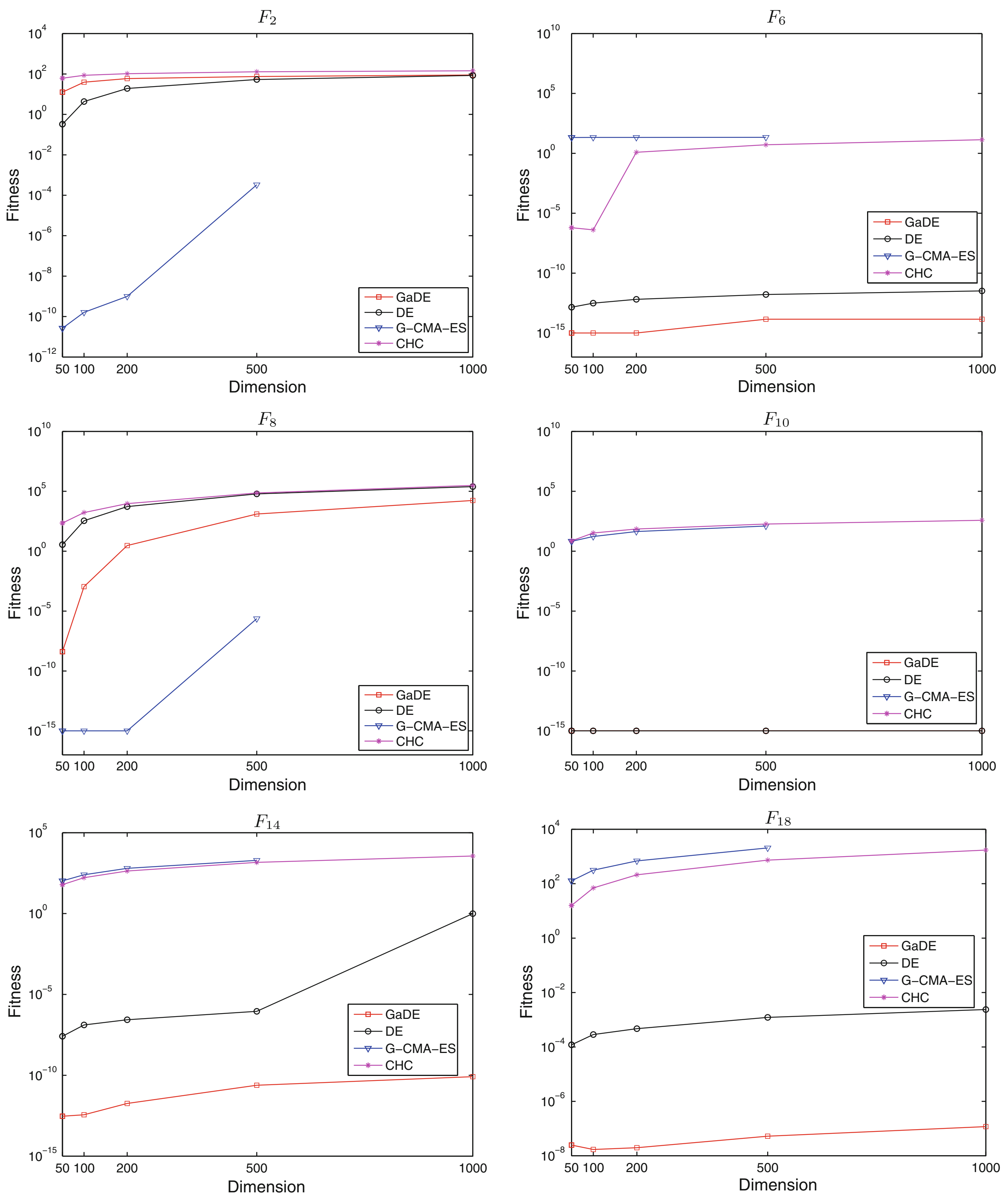

Fig. 1 The scalability curves of GaDE, DE, G-CMA-ES and real-coded CHC on functions $F_2, F_6, F_8, F_{10}, F_{14}$ and F_{18}

Table 6 Comparison of the mean errors among GaDE, GaDE$_{F=0.5}$, GaDE$_{\mathrm{CR}=0.9}$ and GaDE$_{mp=0.5}$ with $D = 1{,}000$

F_i	GaDE	GaDE$_{F=0.5}$	GaDE$_{\mathrm{CR}=0.9}$	GaDE$_{mp=0.5}$
F_1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_2	8.93E+01	8.38E+01	7.32E+01	**6.36E+01**†
F_3	**9.45E+02**†	9.53E+02	9.52E+02	9.56E+02
F_4	**0.00E+00**	**0.00E+00**	2.66E+01	**0.00E+00**
F_5	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_6	**1.66E−14**	3.21E−14	1.91E−14	1.93E−14
F_7	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_8	1.77E+04	2.12E+04	**1.16E+04**	2.08E+04
F_9	**0.00E+00**†v	2.96E−03	7.76E−02	5.04E−05
F_{10}	4.62E−01	**0.00E+00**†	8.78E+00	8.40E−02
F_{11}	**0.00E+00**†	2.94E−03	9.46E−02	4.75E−05
F_{12}	**3.85E−12**†	2.37E−07	4.29E−04	1.10E−10
F_{13}	**7.15E+02**	7.28E+02	7.17E+02	7.32E+02
F_{14}	**8.82E−11**†	1.70E−07	1.09E+01	3.46E−09
F_{15}	**0.00E+00**	**0.00E+00**	2.37E+00	4.20E−02
F_{16}	**2.35E−12**†	2.95E−07	7.06E−01	4.01E−11
F_{17}	2.19E+02	2.37E+02	**2.08E+02**†	2.30E+02
F_{18}	**1.30E−07**†	1.29E−04	4.10E−01	1.65E−06
F_{19}	3.78E−01	**0.00E+00**†	5.50E+00	2.52E−01

The best results in each row are given in bold face

† Result is significantly better than that of all other algorithms with $p < 0.05$

To validate its efficacy in GaDE, we further implemented the following three algorithms for comparison:

1. GaDE$_{F=0.5}$, which is the same as GaDE except that the scale factor F is set to a constant number 0.5.
2. GaDE$_{\mathrm{CR}=0.9}$, which is the same as GaDE except that the crossover rate CR is set to a constant number 0.9.
3. GaDE$_{mp=0.5}$, which is the same as GaDE except that the probability parameter mp for mutation scheme selection is set to a constant number 0.5.

The mean errors over 25 independent runs of these algorithms on 1,000-dimensional problems are summarized in Table 6.

It can be found that the GaDE, in which full parameter adaptation techniques were applied, achieved the best overall performance. Its results are at least the same good as others in 14 out of 19 functions, and seven of them are significantly better. Other algorithms have notable advantages on only a very small number of specific functions (e.g., F_2, F_{10}, F_{17} and F_{19}), which is inevitable according to the no free lunch theorems (Wolpert and Macready 1997). The comparison results confirmed that the generalized parameter adaptation scheme is generally better than conventional constant parameter settings.

In GaDE, a decreasing factor c was introduced in Eq. 34 to provide a trade-off between the previous and the current learning experience. The parameter was set to $c = 0.1$ empirically in previous experimental studies. To analyze the sensitivity of this parameter, we further tested two extra configurations:

1. GaDE$_{c=0}$, which is the same as GaDE except that the decreasing factor was set to $c = 0$.
2. GaDE$_{c=0.2}$, which is the same as GaDE except that the decreasing factor was set to $c = 0.2$.

The mean errors over 25 independent runs of these configurations on 1,000-dimensional problems are summarized in Table 7.

It can be found that the results of GaDE, in which the parameter c was set to 0.1, are as least as good as others on 16 out of 19 functions. That is why we chosen $c = 0.1$ as the default setting. As for other configurations, comparing to GaDE they have significant advantages on only three functions (i.e. F_2, F_{12} and F_{19}). Although the overall performance is not good, GaDE with other configurations still found the optima of at least six problems. This indicates that the introduced decreasing factor c is not very sensitive.

5.5 Computational running time analysis

The average running time of GaDE for each function is listed in Table 8. The programming language we used is C,

Table 7 Comparison of the mean errors among GaDE, GaDE$_{c=0}$ and GaDE$_{c=0.2}$ with $D = 1{,}000$

F_i	GaDE	GaDE$_{c=0}$	GaDE$_{c=0.2}$
F_1	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_2	8.93E+01	1.04E+02	**6.69E+01**†
F_3	**9.45E+02**	9.52E+02	9.48E+02
F_4	**0.00E+00**	4.82E−07	**0.00E+00**
F_5	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_6	**1.66E−14**†	2.42E−13	2.97E−14
F_7	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_8	**1.77E+04**†	3.28E+05	1.65E+05
F_9	**0.00E+00**	9.16E−03	**0.00E+00**
F_{10}	4.62E−01	**0.00E+00**†	2.52E−01
F_{11}	**0.00E+00**	9.33E−03	**0.00E+00**
F_{12}	**3.85E−12**†	6.86E−08	7.45E−12
F_{13}	**7.15E+02**†	7.26E+02	7.23E+02
F_{14}	**8.82E−11**†	4.98E−05	1.36E−10
F_{15}	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_{16}	**2.35E−12**†	1.77E−07	3.34E−12
F_{17}	**2.19E+02**	2.29E+02	**2.19E+02**
F_{18}	**1.30E−07**†	3.97E−04	1.82E−07
F_{19}	3.78E−01	**0.00E+00**†	2.10E−01

The best results in each row are given in bold face

† Result is significantly better than that of all other algorithms with $p < 0.05$

Table 8 The average running time (in seconds) of GaDE for each function over 25 independent runs

Function	$D = 50$	$D = 100$	$D = 200$	$D = 500$	$D = 1{,}000$
F_1	2	5	12	58	184
F_2	2	5	16	75	257
F_3	1	4	12	58	205
F_4	2	8	29	160	587
F_5	3	7	26	140	560
F_6	3	8	26	157	590
F_7	2	4	12	60	212
F_8	2	3	13	56	194
F_9	5	18	66	381	1,519
F_{10}	2	8	25	143	550
F_{11}	5	17	62	371	1,466
F_{12}	3	7	25	140	543
F_{13}	2	8	27	148	563
F_{14}	3	11	39	222	852
F_{15}	2	5	17	85	370
F_{16}	3	11	39	230	889
F_{17}	4	14	55	320	1,258
F_{18}	4	15	58	349	1,349
F_{19}	2	7	24	136	542

and the experiments were carried out using a cluster with eight 2.13 GHz cpu cores, 24 GB RAM. Given that the assigned maximal number of FEs is increased in the order of $O(D)$, it is expected that the running time for each function should increased as least in the order of $O(D)$. From Table 8, it can be found that for problems from $D = 50$ to 1,000 the running time incense rate is between 1.5 and 6.04. Considering that the time for each single fitness evaluation is also increased, the increase rate range [1.5, 6.04] is still acceptable. The presented running time suggests that the computational complexity of GaDE is low, and thus it is applicable for large-scale problems.

5.6 Discussion on benchmark functions

Usually we do not expect appealing results when applying EAs like DE to large-scale problems directly (Yang et al. 2008). However, the experimental results in this study showed that GaDE and DE actually performed very well. To find out the reason, we first list the properties of the used benchmark functions in Table 9. Since functions F_{12}–F_{19} are hybrid composition functions based on F_1–F_{11}, we only need to look at the first eleven functions.

Firstly, it can be found that most of the functions are unimodal. Only four functions, i.e. F_3–F_6, are multimodal. Unimodal functions are usually easier to optimize, especially when DE and GaDE used quite small population size NP = 60.

Table 9 The properties of F_1–F_{11} in the testing suite

Function	Unimodal/ multimodal	Shifted	Separable	Easily optimized dimension by dimension
F_1	U	Y	Y	Y
F_2	U	Y	N	N
F_3	M	Y	N	Y
F_4	M	Y	Y	Y
F_5	M	Y	N	N
F_6	M	Y	Y	Y
F_7	U	Y	Y	Y
F_8	U	Y	N	N
F_9	U	Y	N	Y
F_{10}	U	Y	N	N
F_{11}	U	Y	N	Y

U unimodal, *M* multimodal, *Y* yes, *N* no

Second, most of the functions can be easily optimized dimension by dimension. Table 9 shows that only F_2, F_5, F_8 and F_{10} are difficult to optimize using dimension by dimension method. Actually, the studies in Yang et al. (2008) showed that F_5 can also be optimized dimension by dimension. In DE and GaDE, exponential crossover rather than binomial crossover were employed. It means DE and GaDE mutate only a very small number of dimensions at a time because the probability to mutate m dimensions using exponential crossover is

$$p_m = \mathrm{CR}^m \tag{39}$$

Even if we set CR to a quite large value, say 0.9 in DE, the expected number of the mutated dimensions can still be small. For example, with $m = 50$, the probability of changing more than 50 dimensions at a time is $p_{50} = 0.9^{50} = 0.0052$, which is very low. As a result, DE and GaDE are able to solve the large-scale problems in the same way for small-scale problems. This has been proved using binomial crossover. In "experimental results of DE with exponential crossover and binomial crossover" (http://sci2s.ugr.es/eamhco/decrossvalues.xls), the performance of DE using binomial crossover are much worse than the presented results in this paper. It can be concluded that the large-scale problems presented in the used benchmark functions are not really large-scale for DE or adaptive DE using exponential crossover, because they always carry out the search in a very low-dimensional subspace.

6 Conclusion and future work

In this paper we focused on investigating the performance and scalability of adaptive DE algorithm. We first reviewed a number recently published adaptive DE variants. The similarities and drawbacks of the parameter adaptation techniques they used were analyzed. And then, we proposed a generalized parameter adaptation scheme, and employed it to design a new adaptive DE variant, GaDE. GaDE was evaluated on a testing suite including 19 benchmark functions, with dimensionality from 50 to 1,000.

In comparison with other algorithms, it was found that in overall GaDE outperformed not only classical DE, but also two other algorithms, G-CMA-ES and real-coded CHC. The experimental results were analyzed based on the characteristics of different benchmark functions. For separable functions, both classical DE and GaDE performed well, while G-CMA-ES encountered difficulty on most of the tested functions. GaDE is generally better than classical DE in the view of handling different problems without parameter tuning. For unimodal nonseparable functions, G-CMA-ES obtained especially good results due to its usage of covariance matrix adaptation. However, the high computational complexity of such techniques often restricts the scalability of G-CMA-ES. For multimodal nonseparable functions, all algorithms including G-CMA-ES performed poorly. Although GaDE was able to find better solutions than other algorithms, the obtained solutions were far away from the global optima. This kind of problems is often regarded as the real challenge of large-scale optimization. In the future work it would be interesting to focus on solving such problems by taking advantages of both GaDE and G-CMA-ES to develop more powerful search algorithms. One promising direction could be applying algorithm portfolio techniques between them (Peng et al. 2009).

Why DE and the proposed GaDE are good for the used benchmark functions was also discussed. It was found that most of these functions can be solved dimension by dimension. Thus, they have some limitations in reflecting all aspects of large-scale problems. More nonseparable large-scale benchmark functions, such as F_2 and F_8, can be consider in future research. Interested readers are referred to Tang et al. (2009), where more such type of benchmark functions are provided.

Acknowledgments This work was partially supported by the Fund for Foreign Scholars in University Research and Teaching Programs (Grant No. B07033), National Natural Science Foundation of China Grants (No. 60802036 and U0835002), and an EPSRC project (No. EP/D052785/1) on "SEBASE: Software Engineering By Automated SEarch".

References

Auger A, Hansen N (2005) A restart CMA evolution strategy with increasing population size. In: Proceedings of the 2005 IEEE congress on evolutionary computation, pp 1769–1776

Brest J, Greiner S, Boskovic B, Mernik M, Žumer V (2006) Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans Evol Comput 10(6):646–657

Benchmark functions for the Special Issue of *Soft Computing*. http://sci2s.ugr.es/eamhco/updated-functions1-19.pdf

Components and Parameters of DE, Real-coded CHC, and G-CMA-ES. http://sci2s.ugr.es/eamhco/descriptions.pdf

Das S, Abraham A, Chakraborty UK, Konar A (2009) Differential evolution using a neighborhood-based mutation operator. IEEE Trans Evol Comput 13(3):526–553

Eshelman LJ, Schaffer JD (1993) Real-coded genetic algorithms and interval-schemata. In: Whitley LD, (ed), Foundations of genetic algorithms 2. Morgan Kaufmann, pp 187–202

Eshelman LJ (1991) The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. Found Genetic Algorithms 1:265–283

Eshelman LJ, Schaffer JD (1993) Real-coded genetic algorithms and interval-schemata. Found Genetic Algorithms 2(1993):187–202

Experimental results of differential evolution with exponential crossover and binomial crossover. http://sci2s.ugr.es/eamhco/decross_values.xls

García S, Molina D, Lozano M, Herrera F (2009) A study on the use of non-parametric tests for analyzing the evolutionary algorithms behaviour: a case study on the CEC2005 special session on real parameter optimization. J Heuristics 15(6):617–644

Hansen N (2005) Compilation of results on the CEC benchmark function set, technical report, vol 13. Institute of Computational Science, ETH Zurich, Switerland

Hansen N, Ostermeier A (2001) Completely derandomized self-adaptation in evolution strategies. Evol Comput 9(2):159–195

Hansen N, Kern S (2004) Evaluating the CMA evolution strategy on multimodal test functions. In: Proceedings of the eighth international conference on parallel problem solving from nature (PPSN VIII), pp 282–291

Liu Y, Yao X, Zhao Q, Higuchi T (2001) Scaling up fast evolutionary programming with cooperative coevolution. In: Proceedings of the 2001 congress on evolutionary computation, pp 1101–1108

Lee CY, Yao X (2004) Evolutionary programming using mutations based on the Lévy probability distribution. IEEE Trans Evol Comput 8(1):1–13

Neri F, Tirronen V (2010) Recent advances in differential evolution: a survey and experimental analysis. Artif Intell Rev 33(1):61–106

Price KV, Storn RM, Lampinen JA (2005) Differential evolution: a practical approach to global optimization. Springer, Berlin. ISBN:3-540-20950-6

Peng F, Tang K, Chen G, Yao X (2009) Population-based algorithm portfolios for numerical optimization. IEEE Trans Evol Comput (in press)

Qin AK, Suganthan PN (2005) Self-adaptive differential evolution algorithm for numerical optimization. In: Proceedings of the 2005 IEEE congress on evolutionary computation, vol 2, pp 1785–1791

Qin AK, Huang VL, Suganthan PN (2009) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evol Comput 13(2):398–417

Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Global Optimization 11(4):341–359

Storn R (1999) System design by constraint adaptation and differential evolution. IEEE Trans Evol Comput 3(1):22–34

Tang K, Li X, Suganthan PN, Yang Z, Weise T (2009) Benchmark Functions for the CEC2010 Special session and competition on large-scale global optimization, technical report. Nature Inspired Computation and Applications Laboratory, University of Science and Technology of China, China. http://nical.ustc.edu.cn/cec10ss.php

Vesterstrom J, Thomsen R (2004) A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In: Proceedings of the 2004 IEEE congress on evolutionary computation, vol 2, pp 1980–1987

Whitley D, Rana S, Dzubera J, Mathias KE (1996) Evaluating evolutionary algorithms. Artif Intell 85(1–2):245–276

Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82

Yang Z, Tang K, Yao X (2008) Large scale evolutionary optimization using cooperative coevolution. Inf Sci 178(15):2985–2999

Yang Z, He J, Yao X (2008) Making a difference to differential evolution. In: Michalewicz Z, Siarry P, (eds) Advances in metaheuristics for hard optimization. Springer, Berlin, pp 397–414

Yang Z, Tang K, Yao X (2008) Self-adaptive differential evolution with neighborhood search. In: Proceedings of the 2008 IEEE congress on evolutionary computation, pp 1110–1116

Yang Z, Zhang J, Tang K, Yao X, Sanderson AC (2009) An adaptive coevolutionary differential evolution algorithm for large-scale optimization. In: Proceedings of the 2009 IEEE congress on evolutionary computation, pp 102–109

Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. IEEE Trans Evol Comput 3(2):82–102

Zaharie D (2002) Critical values for the control parameters of differential evolution algorithms. In: Proceedings of the 8th international conference on soft computing, pp 62–67

Zhang J, Sanderson AC (2009) JADE: adaptive differential evolution with optional external archive. IEEE Trans Evol Comput 13(5):945–958