He RJ, Yang ZY. Differential evolution with adaptive mutation and parameter control using Lévy probability distribution. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 27(5): 1035–1055 Sept. 2012. DOI 10.1007/s11390-012-1283-3

Differential Evolution with Adaptive Mutation and Parameter Control Using Lévy Probability Distribution

Ren-Jie He (贺仁杰) and Zhen-Yu Yang* (杨振宇), *Member*, *CCF*, *ACM*, *IEEE*

Department of Management Science and Engineering, College of Information System and Management, National University of Defense Technology, Changsha 410073, China

E-mail: renjiehe@nudt.edu.cn; zhyuyang@mail.ustc.edu.cn

Received October 14, 2011; revised February 23, 2012.

Abstract Differential evolution (DE) has become a very popular and effective global optimization algorithm in the area of evolutionary computation. In spite of many advantages such as conceptual simplicity, high efficiency and ease of use, DE has two main components, i.e., mutation scheme and parameter control, which significantly influence its performance. In this paper we intend to improve the performance of DE by using carefully considered strategies for both of the two components. We first design an adaptive mutation scheme, which adaptively makes use of the bias of superior individuals when generating new solutions. Although introducing such a bias is not a new idea, existing methods often use heuristic rules to control the bias. They can hardly maintain the appropriate balance between exploration and exploitation during the search process, because the preferred bias is often problem and evolution-stage dependent. Instead of using any fixed rule, a novel strategy is adopted in the new adaptive mutation scheme to adjust the bias dynamically based on the identified local fitness landscape captured by the current population. As for the other component, i.e., parameter control, we propose a mechanism by using the Lévy probability distribution to adaptively control the scale factor F of DE. For every mutation in each generation, an F_i is produced from one of four different Lévy distributions according to their historical performance. With the adaptive mutation scheme and parameter control using Lévy distribution as the main components, we present a new DE variant called Lévy DE (LDE). Experimental studies were carried out on a broad range of benchmark functions in global numerical optimization. The results show that LDE is very competitive, and both of the two main components have contributed to its overall performance. The scalability of LDE is also discussed by conducting experiments on some selected benchmark functions with dimensions from 30 to 200.

Keywords differential evolution, global optimization, Lévy distribution, parameter adaptation

1 Introduction

Differential evolution (DE), proposed by Storn and Price[1], has become a very popular evolutionary algorithm (EA) for global numerical optimization problems. It has shown superior performance in solving both function optimization benchmarks and real-world applications[2-4]. Like other evolutionary algorithms (EAs), DE adopts a population-based iterative stochastic search procedure, and in each generation it uses mutation, crossover and selection operators to move the current population toward the global optimum. In the classical DE[1] there are only three control parameters, i.e., the population size NP, the mutation scale factor F, and the crossover rate CR. Such a desired feature makes it very easy to use in practice.

Besides the classical DE, many DE variants have been proposed in the last decade to study and improve its performance. The previous work found that the performance of DE mainly depends on its two components. One is the mutation scheme, which provides the original driving force for solution variations. In the classical DE, mutation is implemented by adding a weighted difference vector between two individuals to a third individual[1]. Although such a classical mutation scheme based on three randomly selected individuals is quite simple and effective in some scenarios, it forms the difference vector only randomly, and thus is often not efficient enough. Some more advanced ideas are to formulate the mutation equation with the bias of some superior individuals in the current population[5-6], or even utilize multiple different mutation schemes simultaneously[7-8]. In spite of these methods did achieve better results, it is often very hard

Regular Paper

This work was partially supported by the National Natural Science Foundation of China under Grant Nos. 71071106 and 70801062.

*Corresponding Author

to determine what kind of individuals should be included in the mutation equation to provide such a bias, and in practice only some heuristic rules (e.g., choose the best individual so far) were suggested. There still lacks a technique, which can adaptively control the bias, to maintain a reasonable balance between exploration and exploitation during the search process. The other component that affects DE's performance significantly is how to control its parameters. Empirical parameter settings are not sufficient, because DE's performance is very sensitive to them, and there is no fixed parameter setting that is suitable for various problems or even at different evolution stages of a single problem[9]. In addition, suitable parameter settings also correlate with the choice of mutation scheme. The work in [10] found that it is beneficial to provide multiple choices for the generation of trial solutions by randomly combining three mutation schemes with three parameter settings. A commonly used parameter control technique is to generate parameter values randomly according to a certain probability distribution, and then adaptively adjust the probability distribution in the succeeding generations based on the feedback of the search process. In particular, DE with parameter control using the uniform distribution[11-12], the Gaussian distribution[13-14] and the Cauchy distribution[8,15] were frequently investigated. However, which probability distribution is better and can they fit into a more general distribution are still not clear.

In this paper, we aim at improving the performance of DE by utilizing more carefully considered strategies for the two important components discussed in the last paragraph. To be specific, for the first component we propose an adaptive mutation scheme that can adaptively make use of the bias of superior individuals in generating mutated individuals. A novel strategy is adopted to adjust the bias dynamically based on the identified local fitness landscape captured by the current population. For the other component, we design a mechanism based on the Lévy distribution to adaptively control the scale factor F of DE. For every mutation in each generation, an F_i is produced from one of four predefined Lévy distributions according to their historical performance. The motivation of introducing the Lévy distribution is as follows. As one of the most important parameters of DE, it was found that the scale factor F has direct influence on DE's search step size[15]. Inspired by the search step size control strategies in evolutionary programming (EP)[16-17], both Gaussian distribution and Cauchy distribution, which are special cases of more generalized Lévy distribution, have been introduced into DE to control the parameter F appropriately. Although Lévy distribution was proven to be better for search step size controlling[18], it has been seldom used in DE. That is why DE with parameter control using Lévy distribution deserves more investigation. With the adaptive mutation scheme and parameter control using Lévy distribution as the main contributions, we present a new DE variant, called Lévy DE (LDE). Experiments have been conducted to evaluate the efficacy of LDE and its components on a test suite with 28 benchmark functions. The results are compared with the classical DE, three state-of-the-art DE variants and two EP variants. To further study its scalability, LDE has also been evaluated on some selected benchmark functions with dimensionality from 30 to 200.

The rest of this paper is organized as follows. Section 2 gives some preliminaries, including the problem formulation, the classical DE and the Lévy distribution. Section 3 provides a review on recent adaptive DE variants. Section 4 describes the proposed LDE algorithm. Section 5 presents the experimental studies and discussions. Finally, Section 6 concludes this paper briefly with several remarks and future research directions.

2 Preliminaries

2.1 Global Numerical Optimization

In this paper we have chosen global numerical optimization as the test-bed. According to the description in [17], a global numerical optimization problem① can be formalized as a pair $(\mathbb{S}, f)$, where $\mathbb{S} \subseteq \mathbb{R}^D$ is a bounded set on $\mathbb{R}^D$, and $f : \mathbb{S} \mapsto \mathbb{R}$ is a D-dimensional real-valued function. The problem is to find a point $\boldsymbol{x}^* \in \mathbb{S}$ such that $f(\boldsymbol{x}^*)$ is a global optimum on $\mathbb{S}$. More specifically, the task is to find an $\boldsymbol{x}^* \in \mathbb{S}$ such that

$$\forall \boldsymbol{x} \in \mathbb{S} : f(\boldsymbol{x}^*) \leqslant f(\boldsymbol{x}), \tag{1}$$

where f is not required to be continuous, but it must be bounded. Here we consider only single-objective unconstrained (or bound-constrained) numerical optimization. The key difficulty in global numerical optimization is to locate the global optimum efficiently without being trapped in local optima.

2.2 Classical DE

DE algorithm aims at evolving a population of NP individuals, i.e., $\boldsymbol{x}_i, \forall i \in \{1, \ldots, NP\}$, for problem solving. The parameter NP is called population size. Each individual in the population is corresponding to a candidate solution in the search space, and for numerical optimization problems it is often represented using a

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

D-dimensional real-valued vector, where D is the number of decision variables. The main operations of the classical DE can be summarized as follows[2]:

1) *Mutation*:

$$\boldsymbol{v}_i = \boldsymbol{x}_{r_1} + F \cdot (\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3}), \tag{2}$$

where $r_1, r_2, r_3 \in \{1, \ldots, NP\}$ are random and mutually different integers, and they are also different with the index i. Scale factor $F \in (0, 2]$ is a real constant factor and it is often set to 0.5 empirically.

2) *Crossover*:

$$\boldsymbol{u}_i(j) = \begin{cases} \boldsymbol{v}_i(j), & \text{if } U_j(0,1) \leqslant CR \text{ or } j = j_r, \\ \boldsymbol{x}_i(j), & \text{otherwise,} \end{cases} \tag{3}$$

where $U_j(0, 1)$ stands for a uniform random number between 0 and 1, and $j_r \in \{1, \ldots, NP\}$ is a randomly chosen index to ensure that the trial solution $\boldsymbol{u}_i$ does not duplicate $\boldsymbol{x}_i$. The parameter $CR \in [0, 1]$ is the crossover rate, which is often set to 0.9 empirically. The crossover is called binomial crossover in DE.

3) *Selection*:

$$\boldsymbol{x}'_i = \begin{cases} \boldsymbol{u}_i, & \text{if } f(\boldsymbol{u}_i) \leqslant f(\boldsymbol{x}_i), \\ \boldsymbol{x}_i, & \text{otherwise,} \end{cases} \tag{4}$$

where $\boldsymbol{x}'_i$ is the offspring of $\boldsymbol{x}_i$ for the next generation.

In spite of the classical one described above, there are several variants of DE based on different mutation schemes they adopt. Some popular mutation schemes are listed as follows[2].

$$\boldsymbol{v}_i = \boldsymbol{x}_{r_1} + F \cdot (\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3}), \tag{5}$$

$$\boldsymbol{v}_i = \boldsymbol{x}_{\text{best}} + F \cdot (\boldsymbol{x}_{r_1} - \boldsymbol{x}_{r_2}), \tag{6}$$

$$\boldsymbol{v}_i = \boldsymbol{x}_{r_1} + F \cdot (\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3} + \boldsymbol{x}_{r_4} - \boldsymbol{x}_{r_5}), \tag{7}$$

$$\boldsymbol{v}_i = \boldsymbol{x}_{\text{best}} + F \cdot (\boldsymbol{x}_{r_1} - \boldsymbol{x}_{r_2} + \boldsymbol{x}_{r_3} - \boldsymbol{x}_{r_4}), \tag{8}$$

$$\boldsymbol{v}_i = \boldsymbol{x}_i + F \cdot (\boldsymbol{x}_{\text{best}} - \boldsymbol{x}_i + \boldsymbol{x}_{r_1} - \boldsymbol{x}_{r_2}). \tag{9}$$

Schemes (5) and (9), with notations as "DE/rand/1" and "DE/current-to-best/2", are frequently used in practice due to their superior performance[2-3,7].

Binomial crossover is also not the only choice for the crossover operation in DE. Recent studies indicate that exponential crossover[19] appears to be better for some large-scale optimization problems[20]. Due to design preference, both the binomial crossover and exponential crossover only visit one vertex of the hyper-rectangle defined by the mutant and target vectors. To complement such an issue, Wang *et al.*[21] introduced a quantization orthogonal crossover (QOX) into DE. It was found that the QOX operator is effective and efficient in a number of numerical optimization test instances.

2.3 Lévy Distribution

The Lévy distribution, which is a class of probability distributions having an *infinite* second central moment and governing the sum of these random variables[22-23], was discovered by P. Lévy in the 1930s. According to the description in [22-23], it can be given in the following form:

$$L_{\alpha,\gamma}(y) = \frac{1}{\pi} \int_0^\infty e^{-\gamma q^\alpha} \cos(qy) \mathrm{d}q, \quad y \in \mathbb{R}. \tag{10}$$

The distribution is symmetric with respect to $y = 0$ and has two parameters γ and α. γ is the scaling factor satisfying $\gamma > 0$, and α controls the shape of the distribution, requiring $\alpha \in (0, 2)$. The analytic form of the integral is unknown for general α, but is known for a few cases. In particular, for $\alpha = 1$, the integral can be reduced to the analytic form resulting in the Cauchy distribution, and in the limit of $\alpha \to 2$, the distribution is no longer the Lévy distribution and becomes the Gaussian distribution[18].

3 Review of Recent Adaptive DE Variants

3.1 jDE

jDE was proposed by Brest *et al.*[11] based on the self-adaptation of the scale factor F, and the crossover rate CR. It associates each individual with its own control parameters F_i and CR_i. Before the algorithm starts, F_i and CR_i are set to 0.5 and 0.9 respectively. And then, at each generation G new values for F_i and CR_i are produced as follows:

$$F_{i,G+1} = \begin{cases} 0.1 + 0.9 \times rand_1, & \text{if } rand_2 < \tau_1, \\ F_{i,G}, & \text{otherwise,} \end{cases} \tag{11}$$

and

$$CR_{i,G+1} = \begin{cases} 0.1 + 0.9 \times rand_3, & \text{if } rand_4 < \tau_2, \\ CR_{i,G}, & \text{otherwise,} \end{cases} \tag{12}$$

where $rand_j$, $j = 1, 2, 3, 4$, 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 survives in the selection process.

3.2 SaDE

SaDE proposed by Qin *et al.*[7,24] gave the first attempt to simultaneously adopt more than one mutation schemes in DE. The main propose is to reduce the

problem-solving risk by distributing available computational resources to multiple search techniques with different biases. Some similar ideas have been successfully used to build algorithm portfolios[25]. The utilized adaptation techniques in SaDE consist of two parts: 1) the adaptation of the probability, p_k, of applying the k-th mutation scheme for each individual; and 2) the adaptation of the parameters F and CR.

For the adaptation of mutation schemes, the probability of applying the k-th mutation scheme is represented using p_k, $k = 1, 2, \ldots, m$, where m is the total number of candidate schemes. All the p_k's are initialized as $\frac{1}{m}$, which means all schemes have the same probability to be chosen. Four different mutation schemes, i.e., "DE/rand/1", "DE/rand-to-best/2", "DE/rand/2" and "DE/current-to-rand/1", are used as candidates in SaDE[7].

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 recorded as $nf_{k,G}$. And then the probability of choosing the k-th scheme is updated by

$$p_{k,G} = \frac{S_{k,G}}{\sum_{k=1}^{m} S_{k,G}}, \quad k = 1, \ldots, m, \tag{13}$$

with

$$S_{k,G} = \frac{\sum_{g=G-LP}^{G-1} ns_{k,g}}{\sum_{g=G-LP}^{G-1} ns_{k,g} + \sum_{g=G-LP}^{G-1} nf_{k,g}} + \epsilon, \tag{14}$$

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 rate.

For the adaptation of the 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.:

$$F_i = N_i(0.5, 0.3), \tag{15}$$

where $N_i(\mu, \delta^2)$ denotes a Gaussian random number with mean μ and standard deviation δ. All F_i will be truncated to the interval $(0, 2]$ based on the empirical range of F.

For the adaptation of crossover rate CR, individual-based CR_i are generated every 5 generations according to a Gaussian distribution with mean CRm and standard deviation 0.1, i.e.:

$$CR_i = N_i(CRm, 0.1). \tag{16}$$

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

$$CRm = \frac{1}{|CR_{\text{suc}}|} \sum_{j=1}^{|CR_{\text{suc}}|} CR_{\text{suc}}(j), \tag{17}$$

where CR_{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 [8] can be regarded as an improved version of the SaDE in Subsection 3.2. Its mutation is executed in the same way as SaDE except that only two mutation schemes are used, and the scale factor F in the adopted mutation schemes are generated according to either a Gaussian distribution or a Cauchy distribution, i.e.,

$$F_i = \begin{cases} N_i(0.5, 0.3), & \text{if } U_i(0, 1) < fp, \\ C_i(0, 1), & \text{otherwise}, \end{cases} \tag{18}$$

where $C_i(\mu, \delta)$ denotes a random value from a Cauchy distribution with location and scale parameters μ, δ. The probability, fp, of applying either of the two distributions is controlled adaptively in a manner similar to (13) in SaDE.

As for the adaptation of the crossover rate CR, SaNSDE follows the method in SaDE except that a weighting strategy is applied:

$$CRm = \sum_{k=1}^{|CR_{\text{suc}}|} w_k \times CR_{\text{suc}}(k), \tag{19}$$

$$w_k = \delta_f(k) \Big/ \Big(\sum_{k=1}^{|\delta_f|} \delta_f(k) \Big), \tag{20}$$

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

3.4 JADE

JADE[6] is another recent DE variant, in which a new mutation scheme named "DE/current-to-$p$best" is adopted. The mutated individuals are generated as follows:

$$\boldsymbol{v}_i = \boldsymbol{x}_i + F_i \cdot (\boldsymbol{x}_{p\,\text{best}} - \boldsymbol{x}_i + \boldsymbol{x}_{r_1} - \tilde{\boldsymbol{x}}_{r_2}), \tag{21}$$

where $\boldsymbol{x}_{p\,\text{best}}$ is randomly chosen as one of the best $100p\%$ individuals in the current population with $p \in$

$(0, 1]$. Individual $\boldsymbol{x}_{r_1}(r_1 \neq i)$ is randomly selected from the population, and $\tilde{\boldsymbol{x}}_{r_2}$ is an individual (distinct from $\boldsymbol{x}_i$ and $\boldsymbol{x}_{r_1}$) 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.

At each generation, F_i and CR_i in JADE are randomly generated for each individual according to the following equations:

$$F_i = C_i(F_m, 0.1), \tag{22}$$

$$CR_i = N_i(CR_m, 0.1), \tag{23}$$

where CR_m and F_m are updated in the following adaptive manner:

$$Fm = (1 - c) \times Fm + c \times \text{mean}_L(F_{\text{suc}}), \tag{24}$$

$$CRm = (1 - c) \times CRm + c \times \text{mean}_A(CR_{\text{suc}}), \tag{25}$$

where CR_{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. $\text{mean}_A(\cdot)$ is the usual arithmetic mean and $\text{mean}_L(\cdot)$ is the Lehmer mean:

$$\text{mean}_L(F_{\text{suc}}) = \frac{\sum_{F \in F_{\text{suc}}} F^2}{\sum_{F \in F_{\text{suc}}} F}, \tag{26}$$

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

4 DE with Adaptive Mutation and Parameter Control Using Lévy Distribution

4.1 Motivation and Adaptive Mutation Scheme

As mentioned in Section 1, mutation is one of the key operators in DE. It is actually the only operator that can provide the original driving force for individual variations. Although the idea of utilizing the differences among individuals in mutation is simple enough, it is not easy to determine how the mutation scheme should be formulated. The classical mutation scheme, which is given as (5) based on three randomly selected individuals, is frequently used as a fair choice. However, since such a scheme forms the difference vector only randomly, it is often not efficient enough, and thus slows down the overall convergence speed of DE. In designing more efficient mutation schemes, a commonly used idea is to take advantage of the bias of some superior individuals in the population. The schemes given as (6), (8) and (9) are of this kind. They all generate new individuals with the consideration of the best individual found so far. Although with this kind of mutation DE's performance could be improved in some scenarios, a number of side effects are also introduced unfortunately. The most notable one is that the probability of the algorithm to be trapped in the local optima of multimodal problems is increased significantly, compared with the classical DE[6]. So how to maintain the balance between global optimization ability (i.e., exploration) and search efficiency (i.e., exploitation) has become a very important issue in designing suitable mutation schemes for DE.

The idea of balancing exploration and exploitation has already been used to design more advanced mutation schemes for DE. The SaDE[7,24] described in Subsection 3.2 simultaneously adopts more than one mutation schemes with different search characteristics. A new parameter is introduced for each mutation scheme to control the probability to use it for each individual, and it is adaptively updated based on the performance of a corresponding mutation scheme in a predefined *learning period*. The JADE[6] described in Subsection 3.4 intends to acquire such a balance by proposing a new mutation scheme "DE/current-to-$p$best", as given in (21). The basic idea is to perform the mutation with the bias of one of the best $100p\%$ individuals in the current population, where $p \in (0, 1]$ is a new introduced probability parameter. Although both SaDE and JADE have achieved great success over the classical DE, there is still room for improvements. For SaDE, it is often hard to determine which and how many mutation schemes should be included as the candidates. As for JADE, the optimal setting of the new parameter p is a hard optimization problem itself. Only a fixed value in an empirical interval $[5\%, 20\%]$ was suggested[6].

In order to seek for a more appropriate trade-off between exploration and exploitation, in this paper we propose a new mutation scheme as follows:

$$\boldsymbol{v}_i = \boldsymbol{x}_{r_1} + F_i \cdot (\tilde{\boldsymbol{x}}_{p\,\text{best}} - \boldsymbol{x}_{r_1} + \boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3}), \tag{27}$$

where $r_1, r_2, r_3 \in [1, NP]$ are random and mutually different integers, and $\tilde{\boldsymbol{x}}_{p\,\text{best}}$ is a member randomly chosen from a set S^p, which is generated at the beginning of each generation according to the procedure described in Algorithm 1, where

p: probability parameter,
NP: populationsize,
POP: the current population,
$\max(a, b)$: return the larger one between a and b,
$round(a)$: return the nearest integer to a.

Algorithm 1: Procedure to Generate the Set S^p for $\tilde{\boldsymbol{x}}_{p\,\text{best}}$ Candidates

Input: p, NP, $POP = \{\boldsymbol{x}_i |\ 1 \leqslant i \leqslant NP\}$
Output: S^p

```
S^p = ∅;
elemNum = max(round(p × NP), 1);
neighborNum = max(round(NP/elemNum), 1);
while (POP ≠ ∅) do
    Select the best individual x_best from POP;
    S^p = S^p ∪ {x_best};
    POP = POP − {x_best};
    j = 1;
    while (j < neighborNum) and (POP ≠ ∅) do
        Select the nearest neighbor x to x_best from
        POP, with euclidean distance as the metric;
        POP = POP − {x};
        j = j + 1;
    end
end
return S^p;
```

Based on the traditional naming rule in DE research, the new mutation scheme can be denoted as "DE/rand-to-$p$best/2". "DE/rand-to-$p$best/2" given as (27) may look similar to the "DE/current-to-$p$best/2" of JADE in (21). But they are actually quite different in at least the following aspects:

1) In the scheme "DE/rand-to-$p$best/2", the base vector, i.e., $\boldsymbol{x}_{r_1}$, is selected randomly. Compared with JADE, the number of different possible mutations is increased from $NP(NP-1)$ to $NP(NP-1)(NP-2)$. This would enhance the flexibility of mutation in DE.

2) In JADE, $\boldsymbol{x}_{p\,\text{best}}$ is randomly chosen from the best $100p\%$ individuals in the current population. If all these individuals are in the attractive basin of a single local optimum, the risk of DE to be trapped in local optima is very high. In contrast, $\tilde{\boldsymbol{x}}_{p\,\text{best}}$ is selected based on not only its fitness, but also its location in the distribution of the current population. The search process is not likely to be overwhelmingly affected by any single local optimum.

The crucial idea of the new mutation scheme "DE/rand-to-$p$best/2" is to efficiently utilize the bias of superior individuals without introducing the high risk of being trapped by local optima, which is desired in balancing exploration and exploitation. Like the mutation in JADE, a new probability parameter, p, is introduced in "DE/rand-to-$p$best/2" as well. Instead of using any fixed value, it would be nice if we have an adaptive strategy to set and adjust it automatically.

From Algorithm 1 it is easy to find out the scheme "DE/rand-to-$p$best/2" degenerates to "DE/rand/2" in (7) if $p = 1$, and it is more like "DE/current-to-best/2" in (9) when $p \leqslant \frac{1}{NP}$. As p decreases from 1 to $\frac{1}{NP}$, "DE/rand-to-$p$best/2" places more and more emphasis on the bias of top individuals in the current population. For unimodal problems, it is good to set p to a small value since it will help DE converge faster. However, for multimodal problems such a setting will increase the probability of premature convergence, and thus large p values are more suitable. This can be used as the basic guideline to set the parameter p. Although the precise characteristics (e.g., unimodal or multimodal) of an unknown objective problem are often not available, we can use some approximate approaches to have a guess based on the local fitness landscape represented by the current population. Since each individual is in a population, the population forms an observation of the local fitness landscape. By sorting other individuals in the population based on their distances to the best individual in the current population, we can check how the fitness of each changes over the distance[26]. If the fitness is increasing with the distance, then the local fitness landscape is like a unimodal landscape; otherwise, it belongs to a multimodal landscape. Based on the simple idea, the following procedure is proposed in [26] to detect problem characteristics.

1) Find out the best individual, $\boldsymbol{x}_{\text{best}}$, among the population, and then calculate the distance between each individual $\boldsymbol{x}_i$, $i = 1, \ldots, NP$ and $\boldsymbol{x}_{\text{best}}$ with a certain distance metric:

$$d_i = |\boldsymbol{x}_i - \boldsymbol{x}_{\text{best}}|. \tag{28}$$

The Euclidean distance will be used as the distance metric in this paper.

2) Sort the individuals based on the distance value in ascending order: $\boldsymbol{x}_{s_1}, \ldots, \boldsymbol{x}_{s_{NP}}$.

3) Introduce a metric χ to measure the shape of the local fitness landscape. Assume χ to be 0 initially, and then it will be increased by 1 if $f(\boldsymbol{x}_{s_{i+1}}) \leqslant f(\boldsymbol{x}_{s_i})$. Normalize the value as:

$$\varphi = \frac{\chi}{NP}. \tag{29}$$

As local fitness landscape is actual a fuzzy concept, the indicator φ can be regarded as the roughness of observed fitness landscape. If $\varphi = 0$, the local fitness landscape is more like a unimodal landscape; if $\varphi = 1$, the local fitness landscape is very rough, and thus is more likely to be multimodal. So basically φ can be used to set the parameter p in the mutation scheme "DE/rand-to-$p$best/2". However, since the detection procedure is not always precise, it is not proper to set $p = \varphi$ directly. For example, set p very close to 0, which means to use only the bias of $\boldsymbol{x}_{\text{best}}$, is quite risky

if a multimodal problem is regarded as unimodal by mistake. A lower bound p_l for p should be introduced to avoid such a case. On the other hand, set p very close to 1, which means to perform the mutation randomly, is not a clever choice either because even for multimodal problems the bias of superior individuals is still very helpful for the algorithm to make progress. So an upper bound p_u for p is also needed. Based on these discussions, in this paper the parameter p is set as:

$$p = p_l + (p_u - p_l) \times \varphi, \tag{30}$$

where p_l and p_u are the lower and upper bounds, respectively. According to the study in [6], it is reasonable to set $p_l = 0.05$. As for the upper bound, we suggest $p = 0.5$ which means that at least the top 50% individuals of the current population should be used to provide the search bias.

With (27) and (30), we have presented a new mutation scheme "DE/rand-to-*p*best/2". The new scheme is expected to be able to maintain more appropriate balance between exploration and exploitation by adaptively adjusting the bias of superior individuals dynamically.

It is notable that the adaptive mutation scheme may have some computational overhead compared with the classical mutation scheme, because the distance between some individuals is needed when selecting the superior individuals and updating the parameter p. In the worst case, the distance is calculated between any two individuals. So the computational complexity of calculating these distances is $NP \times (NP-1)/2$, where NP is the population size. Since we use the euclidean distance, which can be calculated very fast, as the distance metric, the computational overhead in the adaptive mutation would not be significant as compared with the time spent on fitness evaluations.

4.2 Parameter Control Using Lévy Distribution

Search step size controlling is very important in EAs, because it affects the efficiency of the entire search process significantly[27]. Based on the main operations given in Subsection 2.2, the search step size of DE mainly depends on two parts. The first part is the difference vector, which is formed using the difference between selected individuals. The other part is the scale factor F which is used to scale the difference vector. Since the difference vector is determined by the diversity of the current population and thus not easy to manipulate, the adaptation of the scale factor F has been used as one of the main strategies in seeking for appropriate search step size controlling in DE.

Inspired by the related work on search step size controlling in EP[17], both Gaussian and Cauchy distributions, which are two special cases of the more generalized Lévy distribution, have been successfully introduced into DE for the adaptation of the scale factor F[7-8]. Although Lévy distribution was found even better for search step size controlling[18], it has seldom been used in DE. To study the characteristics of Lévy distribution for search step size controlling in DE, we propose a new adaptive strategy to control the scale factor F. Firstly, an F_i is associated with every mutation in each generation. Then F_i will be produced using one Lévy distribution selected from the candidate set $\{L(\alpha_j, \gamma)|j = 1, 2, \ldots, m\}$ with the probability $\{\psi_j|j = 1, 2, \ldots, m\}$ respectively, i.e.,

$$F_i = L(\alpha_j, \gamma), \quad \text{if } \sum_{k=1}^{j-1} \psi_k < U_i(0,1) \leqslant \sum_{k=1}^{j} \psi_k, \tag{31}$$

where α_j and γ are the parameters of Lévy distribution, and $U_i(0,1)$ stands for a uniform random number between 0 and 1. Obviously, the condition $\sum_{j=1}^{m} \psi_j \equiv 1$ holds for the parameters $\{\psi_j|j = 1, 2, \ldots, m\}$. According to the study in [18], we set $\gamma = 1$, $m = 4$, and $\alpha_j \in \{1.0, 1.3, 1.7, 2.0\}$.

The next question is how to set and adapt the probabilities $\{\psi_j|j = 1, 2, \ldots, m\}$, with the purpose of choosing a suitable Lévy distribution from the candidate set $\{L(\alpha_j, \gamma)|j = 1, 2, \ldots, m\}$. Initially, it is reasonable to set $\psi_j = \frac{1}{m}$, $j = 1, 2, \ldots, m$, because we have no clue which one is better. But after a number of generations we should be able to adjust them adaptively based on the performance of each $L(\alpha_j, \gamma)$.

Assuming the individual $\boldsymbol{x}_{i,G+1}$ is the offspring of $\boldsymbol{x}_{i,G}$ at the generation G, the fitness improvement between them is denoted as $\Delta f_{i,G}$, and all fitness improvements obtained by $L(\alpha_j, \gamma)$ are accumulated in $\Delta F_{j,G}$. We introduce fitness improvement memories to store these numbers within a fixed number of previous generations hereby named learning period (LP). After the initial LP generations, the probabilities of choosing different Lévy distributions will be updated at each subsequent generation based on the recorded fitness improvement memories. For example, at the generation G, the probability of choosing $L(\alpha_j, \gamma)$ is updated by

$$\psi_j = \frac{S_{j,G}}{\sum_{k=1}^{m} S_{k,G}}, \quad j = 1, \ldots, m, \tag{32}$$

where

$$S_{j,G} = \sum_{g=G-LP}^{G-1} \frac{\Delta F_{j,g}}{\max_{k=1}^{NP} \Delta f_{k,g} - \min_{k=1}^{NP} \Delta f_{k,g} + \epsilon}, \tag{33}$$

where $S_{j,G}$, $j = 1, \ldots, m$ represents the normalized fitness improvement obtained by using $L(\alpha_j, \gamma)$ to generate F_i over the previous LP generations. The small constant value $\epsilon = 0.01$ is used to avoid possible null fitness improvement. Through such an adaptation strategy, the larger the fitness improvement for $L(\alpha_j, \gamma)$ within the previous LP generations, the larger the probability of applying it to generate F_i at the current generation. LP can be set to 50 as suggested in SaDE[7].

4.3 LDE: Proposed Algorithm

In Subsections 4.1 and 4.2 we have proposed an adaptive mutation scheme and a parameter control strategy for the scale factor F using the Lévy distribution, respectively. They are the two key components to formulate an effective DE variant. As for another parameter, i.e., crossover rate CR, we will not use any sophisticated adaptation methods because it was found that the performance of DE is superior and stable with empirical values around $CR = 0.1$ or $CR = 0.9$. The following very simple method will be used to generate $CR_{i,G}$ for the crossover of each individual in the generation G:

$$CR_{i,G} = \begin{cases} CR_{i,G-1}, & \text{if } f(\boldsymbol{u}_{i,G-1}) \leqslant f(\boldsymbol{x}_{i,G-1}), \\ 0.1, & \text{if } U_i(0,1) < 0.5, \\ 0.9, & \text{otherwise}, \end{cases} \tag{34}$$

where $U_i(0,1)$ stands for a uniform random number in $[0,1]$, and $CR_{i,0} = 0.9$ is used as the initial setting.

With the new adaptive mutation scheme "DE/rand-to-$p$best/2" and adaptive parameter control strategy, we have proposed a new DE variant. Since the introducing of Lévy distribution is one of the main contributions, the new algorithm is denoted as Lévy DE (LDE) in this paper. The complete procedure of LDE is summarized as follows.

1) Generate the initial population of NP individuals, i.e., $\{\boldsymbol{x}_i | 1 \leqslant i \leqslant NP\}$, and set $G = 1$.

2) Set $\alpha_j \in \{1.0, 1.3, 1.7, 2.0\}$, $\gamma = 1.0$ for $L(\alpha_j, \gamma)$, and $\psi_j = 0.25$, $j = 1, 2, 3, 4$.

3) Evaluate the fitness for each individual $\boldsymbol{x}_i$ of the population based on the objective function, $f(\boldsymbol{x}_i)$.

4) Calculate the parameter p according to (30).

5) Generate the set S^p for $\tilde{\boldsymbol{x}}_{p\,\text{best}}$ members based on Algorithm 1.

6) if $G > LP$, update ψ_j, $j = 1, 2, 3, 4$ based on (32) and (33).

7) For each individual $\boldsymbol{x}_i$, produce F_i with (31), randomly select $\tilde{\boldsymbol{x}}_{p\,\text{best}}$ from S^p, and then generate a mutated individual $\boldsymbol{v}_i$ using "DE/rand-to-$p$best/2" as given in (27).

8) For each pair $(\boldsymbol{x}_i, \boldsymbol{v}_i)$, produce $CR_{i,G}$ with (34), and then generate a trial individual $\boldsymbol{u}_i$ by:

$$\boldsymbol{u}_i(j) = \begin{cases} \boldsymbol{v}_i(j), & \text{if } U_j(0,1) \leqslant CR_{i,G} \text{ or } j = j_r, \\ \boldsymbol{x}_i(j), & \text{otherwise}, \end{cases} \tag{35}$$

where $U_j(0,1)$ stands for a uniform random number between 0 and 1, and j_r is a randomly chosen index in $[1, NP]$ to ensure that the trial vector $\boldsymbol{u}_i$ does not duplicate $\boldsymbol{x}_i$.

9) For each pair $(\boldsymbol{x}_i, \boldsymbol{u}_i)$, replace $\boldsymbol{x}_i$ with $\boldsymbol{u}_i$ and record the fitness improvement if $f(\boldsymbol{u}_i) \leqslant f(\boldsymbol{x}_i)$.

10) Stop if the halting criterion is satisfied; otherwise, $G = G + 1$, and go to step 4.

5 Experimental Studies

5.1 Benchmark Functions

A test suite with 28 benchmark functions was used in our experimental studies. The first 23 functions are from a classical test suite, which has been widely used to evaluate and analyze EAs' performance[16-17,28]. The other functions were selected from a new benchmark set developed for the special session on real-parameter optimization[29] in the 2005 IEEE Congress on Evolutionary Computation (CEC2005).

The definitions of the classical 23 functions are given in Table 1. Functions $f_1 \sim f_{13}$ are scalable problems, while functions $f_{14} \sim f_{23}$ are fixed low-dimensional problems. The 23 functions can be classified into three categories based on different problem characteristics. Functions $f_1 \sim f_6$ are unimodal. Function f_5 is the step function, which has one optimum and is discontinuous. Function f_6 is a noisy quartic function, where $random[0,1)$ is a uniformly distributed random variable in interval $[0,1)$. Functions $f_7 \sim f_{13}$ are multimodal functions where the number of local optima increases exponentially with the problem dimension[28]. Functions $f_{14} \sim f_{23}$ are low-dimensional functions which have only a few local optima. In practice, the second category, i.e., multimodal functions with many optima, appears to be the most difficult class of problems for many optimization algorithms. A more detailed description of each benchmark function can be found in the Appendix of [17].

Although the CEC2005 test suite includes 25 functions with different complexities, only the first 14 functions are resoluble with existing methods. All the 14

Table 1. Classical 23 Functions

Function	$\mathbb{S}$	$f_{\min}$
$f_1(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]^n$	0
$f_2(\boldsymbol{x}) = \sum_{i=1}^{n} \lvert x_i\rvert + \Pi_{i=1}^{n} \lvert x_i\rvert$	$[-10, 10]^n$	0
$f_3(\boldsymbol{x}) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	$[-100, 100]^n$	0
$f_4(\boldsymbol{x}) = \max_i\{\lvert x_i\rvert, 1 \leqslant i \leqslant n\}$	$[-100, 100]^n$	0
$f_5(\boldsymbol{x}) = \sum_{i=1}^{n} (\lfloor x_i + 0.5\rfloor)^2$	$[-100, 100]^n$	0
$f_6(\boldsymbol{x}) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$	$[-1.28, 1.28]^n$	0
$f_7(\boldsymbol{x}) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-30, 30]^n$	0
$f_8(\boldsymbol{x}) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i\rvert})$	$[-500, 500]^n$	$\approx -418.982\,887 \times n$
$f_9(\boldsymbol{x}) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	$[-5.12, 5.12]^n$	0
$f_{10}(\boldsymbol{x}) = -20\exp\Big(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\Big) - \exp\Big(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\Big) + 20 + e$	$[-32, 32]^n$	0
$f_{11}(\boldsymbol{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \Pi_{i=1}^{n}\cos\Big(\frac{x_i}{\sqrt{i}}\Big) + 1$	$[-600, 600]^n$	0
$f_{12}(\boldsymbol{x}) = \frac{\pi}{n}\Big[10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2(1 + 10\sin^2(\pi y_{i+1})) + (y_n - 1)^2\Big] + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$, $y_i = 1 + \frac{1}{4}(x_i + 1)$, $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & \text{if } x_i > a, \\ 0, & \text{if } -a \leqslant x_i \leqslant a, \\ k(-x_i - a)^m, & \text{if } x_i < -a \end{cases}$	$[-50, 50]^n$	0
$f_{13}(\boldsymbol{x}) = 0.1[\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2(1 + \sin^2(3\pi x_{i+1})) + (x_n - 1)(1 + \sin^2(2\pi x_n))] + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	$[-50, 50]^n$	0
$f_{14}(\boldsymbol{x}) = \Big[\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\Big]^{-1}$	$[-65.536, 65.536]^2$	$\approx 0.998\,004$
$f_{15}(\boldsymbol{x}) = \sum_{i=1}^{11}[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}]$	$[-5, 5]^4$	$\approx 0.000\,307\,486$
$f_{16}(\boldsymbol{x}) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	$[-5, 5]^2$	$\approx -1.031\,63$
$f_{17}(\boldsymbol{x}) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos x_1 + 10$	$[-5, 10] \times [0, 15]$	$\approx 0.397\,887$
$f_{18}(\boldsymbol{x}) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	$[-2, 2]^2$	≈ 3
$f_{19}(\boldsymbol{x}) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2)$	$[0, 1]^3$	$\approx -3.862\,78$
$f_{20}(\boldsymbol{x}) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2)$	$[0, 1]^6$	$\approx -3.322\,37$
$f_{21}(\boldsymbol{x}) = -\sum_{i=1}^{5}[(\boldsymbol{x} - a_i)(\boldsymbol{x} - a_i)^{\mathrm{T}} + c_i]^{-1}$	$[0, 10]^4$	$\approx -10.153\,2$
$f_{22}(\boldsymbol{x}) = -\sum_{i=1}^{7}[(\boldsymbol{x} - a_i)(\boldsymbol{x} - a_i)^{\mathrm{T}} + c_i]^{-1}$	$[0, 10]^4$	$\approx -10.402\,9$
$f_{23}(\boldsymbol{x}) = -\sum_{i=1}^{10}[(\boldsymbol{x} - a_i)(\boldsymbol{x} - a_i)^{\mathrm{T}} + c_i]^{-1}$	$[0, 10]^4$	$\approx -10.536\,4$

Note: n is the problem dimension, $f_{\min}$ is the global optimum, and $\mathbb{S} \subseteq \mathbb{R}^n$ is the search space. The parameters of f_{14}, f_{15} and $f_{19} \sim f_{23}$ can be found in the Appendix of [17].

functions are scalable, and many of them are the shifted, rotated variants of the classical functions in order to make them more resistant to simple search tricks[30]. Since a single "shift" operation has little influence on the performance of DE variants, we consider only the five shifted and rotated functions in this test suite which are listed as $f_{24} \sim f_{28}$ in Table 2.

5.2 Experimental Setting and Results

The performance of LDE was compared with that of the classical DE, three recent DE variants, i.e., JADE, jDE and SaNSDE, and two EP variants FEP and LEP. The problem dimension was set to 30 for all the scalable functions, i.e., $f_1 \sim f_{13}$ and $f_{24} \sim f_{28}$. Error value, which is the difference between the current function value and the optimum, was used to indicate the quality of a solution. For fair comparison, all the algorithms used the same number of maximum fitness evaluations (*MaxFEs*), i.e., 150 000 for f_1, f_2, f_5, f_6, and $f_8 \sim f_{13}$, 300 000 for f_3, f_4 and $f_{24} \sim f_{28}$, 900 000 for f_7, and 20 000 for low-dimensional functions $f_{14} \sim f_{23}$.

Table 2. Shifted and Rotated Functions Used in the Experimental Studies

Function	$\mathbb{S}$	$f_{\min}$
$f_{24}(\boldsymbol{x}) = \sum_{i=1}^{D}(10^6)^{\frac{i-1}{D-1}} z_i^2 - 450$	$[-100, 100]^n$	-450
$f_{25}(\boldsymbol{x}) = \sum_{i=1}^{D} \frac{z_i^2}{4000} - \Pi_{i=1}^{D} \cos(\frac{z_i}{\sqrt{i}}) + 1 - 180$	$[-600, 600]^n$	-180
$f_{26}(\boldsymbol{x}) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} z_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi z_i)) + 20 + e - 140$	$[-32, 32]^n$	-140
$f_{27}(\boldsymbol{x}) = \sum_{i=1}^{D}(z_i^2 - 10\cos(2\pi z_i) + 10) - 330$	$[-5, 5]^n$	-330
$f_{28}(\boldsymbol{x}) = \sum_{i=1}^{D}(\sum_{k=0}^{20}[a^k \cos(2\pi b^k(z_i + 0.5))]) - D\sum_{k=0}^{20}(a^k \cos(2\pi b^k 0.5)) + 90$	$[-0.5, 0.5]^n$	90

Note: $z = (\boldsymbol{x} - \boldsymbol{o}) \times \boldsymbol{M}$ for each of the functions, where the shift vector $\boldsymbol{o}$ and rotation matrix $\boldsymbol{M}$ are given in [29], n is the dimension, $f_{\min}$ is the global optimum, and $\mathbb{S} \subseteq \mathbb{R}^n$ is the search space.

Due to the representation limitation of floating point numbers, the quality of the obtained final solution when *MaxFEs* are used up may not be distinguishable when all algorithms have found very high-precision solutions. So we introduced the Expected Running Time (*ERT*) as another performance comparison metric to complement the final solution quality indicator. According to [31], the definition of *ERT* can be described as follows. Given a Value to Reach (*VTR*), a run of an optimizer is considered to be successful if and only if the optimizer has discovered a candidate solution for which the objective function takes on a value below or equal to *VTR*. The Function Evaluations to Success (*FES*) is further defined to be the number of function evaluations (*FEs*) spent in one specific successful run until such a candidate solution is discovered. Let $\overline{FES}$ be the arithmetic mean over all *FES*s of all successful runs according to one objective function and *VTR*. *ERT* is defined as follows.

$$ERT = \frac{SR \times \overline{FES} + (1 - SR) \cdot MaxFEs}{SR}, \quad (36)$$

where SR stands for success rate:

$$SR = \frac{\text{Number of successful runs}}{\text{Number of total runs}}. \quad (37)$$

ERT will be given as "-" when $SR = 0$ in the following. To be consistent with previous work[31], *VTR* was set to 1.0e−08 for all benchmark functions except f_6, which is a noise function and $VTR = 1.0\text{e} - 02$ was used.

The parameters of LDE were set exactly the same as the description given in Section 4. The main parameters are $\alpha_j \in \{1.0, 1.3, 1.7, 2.0\}$, $\gamma = 1.0$ for Lévy distribution, $p_l = 0.05$, $p_u = 0.5$ for the lower and upper bounds of parameter p, and $LP = 50$ for the *learning period*. As for other algorithms, we followed the parameter settings in their original papers. The experimental results over 30 independent runs are summarized in Tables 3∼6, where the best result in each row is given in boldface. The evolution curves of the compared algorithms on some selected functions are shown in Figs. 1 and 2.

For unimodal functions $f_1 \sim f_6$, it can be found that LDE and JADE are the top two among all the compared algorithms. Their results are better than those of others on all the functions except for the simple step function f_5, on which all the algorithms found exactly the same result. The results of all DE variants, including the classical DE, are significantly better than those of EP variants on five out of the six functions. That may be one of the reasons that DE is very popular for numerical optimization. On the comparison between LDE and JADE, there is no clear winner because JADE is better on f_1, f_3 and f_6, while LDE is better on f_2 and f_4. JADE performed well because it is quite greedy, which is very important in solving unimodal problems, by utilizing the bias of the top 5% individuals for mutation. However, too much greedy is not good for multimodal problems. The main strategies used in LDE are to control the degree of greedy without affecting its efficiency. The results on unimodal functions confirm that these strategies did not introduce any serious side effect since the performance of LDE is still at least comparable to state-of-the-art DE variants.

For the multimodal functions $f_7 \sim f_{13}$, LDE is the clear winner in terms of overall performance. Compared with all other algorithms, LDE performed better or at least the same on six out of the seven functions for the final solution quality. For the generalized Rosenbrock's function f_7, although SaNSDE achieved the best result, LDE still performed significantly better than other five algorithms. Robustness is very important in solving this function because some of its local optima are very deceptive. The "Mean" and "Std Dev" values in Table 4 indicate that LDE can find the global optimum of f_7 consistently over the tested 30 independent runs. The *ERT* metric indicate that LDE converged very fast to *VTR* of 1.0e−08 on f_7. Compared with JADE, LDE was outperformed only on f_{10} and f_{13} by ERT, and the superiority is very slight. In contrast, LDE outperformed JADE significantly on f_7, for which JADE performed quite worse because it was trapped in the

Table 3. Experimental Results on the 30-D Unimodal Functions $f_1 \sim f_6$

		LDE	JADE	jDE	SaNSDE	DE	FEP	LEP
f_1	Mean	2.28e−53	**7.03e−62**†	1.68e−29	4.07e−23	1.77e−13	5.7e−04	6.3e−04
	Std Dev	2.27e−53	2.07e−61	1.78e−29	6.53e−23	1.35e−13	1.3e−04	7.6e−05
	ERT	33 407	**29 353**	56 643	73 547	108 500	–	–
f_2	Mean	**1.99e−27**†	1.29e−23	1.04e−17	4.03e−11	7.43e−07	8.1e−03	1.3e−03
	Std Dev	8.75e−28	3.92e−23	6.87e−18	4.87e−11	4.04e−07	7.7e−04	9.1e−04
	ERT	**50 933**	51 523	80 000	119 837	–	–	–
f_3	Mean	4.33e−19	**3.18e−34**†	1.97e−07	2.57e−15	9.22e−05	1.6e−02	4.2e−02
	Std Dev	7.39e−19	1.64e−33	3.29e−07	8.21e−15	8.36e−05	1.4e−02	6.0e−02
	ERT	155 073	**94 250**	1 791 780	198 783	–	–	–
f_4	Mean	**9.75e−20**†	1.18e−14	4.67e−01	1.14e−05	4.80e−02	3.0e−01	1.4e−02
	Std Dev	2.14e−19	2.99e−14	7.50e−01	1.99e−05	1.23e−01	5.0e−01	6.8e−02
	ERT	**135 217**	166 650	–	–	–	–	–
f_5	Mean	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	Std Dev	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
	ERT	**10 843**	11 040	21 050	28 557	40 623	–	–
f_6	Mean	1.84e−03	**1.18e−03**†	6.31e−03	7.23e−03	9.31e−03	7.6e−03	7.4e−03
	Std Dev	5.94e−04	4.76e−04	1.64e−03	2.22e−03	2.86e−03	2.6e−02	3.9e−03
	ERT	27 920	**26 513**	99 627	131 308	295 864	–	–

Note: "Mean" indicates the average of the obtained minimum error values in 30 independent runs, "Std Dev" stands for the corresponding standard deviation, "ERT" is the expected running time calculated based on (36), and "†" denotes that the result is significantly better than all the others in the same row by Wilcoxon signed-rank test with $p < 0.05$.

Table 4. Experimental Results on the 30-D Multimodal Functions $f_7 \sim f_{13}$

		LDE	JADE	jDE	SaNSDE	DE	FEP	LEP
f_7	Mean	4.12e−28	1.33e−01	1.33e−01	**6.41e−30**	1.33e−01	5.1e+00	4.3e+01
	Std Dev	1.90e−27	7.28e−01	7.28e−01	1.02e−29	7.28e−01	5.9e+00	3.2e+01
	ERT	**237 753**	276 797	615 279	268 837	456 248	–	–
f_8	Mean	**0.00e+00**	3.95e+00	**0.00e+00**	3.64e−13	1.54e+03	1.1e+03	1.1e+03
	Std Dev	0.00e+00	2.16e+01	0.00e+00	8.81e−13	1.77e+03	3.1e+02	5.8e+00
	ERT	**74 520**	130 479	88 223	121 227	–	–	–
f_9	Mean	**0.00e+00**†	1.73e−11	1.18e−16	2.34e−05	1.78e+02	2.4e+01	5.9e+00
	Std Dev	0.00e+00	1.02e−11	6.49e−16	3.24e−05	1.24e+01	1.2e−02	4.7e+00
	ERT	**109 593**	129 503	114 977	–	–	–	–
f_{10}	Mean	**4.44e−15**	**4.44e−15**	7.28e−15	2.33e−12	1.40e−07	8.9e−02	1.9e−02
	Std Dev	0.00e+00	0.00e+00	1.45e−15	1.70e−12	5.25e−08	9.1e−03	1.0e−03
	ERT	51 227	**45 373**	87 230	111 953	–	–	–
f_{11}	Mean	**0.00e+00**	2.47e−04	**0.00e+00**	**0.00e+00**	5.58e−13	1.6e−02	2.4e−02
	Std Dev	0.00e+00	1.35e−03	0.00e+00	0.00e+00	5.04e−13	2.2e−02	2.8e−02
	ERT	**37 917**	41 314	60 363	78 453	112 057	–	–
f_{12}	Mean	**1.57e−32**	**1.57e−32**	7.14e−31	3.52e−22	1.72e−14	9.2e−06	6.0e−06
	Std Dev	5.57e−48	5.57e−48	6.22e−31	1.50e−21	1.07e−14	3.6e−06	1.0e−06
	ERT	**28 867**	29 747	50 987	69 890	100 323	–	–
f_{13}	Mean	**1.35e−32**	**1.35e−32**	3.83e−30	2.50e−23	1.61e−13	1.6e−04	9.8e−05
	Std Dev	5.57e−48	5.57e−48	3.88e−30	3.22e−23	1.15e−13	7.3e−05	1.2e−05
	ERT	31 520	**30 030**	54 873	72 487	107 703	–	–

Note: "Mean" indicates the average of the obtained minimum error values in 30 independent runs, "Std Dev" stands for the corresponding standard deviation, "ERT" is the expected running time calculated based on (36), and "†" denotes that the result is significant better than all others in the same row by Wilcoxon signed-rank test with $p < 0.05$.

Table 5. Experimental Results on the Low-Dimensional Multimodal Functions $f_{14} \sim f_{23}$

		LDE	JADE	jDE	SaNSDE	DE
f_{14}	Mean	**1.70e−16**	2.22e−16	2.15e−16	2.15e−16	2.15e−16
	Std Dev	9.55e−17	0.00e+00	4.05e−17	4.05e−17	4.05e−17
	ERT	4 873	**4 257**	4 743	4 887	5 030
f_{15}	Mean	2.34e−12	7.92e−12	1.23e−08	1.07e−04	**1.28e−13**
	Std Dev	8.87e−12	4.17e−11	2.89e−08	9.46e−05	3.89e−13
	ERT	14 653	14 697	22 667	–	**12 680**
f_{16}	Mean	**0.00e+00**	4.44e−17	**0.00e+00**	7.48e−15	1.48e−17
	Std Dev	0.00e+00	9.03e−17	0.00e+00	2.45e−14	5.63e−17
	ERT	**5 697**	5 717	5 803	9 493	5 797
f_{17}	Mean	2.84e−14	**0.00e+00**	**0.00e+00**	4.80e−09	**0.00e+00**
	Std Dev	1.56e−13	0.00e+00	0.00e+00	2.22e−08	0.00e+00
	ERT	7 883	7 607	7 370	14 739	**6 787**
f_{18}	Mean	2.62e−15	**1.92e−15**	2.47e−15	2.86e−15	2.26e−15
	Std Dev	2.43e−16	1.05e−15	5.30e−16	1.01e−15	6.84e−16
	ERT	5 223	**4 563**	4 760	8 533	4 623
f_{19}	Mean	**0.00e+00**	**0.00e+00**	**0.00e+00**	7.40e−17	**0.00e+00**
	Std Dev	0.00e+00	0.00e+00	0.00e+00	1.68e−16	0.00e+00
	ERT	**4 440**	4 543	5 067	7 467	4 927
f_{20}	Mean	3.96e−03	1.59e−02	7.93e−03	**1.01e−04**	3.17e−02
	Std Dev	2.17e−02	4.11e−02	3.02e−02	1.66e−04	5.35e−02
	ERT	**11 807**	19 295	14 389	53 064	21 705
f_{21}	Mean	**1.36e−15**	1.68e−01	2.76e−12	6.22e−03	1.95e−15
	Std Dev	7.64e−16	9.22e−01	1.51e−11	2.21e−02	5.42e−16
	ERT	**11 227**	13 848	12 593	600 000	12 263
f_{22}	Mean	**1.60e−15**	1.89e−15	2.25e−15	2.10e−03	1.89e−15
	Std Dev	5.42e−16	7.99e−16	9.25e−16	7.19e−03	4.51e−16
	ERT	**10 407**	10 863	11 437	84 357	11 537
f_{23}	Mean	**1.07e−15**	2.07e−15	5.92e−15	2.72e−04	1.66e−15
	Std Dev	8.85e−16	1.41e−15	2.14e−14	9.12e−04	4.51e−16
	ERT	**10 477**	11 133	11 677	149 300	11 337

Note: "Mean" indicates the average of the obtained minimum error values in 30 independent runs, "Std Dev" stands for the corresponding standard deviation, "ERT" is the expected running time calculated based on (36), and "†" denotes that the result is significant better than all others in the same row by Wilcoxon signed-rank test with $p < 0.05$.

Table 6. Experimental Results on the Shifted and Rotated Functions $f_{24} \sim f_{28}$

		LDE	JADE	jDE	SaNSDE	DE
f_{24}	Mean	**1.71e+04**	2.22e+04	2.03e+05	7.37e+04	7.20e+05
	Std Dev	6.26e+03	2.39e+04	6.82e+04	5.04e+04	3.08e+05
	ERT	–	–	–	–	–
f_{25}	Mean	**0.00e+00**†	1.07e−02	2.84e−14	9.47e−16	1.04e−14
	Std Dev	0.00e+00	5.72e−03	0.00e+00	5.19e−15	1.39e−14
	ERT	**60 547**	4 276 550	85 647	81 627	98 210
f_{26}	Mean	2.09e+01	2.09e+01	2.09e+01	2.09e+01	2.09e+01
	Std Dev	4.79e−02	2.44e−01	6.82e−02	4.97e−02	5.49e−02
	ERT	–	–	–	–	–
f_{27}	Mean	3.98e+01	**2.75e+01**†	4.98e+01	5.07e+01	1.78e+02
	Std Dev	8.77e+00	5.27e+00	8.88e+00	8.16e+00	8.09e+00
	ERT	–	–	–	–	–
f_{28}	Mean	**1.63e+01**†	2.45e+01	2.57e+01	2.71e+01	3.95e+01
	Std Dev	6.85e+00	1.82e+00	4.41e+00	1.43e+00	1.01e+00
	ERT	–	–	–	–	–

Note: "Mean" indicates the average of the obtained minimum error values over 30 independent runs, "Std Dev" stands for the corresponding standard deviation, "ERT" is the expected running time calculated based on (36), and "†" denotes that the result is significant better than all others in the same row by Wilcoxon signed-rank test with $p < 0.05$.

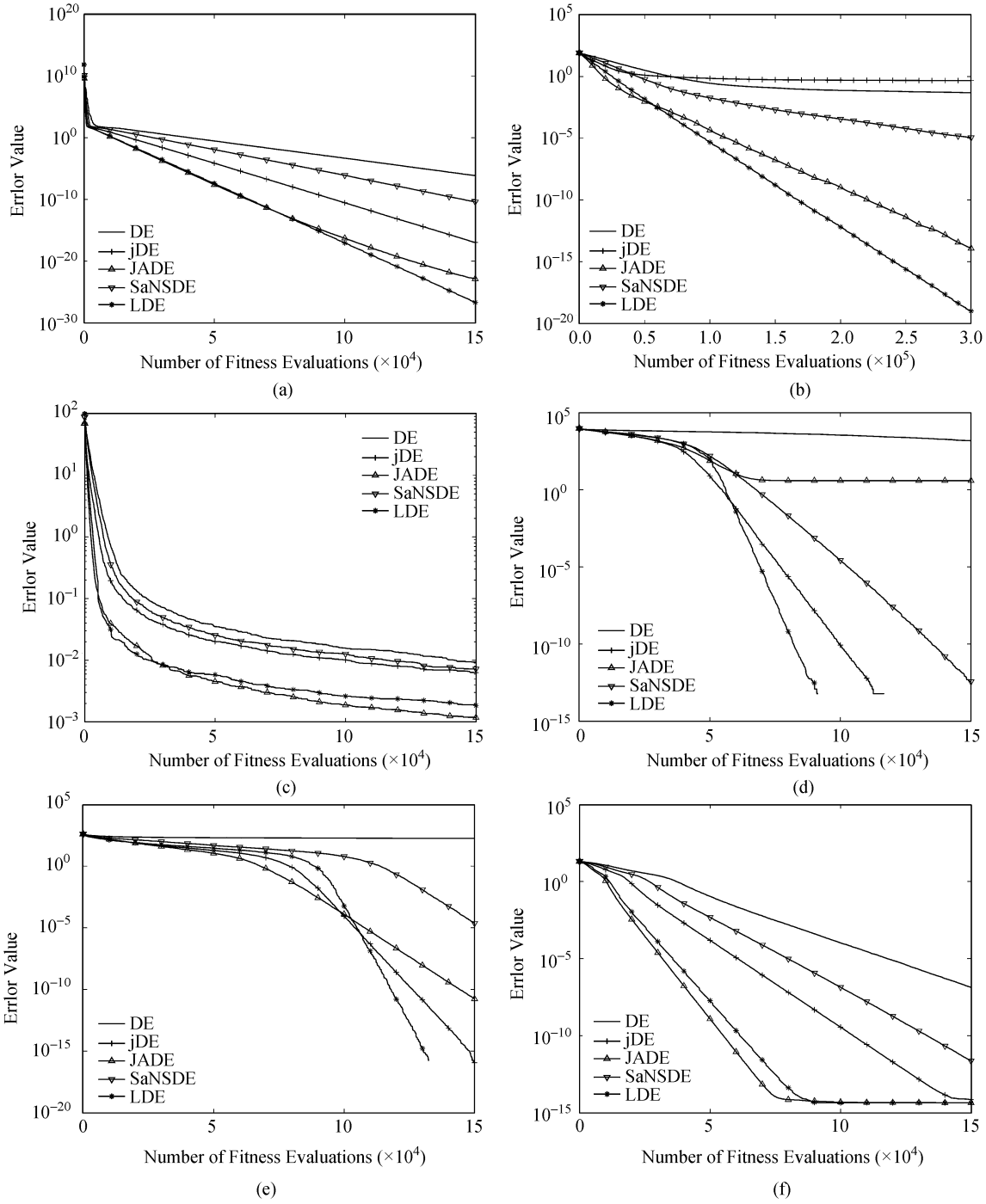

Fig.1. Evolution curves of the compared algorithms on 30-D functions f_2, f_4, f_6, f_8, f_9, and f_{10}. (a) f_2. (b) f_4. (c) f_6. (d) f_8. (e) f_9. (f) f_{10}.

high deceptive local optimum several times during the 30 independent runs. For jDE, DE, FEP and LEP, they failed to find comparable results to that of LDE on most of the functions except for f_8 and f_{11}. The class of multimodal functions with many local optima are often regarded as being difficult to optimize because they have many local optima, and their fitness landscapes appear to be very "rugged". The good performance of LDE on this class of functions indicates that the proposed adaptive mutation scheme and parameter control strategies worked well as expected.

Table 5 shows the results for low-dimensional multimodal functions $f_{14} \sim f_{23}$. The results of FEP and LEP were omitted because their performance was not

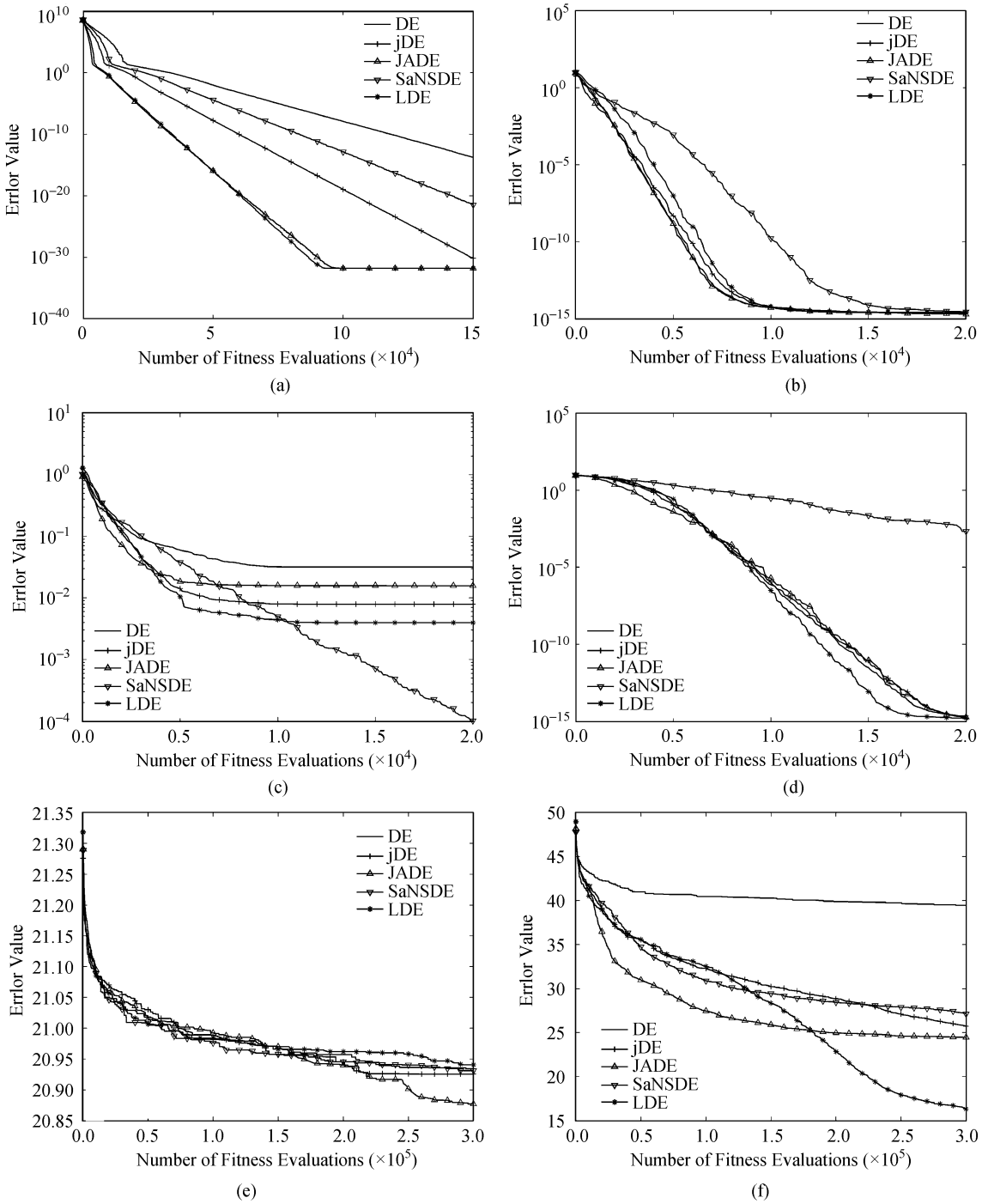

Fig.2. Evolution curves of the compared algorithms on 30-D functions f_{12}, f_{18}, f_{20}, f_{22}, f_{26}, and f_{28}. (a) f_{12}. (b) f_{18}. (c) f_{20}. (d) f_{22}. (e) f_{26}. (d) f_{28}.

comparable to that of DE variants. It can be found that the performance of all the compared algorithms is quite close to each other on most of the tested functions, and the obtained results are also very close to the global optima. For the overall performance, LEP performed well on five out of the ten functions by both the final solution quality and *ERT* values. It is very interesting to note that even the simple classical DE found better results on functions f_{15} and f_{17} than all other adaptive or self-adaptive DE variants. The reason behind the observation can be explained in the following aspects. Firstly, since most of these functions are very easy to solve, it

is hard to say whether advanced adaptive strategies are really needed. And, the used computation effort for solving these functions is very little, i.e., 20 000 in 200 generations, which is usually not enough for any adaptive strategy to learn, and thus affects its efficacy. The observation on the results of low-dimensional functions is consistent with the intuition that adaptive strategies are more helpful in dealing with difficult problems than simple ones.

Table 6 shows the results for shifted and rotated functions $f_{24} \sim f_{28}$. The first impression is that all the compared algorithms performed quite badly on most of these functions. Only one function, i.e., f_{25}, in the five functions was solved successfully. For the detailed results, LDE performed better than all other algorithms on functions f_{24}, f_{25} and f_{28}, and was outperformed by JADE on function f_{27}. All the compared algorithms obtained similar results on function f_{26}. The results in this part show that the compared algorithms may have difficulty in solving rotated benchmark functions. This is also a common problem for many other EAs[32]. It would be very interesting to study how the rotation affects the fitness landscape of a benchmark function and how it is linked to real-world applications. The analysis of EAs' behaviors on rotated functions is one of the focuses of our future work.

5.3 Efficacy of Adaptive Mutation Scheme "DE/rand-to-*p*best/2"

The mutation scheme "DE/rand-to-*p*best/2", which utilizes an adaptive strategy to set its parameter p based on the identified local fitness landscape, is one of the main components underpinning the proposed LDE. To verify its efficacy, we further formulated the following three LDE variants to conduct experimental comparison:

• LDE$^{p=0.05}$, which is the same as LDE except that the parameter p is set to 0.05 constantly.

• LDE$^{p=0.5}$, which is the same as LDE except that the parameter p is set to 0.5 constantly.

• LDE^{r}, which is the same as LDE except that the parameter p is determined by:

$$p = p_l + (p_u - p_l) \times (1 - \varphi), \tag{38}$$

with $p_l = 0.05$ and $p_u = 0.5$ as the lower and upper bounds, respectively.

Since the range of p is $[0.05, 0.5]$, LDE$^{p=0.05}$ and LDE$^{p=0.5}$ are just on the two extreme points. Compared with LDE, LDE^{r} uses a reversed strategy to adjust the parameter p. Summarized in Table 7 are the simulation results of LDE and its three variants, where the best result in each row is given in boldface. Compared with LDE$^{p=0.05}$, LDE performed better on multimodal functions, but was outperformed on unimodal functions. Since LDE$^{p=0.05}$ puts extreme emphasis on the bias of top individuals, it is natural get better results on unimodal functions. However, for difficult multimodal functions such a greedy strategy is harmful because it may increase the risk of the optimizer to be trapped in local optima. Compared with LDE$^{p=0.5}$ and LDE^{r}, LDE is observed better performance on both unimodal functions and multimodal functions. The results suggest that it is useful to introduce the bias of superior individuals in mutation, and the bias should be adaptively controlled in an appropriate manner. For some low-dimensional multimodal functions in the set of $f_{14} \sim f_{23}$, it is interesting to find that there is little difference among all the compared variants. The main reason is that these problems are quite easy to solve, and the bias of superior individuals is not so important because all the global optima were already located within only 200 generations. The performance difference of the compared algorithms is also minor for most of the rotated functions. It might be because the obtained solutions are quite far away from the global optimum, and thus it is difficult for LDE to make use of the bias of superior solutions.

Table 7. Experimental Comparison Among LDE Variants with Different p Settings, Averaged over 30 Independent Runs

		LDE	LDE$^{p=0.05}$	LDE$^{p=0.5}$	LDE^{r}
f_1	ERT	33 407	**31 970**	42 650	39 340
	Mean (Std Dev)	2.28e−53 (2.27e−53)	7.69e−54 (1.08e−54)$^{+}$	2.57e−39 (7.41e−39)$^{-}$	1.13e−42 (1.98e−42)$^{-}$
f_2	ERT	50 933	**47 503**	64 803	59 707
	Mean (Std Dev)	1.99e−27 (8.75e−28)	3.21e−28 (1.41e−28)$^{+}$	5.20e−22 (3.37e−22)$^{-}$	7.23e−24 (3.48e−24)$^{-}$
f_3	ERT	**155 073**	164 643	434 930	482 833
	Mean (Std Dev)	4.33e−19 (7.39e−19)	4.17e−18 (6.29e−18)$^{-}$	1.45e−08 (3.16e−08)$^{-}$	1.26e−08 (1.66e−08)$^{-}$
f_4	ERT	**135 217**	136 753	175 627	361 570
	Mean (Std Dev)	9.75e−20 (2.14e−19)	4.02e−20 (4.06e−20)$^{\approx}$	2.78e−14 (9.82e−14)$^{-}$	5.71e−07 (1.24e−06)$^{-}$
f_5	ERT	10 843	**10 707**	15 267	14 120
	Mean (Std Dev)	0.00e+00 (0.0e+00)	0.00e+00 (0.0e+00)$^{\approx}$	0.00e+00 (0.0e+00)$^{\approx}$	0.00e+00 (0.0e+00)$^{\approx}$

(to be continued)

Table 7. (continued)

		LDE	LDE$^{p=0.05}$	LDE$^{p=0.5}$	LDE^{r}
f_6	ERT	27 920	**23 120**	29 457	23 440
	Mean (Std Dev)	1.84e−03 (5.94e−04)	1.55e−03 (3.94e−04)$^{+}$	2.22e−03 (6.41e−04)$^{\approx}$	1.62e−03 (5.79e−04)$^{\approx}$
f_7	ERT	**237 753**	256 693	497 355	747 325
	Mean (Std Dev)	4.12e−28 (1.90e−27)	1.33e−01 (7.28e−01)$^{\approx}$	1.33e−01 (7.28e−01)$^{-}$	7.97e−01 (1.62e+00)$^{-}$
f_8	ERT	74 520	**74 317**	84 177	76 730
	Mean (Std Dev)	0.00e+00 (0.00e+00)	0.00e+00 (0.00e+00)$^{\approx}$	0.00e+00 (0.00e+00)$^{\approx}$	0.00e+00 (0.00e+00)$^{\approx}$
f_9	ERT	**109 593**	112 000	120 673	118 570
	Mean (Std Dev)	0.00e+00 (0.00e+00)	0.00e+00 (0.00e+00)$^{\approx}$	0.00e+00 (0.0e+00)$^{\approx}$	0.00e+00 (0.0e+00)$^{\approx}$
f_{10}	ERT	**51 227**	55 543	65 570	60 783
	Mean (Std Dev)	4.44e−15 (0.00e+00)	4.44e−15 (0.00e+00)$^{\approx}$	4.44e−15 (0.00e+00)$^{\approx}$	4.44e−15 (0.00e+00)$^{\approx}$
f_{11}	ERT	**37 917**	38 173	43 983	46 079
	Mean (Std Dev)	0.00e+00 (0.00e+00)	0.00e+00 (0.00e+00)$^{\approx}$	0.00e+00 (0.0e+00)$^{\approx}$	2.47e−04 (1.35e−03)$^{\approx}$
f_{12}	ERT	**28 867**	30 830	35 740	32 767
	Mean (Std Dev)	1.57e−32 (5.57e−48)	1.57e−32 (5.57e−48)$^{\approx}$	1.57e−32 (5.57e−48)$^{\approx}$	1.57e−32 (5.57e−48)$^{\approx}$
f_{13}	ERT	**31 520**	39 255	40 317	42 266
	Mean (Std Dev)	1.35e−32 (5.57e−48)	3.66e−04 (2.01e−03)$^{-}$	1.35e−32 (5.57e−48)$^{\approx}$	3.66e−04 (2.01e−03)$^{-}$
f_{14}	ERT	4 873	**4 303**	5 150	4 773
	Mean (Std Dev)	1.70e−16 (9.55e−17)	2.00e−16 (6.78e−17)$^{\approx}$	1.78e−16 (9.03e−17)$^{\approx}$	1.85e−16 (8.42e−17)$^{\approx}$
f_{15}	ERT	**14 653**	15 117	21 146	17 976
	Mean (Std Dev)	2.34e−12 (8.87e−12)	4.07e−12 (4.69e−12)$^{-}$	8.12e−09 (2.96e−08)$^{-}$	1.37e−09 (3.14e−09)$^{-}$
f_{16}	ERT	**5 697**	6 227	6 047	6 247
	Mean (Std Dev)	0.00e+00 (0.00e+00)	0.00e+00 (0.00e+00)$^{\approx}$	7.40e−18 (4.05e−17)$^{\approx}$	7.40e−18 (4.05e−17)$^{\approx}$
f_{17}	ERT	7 883	**7 760**	8 003	7 937
	Mean (Std Dev)	2.84e−14 (1.56e−13)	0.00e+00 (0.00e+00)$^{\approx}$	0.00e+00 (0.00e+00)$^{\approx}$	0.00e+00 (0.00e+00)$^{\approx}$
f_{18}	ERT	5 223	5 277	5 497	**5 173**
	Mean (Std Dev)	2.62e−15 (2.43e−16)	2.35e−15 (6.40e−16)$^{+}$	2.58e−15 (3.38e−16)$^{\approx}$	2.63e−15 (1.62e−16)$^{\approx}$
f_{19}	ERT	**4 440**	4 893	5 267	4 983
	Mean (Std Dev)	0.00e+00 (0.00e+00)	0.00e+00 (0.00e+00)$^{\approx}$	0.00e+00 (0.00e+00)$^{\approx}$	0.00e+00 (0.00e+00)$^{\approx}$
f_{20}	ERT	11 807	13 659	**11 610**	11 493
	Mean (Std Dev)	3.96e−03 (2.17e−02)	1.19e−02 (3.63e−02)$^{\approx}$	9.84e−15 (3.30e−14)$^{+}$	3.96e−03 (2.17e−02)$^{\approx}$
f_{21}	ERT	**11 227**	12 573	12 233	11 487
	Mean (Std Dev)	1.36e−15 (7.64e−16)	2.01e−15 (6.14e−16)$^{-}$	3.20e−15 (5.87e−15)$^{-}$	5.99e−13 (3.27e−12)$^{\approx}$
f_{22}	ERT	**10 407**	11 420	11 557	10 503
	Mean (Std Dev)	1.60e−15 (5.42e−16)	1.78e−15 (0.00e+00)$^{\approx}$	3.49e−15 (8.09e−15)$^{-}$	1.72e−15 (3.24e−16)$^{\approx}$
f_{23}	ERT	**10 477**	11 120	11 443	10 560
	Mean (Std Dev)	1.07e−15 (8.85e−16)	1.36e−15 (7.64e−16)$^{\approx}$	1.84e−15 (9.88e−16)$^{-}$	1.24e−15 (8.28e−16)$^{\approx}$
f_{24}	ERT	–	–	–	–
	Mean (Std Dev)	1.71e+04 (6.26e+03)	1.92e+04 (7.12e+04)$^{\approx}$	3.47e+04 (1.36e+04)$^{-}$	4.44e+04 (1.42e+04)$^{-}$
f_{25}	ERT	**60 547**	63 760	106 607	127 829
	Mean (Std Dev)	0.00e+00 (0.00e+00)	0.00e+00 (0.00e+00)$^{\approx}$	9.47e−16 (5.19e−15)$^{\approx}$	2.99e−02 (1.14e−01)$^{\approx}$
f_{26}	ERT	–	–	–	–
	Mean (Std Dev)	2.09e+01 (4.79e−02)	2.09e+01 (4.99e−02)$^{\approx}$	2.09e+01 (4.03e−02)$^{\approx}$	2.09e+01 (5.87e−02)$^{\approx}$
f_{27}	ERT	–	–	–	–
	Mean (Std Dev)	3.98e+01 (8.77e+00)	3.34e+01 (8.02e+00)$^{+}$	3.95e+01 (8.75e+00)$^{\approx}$	4.07e+01 (8.18e+00)$^{\approx}$
f_{28}	ERT	–	–	–	–
	Mean (Std Dev)	1.63e+01 (6.85e+00)	1.62e+01 (6.10e+00)$^{\approx}$	1.75e+01 (7.28e+00)$^{\approx}$	1.71e+01 (6.04e+00)$^{\approx}$

Note: Indicators "$^{-}$", "$^{+}$" and "$^{\approx}$" denote that the performance of the corresponding algorithm is worse than, better than, and similar to that of LDE, respectively, by the Wilcoxon signed-rank test at a 0.05 significance level.

5.4 Efficacy of Parameter Adaptation Using Lévy Distribution

To identify the benefit of parameter adaptation using Lévy distribution, which is the other main component of LDE, we further consider the following three DE variants:

• gaussianDE, which is the same as LDE except that only Lévy distribution with $\alpha = 2.0$, i.e., the Gaussian distribution, is used to control the scale factor F_i.

• cauchyDE, which is the same as LDE except that only Lévy distribution with $\alpha = 1.0$, i.e., the Cauchy distribution, is used to control the scale factor F_i.

• randomLDE, which is the same as LDE except that the parameter α of Lévy distribution is selected randomly from the candidate set $\{1.0, 1.3, 1.7, 2.0\}$, rather than based on their performance.

Summarized in Table 8 are the simulation results of LDE, gaussianDE, cauchyDE and randomLDE, where the best result in each row is given in boldface. Compared with gaussianLDE, LDE performed better on multimodal functions, but was outperformed on unimodal functions. Compared with cauchyDE and randomLDE, LDE performed better on both unimodal functions and multimodal functions. Similar to the conclusion in Subsection 5.3 for adaptive mutation, there is no much difference among all the compared algorithms on rotated functions $f_{24} \sim f_{28}$. It seems it is very difficult to perform any step size controlling technique when the optimizer keeps making little progress. Based on the comparison in the part, it can be concluded that Gaussian and Cauchy distributions for parameter control have quite different emphases. The introducing of Lévy distribution, which is more general, can improve the overall performance of DE via parameter control. However, the switch of different Lévy distributions should be controlled adaptively based on their performance, rather than randomly.

Table 8. Experimental Comparison Among LDE, gaussianDE, cauchyDE and randomLDE, Averaged over 30 Independent Runs

		LDE	gaussianDE	cauchyDE	randomLDE
f_1	ERT	33 407	**32 193**	40 710	36 560
	Mean (Std Dev)	2.28e−53 (2.27e−53)	3.85e−54 (4.16e−54)$^+$	3.52e−43 (3.25e−43)$^-$	4.09e−48 (4.23e−48)$^-$
f_2	ERT	50 933	**50 117**	64 360	58 567
	Mean (Std Dev)	1.99e−27 (8.75e−28)	8.39e−28 (3.55e−28)$^+$	2.85e−22 (1.08e−22)$^-$	9.66e−25 (3.33e−25)$^-$
f_3	ERT	**155 073**	155 470	178 773	164 363
	Mean (Std Dev)	4.33e−19 (7.39e−19)	4.09e−19 (1.01e−18)$^\approx$	3.86e−16 (5.52e−16)$^-$	3.77e−18 (4.59e−18)$^-$
f_4	ERT	**135 217**	135 300	139 830	136 700
	Mean (Std Dev)	9.75e−20 (2.14e−19)	9.15e−20 (3.65e−19)$^\approx$	1.36e−19 (3.21e−19)$^\approx$	3.08e−19 (5.42e−19)$^\approx$
f_5	ERT	**10 843**	13 087	15 427	14 160
	Mean (Std Dev)	0.00e+00 (0.0e+00)	0.00e+00 (0.0e+00)$^\approx$	0.00e+00 (0.0e+00)$^\approx$	0.00e+00 (0.0e+00)$^\approx$
f_6	ERT	27 920	**23 817**	30 897	26 827
	Mean (Std Dev)	1.84e−03 (5.94e−04)	2.07e−03 (6.17e−04)$^\approx$	2.38e−03 (8.80e−04)$^-$	2.03e−03 (8.03e−04)$^\approx$
f_7	ERT	**237 753**	325 378	382 677	330 493
	Mean (Std Dev)	4.12e−28 (1.90e−27)	3.99e−01 (1.22e+00)$^-$	5.32e−01 (1.38e+00)$^-$	3.99e−01 (1.22e+00)$^-$
f_8	ERT	**74 520**	75 087	85 177	79 717
	Mean (Std Dev)	0.00e+00 (0.00e+00)	0.00e+00 (0.00e+00)$^\approx$	0.00e+00 (0.00e+00)$^\approx$	0.00e+00 (0.00e+00)$^\approx$
f_9	ERT	**109 593**	118 760	126 940	115 697
	Mean (Std Dev)	0.00e+00 (0.00e+00)	3.32e−02 (1.83e−01)$^-$	2.14e−14 (5.12e−14)$^-$	0.00e+00 (0.00e+00)$^\approx$
f_{10}	ERT	**51 227**	52 393	62 147	56 487
	Mean (Std Dev)	4.44e−15 (0.00e+00)	4.44e−15 (0.00e+00)$^\approx$	4.44e−15 (0.00e+00)$^\approx$	4.44e−15 (0.00e+00)$^\approx$
f_{11}	ERT	37 917	**35 880**	42 227	38 817
	Mean (Std Dev)	0.00e+00 (0.00e+00)	0.00e+00 (0.00e+00)$^\approx$	0.00e+00 (0.00e+00)$^\approx$	0.00e+00 (0.00e+00)$^\approx$
f_{12}	ERT	**28 867**	29 260	34 520	31 163
	Mean (Std Dev)	1.57e−32 (5.57e−48)	1.57e−32 (5.57e−48)$^\approx$	1.57e−32 (5.57e−48)$^\approx$	1.57e−32 (5.57e−48)$^\approx$
f_{13}	ERT	**31 520**	32 250	38 140	34 747
	Mean (Std Dev)	1.35e−32 (5.57e−48)	1.35e−32 (5.57e−48)$^\approx$	1.35e−32 (5.57e−48)$^\approx$	1.35e−32 (5.57e−48)$^\approx$
f_{14}	ERT	4 873	4 933	5 200	**4 733**
	Mean (Std Dev)	1.70e−16 (9.55e−17)	1.92e−16 (7.68e−17)$^\approx$	2.00e−16 (6.78e−17)$^\approx$	2.00e−16 (6.78e−17)$^\approx$
f_{15}	ERT	14 653	**14 447**	17 277	15 743
	Mean (Std Dev)	2.34e−12 (8.87e−12)	1.35e−12 (2.74e−12)$^\approx$	4.97e−10 (9.96e−10)$^-$	3.05e−11 (5.38e−11)$^-$

(to be continued)

Table 8. (continued)

		LDE	gaussianDE	cauchyDE	randomLDE
f_{16}	ERT	**5 697**	6 360	6 847	6 330
	Mean (Std Dev)	0.00e+00 (0.00e+00)	0.00e+00 (0.00e+00)$^{\approx}$	7.40e−18 (4.05e−17)$^{\approx}$	0.00e+00 (0.00e+00)$^{\approx}$
f_{17}	ERT	7 883	**7 593**	8 783	8 500
	Mean (Std Dev)	2.84e−14 (1.56e−13)	0.00e+00 (0.00e+00)$^{\approx}$	5.92e−17 (3.24e−16)$^{\approx}$	1.78e−16 (9.73e−16)$^{\approx}$
f_{18}	ERT	**5 223**	5 280	5 887	5 500
	Mean (Std Dev)	2.62e−15 (2.43e−16)	2.53e−15 (3.53e−16)$^{\approx}$	2.40e−15 (6.13e−16)$^{\approx}$	2.59e−15 (3.51e−16)$^{\approx}$
f_{19}	ERT	**4 440**	5 017	5 570	5 320
	Mean (Std Dev)	0.00e+00 (0.00e+00)	0.00e+00 (0.00e+00)$^{\approx}$	0.00e+00 (0.00e+00)$^{\approx}$	0.00e+00 (0.00e+00)$^{\approx}$
f_{20}	ERT	11 807	**10 877**	12 127	11 123
	Mean (Std Dev)	3.96e−03 (2.17e−02)	5.35e−14 (2.86e−13)$^{\approx}$	3.74e−14 (8.64e−14)$^{+}$	4.43e−13 (2.42e−12)$^{\approx}$
f_{21}	ERT	**11 227**	11 567	13 173	11 950
	Mean (Std Dev)	1.36e−15 (7.64e−16)	1.18e−15 (1.08e−15)$^{\approx}$	1.57e−14 (5.69e−14)$^{-}$	1.66e−15 (4.51e−16)$^{\approx}$
f_{22}	ERT	**10 407**	10 560	12 117	11 070
	Mean (Std Dev)	1.60e−15 (5.42e−16)	1.60e−15 (5.42e−16)$^{\approx}$	3.55e−15 (5.32e−15)$^{-}$	1.78e−15 (8.08e−16)$^{\approx}$
f_{23}	ERT	**10 477**	10 490	12 343	11 280
	Mean (Std Dev)	1.07e−15 (8.85e−16)	1.01e−15 (8.95e−16)$^{\approx}$	3.20e−15 (2.40e−15)$^{-}$	1.42e−15 (7.23e−16)$^{\approx}$
f_{24}	ERT	–	–	–	–
	Mean (Std Dev)	1.71e+04 (6.26e+03)	1.56e+04 (7.58e+03)$^{\approx}$	1.94e+04 (6.93e+03)$^{\approx}$	1.69e+04 (8.93e+03)$^{\approx}$
f_{25}	ERT	**60 547**	72 510	82 117	65 650
	Mean (Std Dev)	0.00e+00 (0.00e+00)	1.40e−02 (7.67e−02)$^{-}$	1.40e−02 (7.67e−02)$^{-}$	9.47e−16 (5.19e−15)$^{\approx}$
f_{26}	ERT	–	–	–	–
	Mean (Std Dev)	2.09e+01 (4.79e−02)	2.10e+01 (4.19e−02)$^{\approx}$	2.09e+01 (5.78e−02)$^{\approx}$	2.10e+01 (3.75e−02)$^{\approx}$
f_{27}	ERT	–	–	–	–
	Mean (Std Dev)	3.98e+01 (8.77e+00)	4.26e+01 (8.12e+00)$^{\approx}$	4.00e+01 (6.43e+00)$^{\approx}$	4.30e+01 (1.07e+01)$^{\approx}$
f_{28}	ERT	–	–	–	–
	Mean (Std Dev)	1.63e+01 (6.85e+00)	1.69e+01 (4.81e+00)$^{\approx}$	1.66e+01 (6.35e+00)$^{\approx}$	1.73e+01 (6.70e+00)$^{\approx}$

Note: Indicators "$-$", "$+$" and "$\approx$" denote the performance of the corresponding algorithm is worse than, better than, and similar to that of LDE, respectively, by the Wilcoxon signed-rank test at a 0.05 significance level.

5.5 Experimental Analysis on Scalability of LDE

The scalability of an algorithm is also an important measurement of how good and how applicable the algorithm is[33]. So we have conducted additional experiments to evaluate the scalability of LDE against the growth of problem dimensions. We selected 11 scalable functions from the 28 benchmark functions. Besides the previously tested dimension 30, the problem dimensions D of these functions were set to 50, 100 and 200, respectively. The computation time used by the compared algorithms was set to grow in the order of $O(D)$[33]. Assuming the number of maximum FEs for each 30-D benchmark function is *MaxFEs*, the corresponding maximum FEs will be set to $\frac{D}{30} \times MaxFEs$

Table 9. Scalability Comparison on Selected Functions with Dimensions from 30 to 200

		30D	50D	100D	200D
		Mean (ERT)	Mean (ERT)	Mean (ERT)	Mean (ERT)
f_3	LDE	4.33e−19 (155 073)	7.01e−11 (402 403)	2.64e−03 (–)	5.33e+01 (–)
	DE	9.22e−05 (–)	2.81e+00 (–)	2.38e+03 (–)	3.51e+04 (–)
	jDE	1.97e−07 (1 791 780)	1.77e−03 (–)	4.32e+00 (–)	3.36e+02 (–)
	JADE	**3.18e−34**† (**94 250**)	**4.29e−22**† (**232 597**)	**3.91e−11**† (**819 870**)	**5.54e−04**† (–)
	SaNSDE	2.57e−15 (198 783)	4.63e−10 (444 510)	2.54e−04 (–)	9.18e+00 (–)
f_5	LDE	**0.00e+00** (**10 843**)	**0.00e+00** (**19 447**)	**0.00e+00** (289 100)	7.61e+01 (–)
	DE	**0.00e+00** (40 623)	**0.00e+00** (62 670)	**0.00e+00** (179 023)	**1.64e+00** (**2 313 630**)
	jDE	**0.00e+00** (21 050)	**0.00e+00** (29 560)	3.33e−02 (**62 252**)	1.15e+01 (14 078 000)
	JADE	**0.00e+00** (11 040)	1.33e−01 (56 504)	7.43e+00 (–)	1.95e+02 (–)
	SaNSDE	**0.00e+00** (28 557)	**0.00e+00** (39 583)	6.67e−02 (190 332)	5.00e+00 (29 276 000)

(to be continued)

Table 9. (continued)

		30D	50D	100D	200D
		Mean (ERT)	Mean (ERT)	Mean (ERT)	Mean (ERT)
f_7	LDE	4.12e−28 (237 753)	**1.33e−01** (553 441)	**2.66e−01** (**1 807 718**)	1.20e+00 (5 981 476)
	DE	1.33e−01 (456 248)	**1.33e−01** (1 074 503)	4.33e+01 (–)	2.07e+02 (–)
	jDE	1.33e−01 (615 279)	2.66e−01 (1 214 532)	3.99e−01 (2 727 211)	1.19e+02 (178 996 700)
	JADE	1.33e−01 (**176 797**)	3.99e−01 (**510 641**)	1.20e+00 (2 762 114)	1.45e+01 (19 295 400)
	SaNSDE	**6.41e−30** (268 837)	2.66e−01 (660 682)	6.64e−01 (2 164 744)	**8.09e−01** (**12 197 779**)
f_9	LDE	**0.00e+00**† (**109 593**)	**0.00e+00** (**178 609**)	1.02e+02 (–)	1.62e+02 (–)
	DE	1.78e+02 (–)	3.30e+02 (–)	5.48e+02 (–)	1.52e+02 (–)
	jDE	1.18e−16 (114 977)	**0.00e+00** (179 210)	**0.00e+00**† (**326 823**)	1.33e−01 (760 796)
	JADE	1.73e−11 (129 503)	9.71e−12 (216 623)	3.55e−16 (391 010)	**3.69e−14**† (**758 707**)
	SaNSDE	2.34e−05 (–)	4.66e−04 (–)	2.81e+00 (–)	3.24e+01 (–)
f_{11}	LDE	**0.00e+00** (**37 917**)	9.56e−12 (83 203)	1.69e−10 (161 548)	**1.31e−14**† (477 588)
	DE	5.58e−13 (112 057)	5.86e−15 (168 423)	5.75e−04 (329 950)	3.36e−02 (1 186 719)
	jDE	**0.00e+00** (60 363)	**0.00e+00** (**81 417**)	**0.00e+00**† (**121 317**)	2.79e−03 (**409 788**)
	JADE	2.47e−04 (41 314)	1.89e−03 (106 462)	2.79e−03 (203 325)	2.10e−02 (966 347)
	SaNSDE	**0.00e+00** (78 453)	**0.00e+00** (99 850)	2.14e−03 (291 458)	1.21e−02 (1 243 238)
f_{13}	LDE	**1.35e−32** (31 520)	**1.35e−32** (60 420)	1.46e−03 (282 477)	8.32e−01 (1 338 022)
	DE	1.61e−13 (107 703)	2.04e−14 (169 737)	1.20e−01 (537 758)	5.34e+01 (–)
	jDE	3.83e−30 (54 873)	1.35e−32 (78 927)	**7.32e−04** (164 904)	**1.20e−01** (**352 121**)
	JADE	**1.35e−32** (**30 030**)	**1.35e−32** (**40 093**)	5.32e−02 (209 117)	9.62e−01 (1 024 820)
	SaNSDE	2.50e−23 (72 487)	4.96e−30 (98 910)	1.83e−03 (230 721)	4.43e−01 (793 750)
f_{24}	LDE	**1.71e+04** (–)	2.45e+05 (–)	1.30e+06 (–)	5.97e+06 (–)
	DE	7.20e+05 (–)	7.27e+06 (–)	1.08e+07 (–)	1.46e+07 (–)
	jDE	2.03e+05 (–)	1.06e+06 (–)	3.07e+06 (–)	1.11e+07 (–)
	JADE	2.22e+04 (–)	**2.31e+04**† (–)	**5.48e+05**† (–)	**4.02e+06** (–)
	SaNSDE	7.37e+04 (–)	1.78e+05 (–)	1.04e+06 (–)	5.05e+06 (–)
f_{25}	LDE	**0.00e+00**† (**60 547**)	**2.63e−14** (**99 241**)	**2.84e−14** (**195 679**)	**4.50e−14**† (**577 167**)
	DE	1.04e−14 (98 210)	2.84e−14 (186 910)	3.69e−02 (674 533)	1.29e−01 (23 962 800)
	jDE	2.84e−14 (85 647)	3.69e−14 (155 310)	1.00e−13 (299 537)	1.30e−01 (28 807 800)
	JADE	1.07e−02 (4 276 550)	3.44e−03 (258 316)	6.48e−03 (1 310 676)	4.02e−03 (2 973 447)
	SaNSDE	9.47e−16 (81 627)	2.84e−14 (132 733)	3.02e−02 (393 177)	1.12e−01 (8 553 150)
f_{26}	LDE	2.09e+01 (–)	2.11e+01 (–)	2.13e+01 (–)	2.14e+01 (–)
	DE	2.09e+01 (–)	2.11e+01 (–)	2.13e+01 (–)	2.14e+01 (–)
	jDE	2.09e+01 (–)	2.11e+01 (–)	2.13e+01 (–)	2.14e+01 (–)
	JADE	2.09e+01 (–)	2.11e+01 (–)	**2.10e+01**† (–)	**2.12e+01**† (–)
	SaNSDE	2.09e+01 (–)	2.11e+01 (–)	2.13e+01 (–)	2.14e+01 (–)
f_{27}	LDE	3.98e+01 (–)	8.87e+01 (–)	3.17e+03 (–)	1.43e+04 (–)
	DE	1.78e+02 (–)	3.57e+02 (–)	**1.90e+03** (–)	**5.82e+03** (–)
	jDE	4.98e+01 (–)	8.94e+01 (–)	2.02e+03 (–)	7.29e+03 (–)
	JADE	**2.75e+01** (–)	**5.95e+01** (–)	2.82e+03 (–)	1.27e+04 (–)
	SaNSDE	5.07e+01 (–)	1.10e+02 (–)	2.33e+03 (–)	9.68e+03 (–)
f_{28}	LDE	**1.63e+01** (–)	**4.42e+01** (–)	**1.21e+02** (–)	2.91e+02 (–)
	DE	3.95e+01 (–)	7.33e+01 (–)	1.60e+02 (–)	3.42e+02 (–)
	jDE	2.57e+01 (–)	5.23e+01 (–)	1.25e+02 (–)	**1.49e+02** (–)
	JADE	2.45e+01 (–)	5.21e+01 (–)	1.27e+02 (–)	2.95e+02 (–)
	SaNSDE	2.71e+01 (–)	5.47e+01 (–)	1.28e+02 (–)	3.06e+02 (–)

Note: "Mean" indicates the average of the obtained minimum error values over 30 independent runs, "ERT" is the expected running time calculated based on (36), "†" denotes that the result is significant better than all others in the same row by Wilcoxon signed-rank test with $p < 0.05$.

for the D-dimensional problem. The average results of 30 independent runs are summarized in Table 9, where the best result in each row is in boldface.

For the overall performance, LDE obtained better results than other algorithms on f_5 under dimensions 30, 50, 100, f_7 under dimensions 50, 100, f_9 and f_{13} under dimensions 30, 50, f_{11} under dimensions 30, 200, f_{24} under dimension 30, f_{25} under dimensions from 30 to 200, and f_{28} under dimensions 30, 50, 100 respectively. Compared with JADE, it was outperformed on f_3, because the greedy mutation scheme in JADE is very efficient for such unimodal functions. But LDE is still the second best among all the compared algorithms. JADE also obtained better results than LDE on f_{24} when the problem dimension is larger than 50. Compared with SaNSDE, LDE performed better for functions with dimensions smaller than 100 (included), but was outperformed on seven out of the 11 200-D functions. The phenomenon is very similar when compared with jDE. As for the classical DE, it has only slight advantages on functions f_5 and f_{27} with dimensions 100 and 200. From the results in this part, we can conclude that the proposed LDE is superior for problems with dimensions 30 and 50, and is comparable for 100-D problems, but may perform worse than other EAs on some 200-D problems.

6 Conclusions

In this paper we presented a new DE variant, LDE, by implementing its two main components, i.e., mutation scheme and parameter control with several novel adaptive strategies. For the first component, an adaptive mutation scheme is designed in order to appropriately maintain the balance between exploration and exploitation during the search process. In the new mutation scheme, the bias of superior individuals is taken into account when generating new solutions, and it is adaptively adjusted dynamically based on the identified local fitness landscape. For the other component, inspired by the success of Gaussian and Cauchy distributions in this respect, a more general Lévy distribution was introduced to adaptively control the scale factor F of DE. For every mutation in each generation, an F_i is generated from one of four predefined Lévy distributions. Probability parameters are introduced to determine which distribution to use in practice. Those probability parameters are adaptively updated based on the historical performance of the corresponding Lévy distributions.

To evaluate the efficacy of LDE, experimental studies were carried out on 28 widely used benchmark functions in numerical optimization. LDE was compared with the classical DE, three state-of-the-art DE variants, i.e., JADE, jDE and SaNSDE, and two EP variants, i.e., FEP and LEP. The results indicate that the overall performance of LDE is better than those six competitors. The behaviors of the two main components of LDE were analyzed with additional experiments. It was found that both the proposed adaptive mutation scheme and parameter control using Lévy distribution had contributed significantly to the overall performance of LDE. The scalability of LDE was also evaluated on some selected benchmark functions with dimensions from 30 to 200. Although LDE obtained results comparable to other state-of-the-art EAs, its performance did deteriorate when the problem scale is as large as 200 dimensions.

In recent years, large-scale optimization, which intends to tacke problems with more than 1 000 variables, has become an active research topic. It was found that the performance of most existing EAs deteriorates rapidly as the dimensionality of the search space increases[34]. So it may not be proper to apply traditional EAs to those large-scale optimization problems directly. It has been reported that divide-and-conquer strategy implemented using cooperative co-evolution appears to be a quite promising method[33]. Such an idea has been extended to produce a general framework for large-scale optimization in [30], and has been used to expand the scalability of DE[30] and particle swarm optimization (PSO)[35]. It would be a possible future research direction to scale up LDE by fitting it into such a framework.

References

[1] Storn R, Price K. Differential evolution — A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 1997, 11(4): 341-359.

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

[3] Chakraborty (ed.) U K. Advances in Differential Evolution. Springer-Verlag, 2008.

[4] Das S, Suganthan P N. Differential evolution: A survey of the state-of-the-art. *IEEE Trans. Evolutionary Computation*, 2011, 15(1): 4-31.

[5] Das S, Abraham A, Chakraborty U K, Konar A. Differential evolution using a neighborhood-based mutation operator. *IEEE Trans. Evolutionary Computation*, 2009, 13(3): 526-553.

[6] Zhang J, Sanderson A C. JADE: Adaptive differential evolution with optional external archive. *IEEE Trans. Evolutionary Computation*, 2009, 13(5): 945-958.

[7] Qin A K, Huang V L, Suganthan P N. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Trans. Evolutionary Computation*, 2009, 13(2): 398-417.

[8] Yang Z, Tang K, Yao X. Self-adaptive differential evolution with neighborhood search. In *Proc. the 2008 IEEE Congress on Evolutionary Computation*, June 2008, pp.1110-1116.

[9] Gämperle R, Müller S D, Koumoutsakos P. A Parameter study for differential evolution. In *Proc. WSEAS Interna-*

tional Conference on Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation, 2002, pp.293-298.
[10] Wang Y, C Z, Zhang Q. Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Trans. Evolutionary Computation*, 2011, 15(1): 55-66.
[11] Brest J, Greiner S, Boskovic B, Mernik M, Žumer V. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Trans. Evolutionary Computation*, 2006, 10(6): 646-657.
[12] Das S, Konar A, Chakraborty U K. Two improved differential evolution schemes for faster global search. In *Proc. the 2005 Conf. Genetic and Evolutionary Computation*, June 2005, pp.991-998.
[13] Abbass H A. The self-adaptive Pareto differential evolution algorithm. In *Proc. the 2002 IEEE Congress on Evolutionary Computation*, May 2002, pp.831-836.
[14] Salman A, Engelbrecht A, Omran M. Empirical analysis of self-adaptive differential evolution. *European Journal of Operational Research*, 2007, 183(2): 785-804.
[15] Yang Z, He J, Yao X. Making a difference to differential evolution. In *Advances in Metaheuristics for Hard Optimization*, Michalewicz, Z, Siarry P (eds.), Springer, 2008, pp.397-414.
[16] Bäck T, Schwefel H P. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1993, 1(1): 1-23.
[17] Yao X, Liu Y, Lin G. Evolutionary programming made faster. *IEEE Trans. Evolutionary Computation*, 1999, 3(2): 82-102.
[18] Lee C Y, Yao X. Evolutionary programming using mutations based on the Lévy probability distribution. *IEEE Trans. Evolutionary Computation*, 2004, 8(1): 1-13.
[19] Storn R. System design by constraint adaptation and differential evolution. *IEEE Trans. Evolutionary Computation*, 1999, 3(1): 22-34.
[20] Yang Z, Tang K, Yao X. Scalability of generalized adaptive differential evolution for large-scale continuous optimization. *Soft Computing*, 2011, 15(11): 2141-2155.
[21] Wang Y, Cai Z, Zhang Q. Enhancing the search ability of differential evolution through orthogonal crossover. *Information Sciences*, 2012, 185(1): 153-177.
[22] Lévy P. Théorie de l'addition des variables aléatoires. Gauthier-Villars Paris, 1937.
[23] Gnedenko B, Kolmogorov A. Limit Distribution for Sums of Independent Random Variables. Cambridge, MA, USA: Addition-Wesley, 1954.
[24] Qin A K, Suganthan P N. Self-adaptive differential evolution algorithm for numerical optimization. In *Proc. the 2005 IEEE Congress on Evolutionary Computation*, Vol.2, September 2005, pp.1785-1791.
[25] Peng F, Tang K, Chen G, Yao X. Population-based algorithm portfolios for numerical optimization. *IEEE Trans. Evolutionary Computation*, 2010, 14(5): 782-800.
[26] Shen L, He J. A mixed strategy for evolutionary programming based on local fitness landscape. In *Proc. the 2010 IEEE Congress on Evolutionary Computation*, July 2010, pp.1-8.
[27] Yao X, Lin G, Liu Y. An analysis of evolutionary algorithms based on neighbourhood and step sizes. In *Proc. the 6th International Conference on Evolutionary Programming VI*, April 1997, pp.297-307.
[28] Schwefel H P. Evolution and Optimum Seeking. John Wiley & Sons, 1995.
[29] Suganthan P N, Hansen N, Liang J J, Deb K, Chen Y P, Auger A, Tiwari S. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Technical Report, Nanyang Technological University, Singapore, 2005.
[30] Yang Z, Tang K, Yao X. Large scale evolutionary optimization using cooperative coevolution. *Information Sciences*, 2008, 178(15): 2985-2999.
[31] Hansen N, Auger A, Finck S, Ros R. Real-parameter black-box optimization benchmarking 2010: Experimental setup. Institut National de Recherche en Informatique et en Automatique (INRIA), 2010.
[32] Hansen N. Compilation of results on the 2005 CEC benchmark function set. Technical Report, Computational Laboratory, Institute of Computational Science, ETH Zurich, 2005.
[33] Liu Y, Yao X, Zhao Q, Higuchi T. Scaling up fast evolutionary programming with cooperative coevolution. In *Proc. the 2001 IEEE Congress on Evolutionary Computation*, May 2001, pp.1101-1108.
[34] van den Bergh F, Engelbrecht A P. A cooperative approach to particle swarm optimization. *IEEE Trans. Evolutionary Computation*, 2004, 8(3): 225-239.
[35] Li X, Yao X. Cooperatively coevolving particle swarms for large scale optimization. *IEEE Trans. Evolutionary Computation*, 2012, 16(2): 210-224.

Ren-Jie He received the B.S. degree in mechatronic engineering, the M.S. degree in automatic control, and the Ph.D. degree in management science from the National University of Defense Technology (NUDT), China, in 1997, 2000, and 2004, respectively. Since 2006, he has been an associate professor at NUDT, where he was a lecturer from 2004 to 2006. His main research interest includes constraint satisfaction, assistant decision-making systems for planning, and intelligent scheduling and optimization.

Zhen-Yu Yang received the B.Eng. and Ph.D. degrees from the University of Science and Technology of China in 2005 and 2010 respectively, both in computer science. From October 2008 to September 2009, he was a visiting Ph.D. candidate of the School of Computer Science, University of Birmingham, U.K., supported by the China Scholarship Council (CSC). From July 2010 to October 2011, he was a lecturer with the Department of Computer Science and Technology, East China Normal University, China. He is currently a lecturer with the Department of Management Science and Engineering, College of Information Systems and Management, National University of Defense Technology, China. His research interests include metaheuristics such as evolutionary algorithms for global optimization, large-scale optimization, and various real-world applications. He is a member of CCF, ACM, and IEEE.