Investigation of Mutation Strategies in Differential Evolution for Solving Global Optimization Problems

Miguel Leon and Ning Xiong

Mälardalen University, Västerås, Sweden

Abstract. Differential evolution (DE) is one competitive form of evolutionary algorithms. It heavily relies on mutating solutions using scaled differences of randomly selected individuals from the population to create new solutions. The choice of a proper mutation strategy is important for the success of an DE algorithm. This paper presents an empirical investigation to examine and compare the different mutation strategies for global optimization problems. Both solution quality and computational expense of DE variants were evaluated with experiments conducted on a set of benchmark problems. The results of such comparative study would offer valuable insight and information to develop optimal or adaptive mutation strategies for future DE researches and applications.

Keywords: Evolutionary Algorithm, Differential Evolution, Mutation Strategies, Global Optimization Problem.

1 Introduction

Evolutionary algorithms (EAs) have been proved to be powerful means to solve various optimization problems[1], [2], [3]. Generally, EAs are superior to traditional optimization techniques in two aspects. First, they require no derivative information of the objective function in deciding the search direction. Second, they perform parallel population-based search and thereby exhibiting more chance to find the global optimum in high dimensional problem spaces [4].

Differential evolution (DE) [5] presents one competitive class of evolutionary algorithms. Unlike other EAs, DE modifies solutions by using the difference of parameter vectors of pair(s) of randomly selected individuals from the population. The locations of the selected solutions decide the direction and magnitude of the search. Therefore the mutation in DE is performed based on the distribution of solutions in the population rather than a pre-specified probability density function.

Indeed there are quite a few alternative strategies to implement the mutation operation in DE [6]. One specific mutation strategy specifies which solution to modify (disturb) and how many difference vectors to use to create the disturbance. If the disturbance is made to a randomly selected solution, it implies that the search direction is decided at random without bias. In contrast, when the disturbance is executed on the best individual in the population, exploitation is more favored in the search process. Hence different strategies of mutation can

L. Rutkowski et al. (Eds.): ICAISC 2014, Part I, LNAI 8467, pp. 372–383, 2014.

reflect different attitudes towards exploration and exploitation. More detailed explanation of the various mutation strategies is given in Sections 2 and 3.

However, there is a lack of knowledge about the comparative performance of the various mutation strategies for solving global optimization problems. In practice it is most common to use a difference vector to mutate a random solution (termed as DE/rand/1 strategy) or the best individual from the population (termed as DE/best/1 strategy). The DE/rand/1 strategy was accepted in previous works as starting point for improved mutation operators. The enhanced mutation approach [7] assumed the usage of the DE/rand/1 strategy and suggested a way to strategically select the three random individuals from the entire space. The paper [8] proposed neighborhood-based mutation for multi-modal optimization tasks. It is realized by employing the DE/rand/1 strategy in local subgroups of the population. Moreover, the DE/rand/1 strategy was also combined with many local search methods for further performance improvement, see examples in [9], [10], [11]. More recently, random local search has been coupled into DE using both DE/rand/1 and DE/best/1 strategies [12]. Nevertheless none of the above mentioned works considered other mutation strategies than DE/rand/1 and DE/best/1 for DE improvement.

This paper presents an empirical investigation to examine and compare the different mutation strategies for global optimization problems. Both solution quality and computational expense of DE variants were evaluated with experiments conducted on a set of benchmark functions. The results of experiments enable us to compare the relative performance of the alternative mutation methods, thereby acquiring valuable insight and information to develop optimal or adaptive strategy for future DE researches and applications.

The remaining of the paper is organized as follows. Section 2 outlines the general DE paradigm. Section 3 explains the alternative mutation strategies. In Section 4 we discuss and compare the experiment results. Finally, Section 5 gives the concluding remarks.

2 Basic Differential Evolution Algorithm

DE is a stochastic algorithm maintaining a population with N_p individuals. Every individual in the population stands for a possible solution to the problem. One individual in the population is represented by vector $X_{i,g}$ with $i = 1, 2, \ldots, N_p$ and g referring to the index of the generation. A cycle in DE consists of three consecutive steps: mutation, crossover and selection which are described as follows:

MUTATION. Inspired from biological evolution, mutation is carried out in DE to facilitate random perturbations on the population. For each population member, a mutant vector is generated. In the basic version of DE, the mutant vector is obtained by randomly selecting three different individuals in the population and then adding a scaled difference of any two of the three vectors to the third one. More precisely, this mutant vector is created according to Eq. 1

$$V_{i,g} = X_{r_1,g} + F \times (X_{r_2,g} - X_{r_3,g}) \tag{1}$$

where $V_{i,g}$ represents the mutant vector, i stands for the index of the vector, g stands for the generation, $r_1, r_2, r_3 \in \{ 1,2,\ldots,N_p\}$ are random integers and F is the scaling factor in the interval [0, 2].

Fig. 1 shows how this mutation strategy works. All the variables in the figure appear in Eq. 1 with the same meaning, and d is the difference vector between $X_{r_2,g}$ and $X_{r_3,g}$.

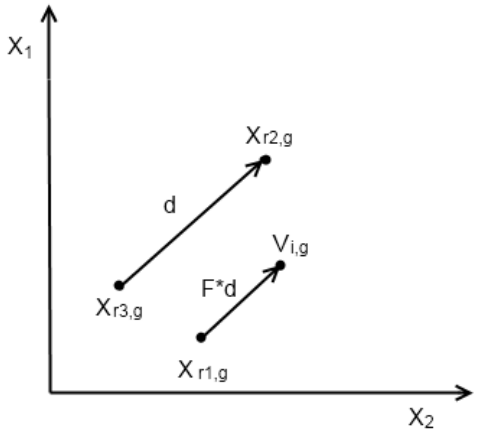

Fig. 1. Random mutation with one difference vector

CROSSOVER. This operation combines every individual in the actual population with the corresponding mutant vector created in the mutation stage. These new solutions created are called trial vectors and we use $T_{i,g}$ to represent the trial vector corresponding to individual i in generation g. Every parameter in the trial vector are decided in terms of Eq. 2

$$T_{i,g}[j] = \begin{cases} V_{i,g}[j] & \text{if } rand[0,1] < CR \quad \text{or} \quad j = j_{rand} \\ X_{i,g}[j] & \text{otherwise} \end{cases} \tag{2}$$

where j stands for the index of every parameter in a vector, J_{rand} is a randomly selected integer between 1 and N_p to ensure that at least one parameter from mutant vector will enter the trial vector and CR is the probability of recombination.

SELECTION. This operation compares a trial vector and its parent solution in the current population to decide the winner to survive into the next generation. Therefore, if the problem of interest is minimization, the individuals in the new generation are chosen using Eq. 3

$$X_{i,g+1} = \begin{cases} T_{i,g} & \text{if } f(T_{i,g}) < f(X_{i,g}) \\ X_{i,g} & \text{otherwise} \end{cases} \tag{3}$$

where $T_{i,g}$ is the trial vector, $X_{i,g}$ is an individual in the population, $X_{i,g+1}$ is the individual in the next generation, $f(T_{i,g})$ represents the fitness value of

the trial vector and $f(X_{i,g})$ stands for the fitness value of the individual in the population.

The pseudocode for the basic DE is given in Fig. 2. First we create the initial population with randomly generated individuals. Every individual in the population is then evaluated with a pre-specified fitness function. After that we create the mutant vectors using Eq. 1 and then we recombine such mutant vectors with their respective parents to obtain a set of offspring. Finally we compare the parents and offspring to select superior ones into a new, updated population. This procedure with steps from 4 to 7 is repeated until the termination condition is satisfied.

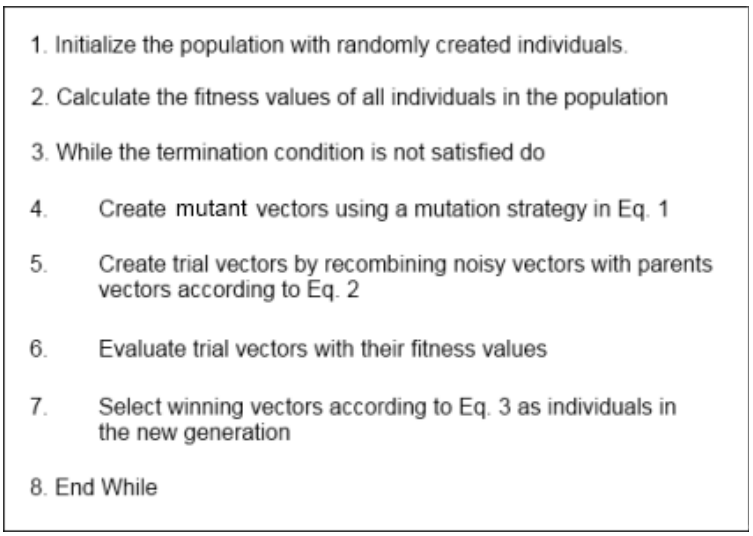

Fig. 2. Pseudocode Differential Evolution

3 Variants of Mutation in Differential Evolution

Various approaches have been proposed to implement mutation in DE [6]. The mutation strategy introduced in the preceding section is termed as random mutation, which is often used in classic DE. In order to distinguish different variants of mutation strategies, the following notation is commonly used in the literature:

$$DE/x/y/z,$$

where x represents the vector to be mutated, y is the number of difference vectors used in mutation and z denotes the crossover operator employed. We will skip z here because we always use the binominal crossover which has been explained in Section 2. Hence the random mutation strategy is notated as DE/rand/1. The five other well Known approaches of mutation will be outlined in the following subsections.

3.1 Best Mutation Strategy with One Difference Vector

Best mutation strategy [13] attempts to mutate the best individual in the population. When only one difference vector is employed in mutation, the approach is represented by DE/best/1. A new, mutated vector is created according to Eq. 4

$$V_{i,g} = X_{best,g} + F \times (X_{r_1,g} - X_{r_2,g}) \tag{4}$$

where $V_{i,g}$ represents the mutant vector, i is the index of the vector, g stands for the generation, $r_1, r_2, r_3 \in \{1,2,\ldots,N_p\}$ are randomly created integers, $X_{best,g}$ represents the best solution in the population and F is the scaling factor in the interval [0, 2].

The main idea of this mutation strategy (notated as DE/best/1) is to use the scaled difference between two randomly selected vectors to mutate the best individual in the population. Fig. 3 shows how a new mutant vector is generated according to this strategy, where d is the difference vector between vectors $X_{r_1,g}$ and $X_{r_2,g}$.

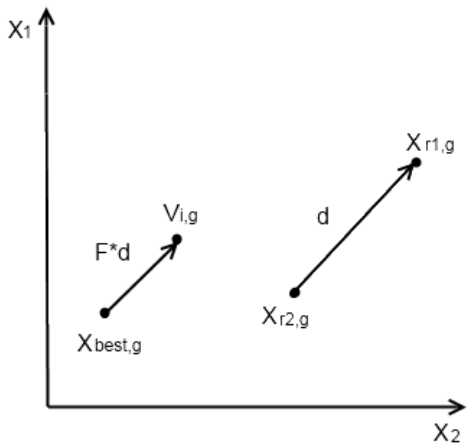

Fig. 3. Best mutation with one difference vector

3.2 Random Mutation Strategy with Two Difference Vectors

Random mutation with two difference vectors [14] is similar to the DE/rand/1 strategy, but it uses two difference vectors and it is notated as DE/rand/2. A mutant vector is created using 5 randomly selected vectors as follows:

$$V_{i,g} = X_{r_1,g} + F1 \times (X_{r_2,g} - X_{r_3,g}) + F2 \times (X_{r_4,g} - X_{r_5,g}) \tag{5}$$

where $F1$, $F2$ are the scaling factors in the interval [0, 2] and $r_1, r_2, r_3, r_4, r_5 \in \{1,2,\ldots,N_p\}$ are randomly created integers, and $V_{i,g}$ stands for the mutant vector i for generation g.

3.3 Best Mutation Strategy with Two Difference Vectors

The strategy of best mutation with two difference vectors [5] is denoted as DE/best/2. It uses two difference vectors in mutation of the best individual of the population. The mutant vector is created in terms of Eq. 6 in the following:

$$V_{i,g} = X_{best,g} + F1 \times (X_{r_1,g} - X_{r_2,g}) + F2 \times (X_{r_3,g} - X_{r_4,g}) \tag{6}$$

where $V_{i,g}$ stands for the the mutant vector, i is the index of the vector, g stands for the generation, $F1$ and $F2$ are the two scaling factors in the interval [0, 2] and $r_1, r_2, r_3, r_4 \in \{ 1,2,\ldots,N_p\}$ are randomly created integers.

3.4 Current to Random Mutation Strategy

The current to rand mutation strategy is referred to as DE/current-to-rand/1. It moves the current individual towards a random vector before being disturbed with a scaled difference of two randomly selected individuals. Thus the mutant vector is created according to Eq. 7 as follows

$$V_{i,g} = X_{i,g} + F1 \times (X_{r_1,g} - X_{i,g}) + F2 \times (X_{r_2,g} - X_{r_3,g}) \tag{7}$$

where $X_{i,g}$ represents the current individual, $V_{i,g}$ stands for the mutant vector, g stands for the generation, i is the index of the vector, $F1$ and $F2$ are the scaling factors in the interval [0, 2] and $r1, r2, r3 \in \{ 1,2,\ldots,N_p\}$ are randomly created integers.

Fig. 4 explains how the DE/current-to-rand/1 strategy works to produce a mutant vector, where $d1$ is the difference vector between the current individual, $X_{i,g}$, and $X_{r_1,g}$, and $d2$ is the difference vector between $X_{r_3,g}$ and $X_{r_2,g}$.

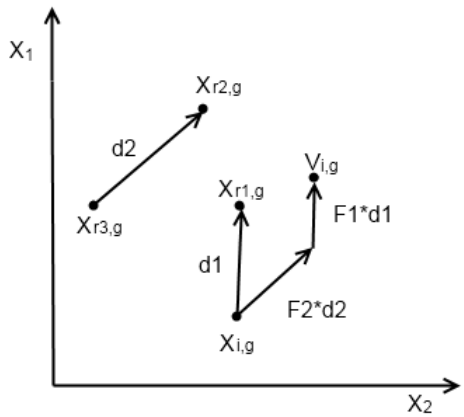

Fig. 4. Current to random mutation

3.5 Current to Best Mutation Strategy

The current to best mutation strategy [14] is referred as DE/current-to-best/1. It moves the current individual towards the best individual in the population before being disturbed with a scaled difference of two randomly selected vectors. Hence the mutant vector is created by

$$V_{i,g} = X_{i,g} + F1 \times (X_{best,g} - X_{i,g}) + F2 \times (X_{r_1,g} - X_{r_2,g}) \tag{8}$$

where $V_{i,g}$ stands for the mutant vector, $X_{i,g}$ and $X_{best,g}$ represent the current individual and the best individual in the population respectively, $F1$ and $F2$ are the scaling factors in the interval [0, 2] and $r_1, r_2 \in \{ 1,2,\ldots,N_p\}$ are randomly created integers.

Fig. 5 shows how the DE/current-to-best/1 strategy works to produce a mutant vector, where $d1$ denotes the difference vector between the current individual $X_{i,g}$, and $X_{best,g}$, $d2$ is the difference vector between $X_{r_1,g}$ and $X_{r_2,g}$.

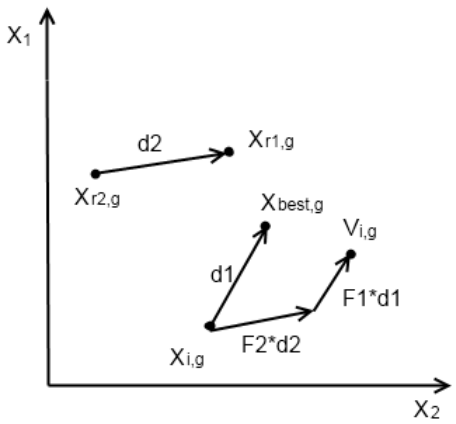

Fig. 5. Current to best mutation

4 Experiments and Results

We tested the performance of the six variants of mutation strategies in DE on a set of benchmark problems. Thirteen different mathematic functions from [15] were used in our experiments, which are highlighted in Table 1. The dimensions of all these functions are 30, with functions 1 to 7 being unimodal and functions 8 to 13 being multimodal.

4.1 Experimental Settings

The three control parameters for DE are: population size (Np), crossover rate (CR) and the scaling factor (F) for mutation. These parameters used in our experiments were specified as follows: $Np = 60$, $CR = 0.85$ and $F = 0.9$ when

Table 1. The thirteen functions used in the experiments

FUNCTION
$f1(x) = \sum_{i=1}^{n} x_i^2$
$f2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$
$f3(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$
$f4(x) = max_i\{\lvert x_i \rvert, 1 \leq i \leq n\}$
$f5(x) = \sum_{i=1}^{n-1} [100 \times (x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$
$f6(x) = \sum_{i=1}^{n} (x_i + 0.5)^2$
$f7(x) = \sum_{i=1}^{n} i \times x_i^4 + random[0, 1)$
$f8(x) = \sum_{i=1}^{n} -x_i \times sin(\sqrt{\lvert x_i \rvert})$
$f9(x) = \sum_{i=1}^{n} [x_i^2 - 10 \times cos(2 \times \pi \times x_i) + 10]$
$f10(x) = -20 \times exp(-0.2 \times \sqrt{\frac{1}{n} \times \sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n} \times \sum_{i=1}^{n} cos(2\pi x_i)) + 20 + e$
$f11(x) = \frac{1}{4000} \times \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) + 1$
$f12(x) = \frac{\pi}{n} \times \{10sin^2(\pi y_i) + \sum_{i=1}^{n-1} ((y_i - 1)^2 [1 + 10sin^2(\pi y_{i+1})]) + (y_n - 1)^2\} +$ $+ \sum_{i=1}^{n} u(x_i, 10, 100, 4)$, where $y_i = 1 + \frac{1}{4}(x_i + 1)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \leq x_i \leq a \\ k(x_i - a)^m, & x_i < -a \end{cases}$
$f13(x) = 0.1 \times \{sin^2(3\pi x_1) + \sum_{i=1}^{n-1} ((x_i - 1)^2 [1 + sin^2(3\pi x_{i+1})]) +$ $+ (x_n - 1)[1 + sin(2\pi x_n)^2]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$

only one difference vector is used and $F1 = 0.3$ and $F2 = 0.7$ when two difference vectors are involved (as in DE/rand/2 and DE/Best/2). The DE variants (with different mutation strategies) were applied and tested on the benchmark functions in attempts to find the best solutions for them. Every DE variant was executed for each function 20 times to get a fair result for the comparison. The terminate condition for the execution of the DE programs is that the error with respect to the global optimum is below $10e$-8 or the number of fitness evaluations has exceeded $300,000$.

The results of experiments will be demonstrated in the following. First we compare the performance (the quality of acquired solutions) of the DE variants in the benchmark functions, and secondly we compare the convergence speed of the DE variants in finding their optima solutions.

4.2 Comparison of the Quality of Solutions

First we consider the quality of solutions obtained by the DE variants. The results obtained on the thirteen benchmark functions are showed in Table 2. The first column corresponds to the test functions used for evaluation. All the other columns are used to present the average error of the results obtained for a certain function with respect to its global optimal value. A figure in boldface means the lowest average error among those achieved by the DE variants.

We compare the quality of solutions on unimodal and multimodal functions respectively. It can be seen from Table 2 that, in unimodal functions (functions

Table 2. Average error of the found solutions

FUNCTION	DE/rand/1	DE/best/1	DE/rand/2	DE/best/2	DE/ctor/1	DE/ctob/1
f1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f2	1.58E-08	4.70E-05	**0.00E+00**	5.00E-01	**0.00E+00**	**0.00E+00**
f3	5.81E+01	2.50E+02	1.69E-01	2.32E-07	8.73E-06	**0.00E+00**
f4	5.81E+00	1.58E-02	1.10E-02	9.08E-06	3.11E-02	**2.35E-08**
f5	2.66E+01	6.54E+00	6.86E-04	7.98E-01	**0.00E+00**	2.00E-01
f6	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f7	1.12E-02	9.58E-03	5.73E-03	5.14E-03	**2.63E-03**	2.65E-03
f8	**2.47E+03**	3.24E+03	6.62E+03	3.29E+03	6.94E+03	2.59E+03
f9	**1.09E+01**	4.24E+01	1.72E+02	4.51E+01	1.53E+02	2.57E+01
f10	1.88E+01	1.40E+01	**0.00E+00**	2.05E+00	**0.00E+00**	**0.00E+00**
f11	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f12	**0.00E+00**	9.85E-02	**0.00E+00**	5.71E-02	**0.00E+00**	**0.00E+00**
f13	**0.00E+00**	3.83E-03	**0.00E+00**	5.47E-03	**0.00E+00**	**0.00E+00**

1-7) , DE/current-to-rand/1 and DE/current-to-best/1 appeared as the strongest alternative sine they reached the global optimum in four of the seven functions. DE/best/2 and DE/rand/2 were also very good in the 7 unimodal functions, with results very similar to those of the strongest candidates. The weakest approaches here were DE/rand/1 and DE/best/1, as they produced significant error in some cases such as function 3, 4, and 5, as shown in the table.

In multimodal functions (functions 8-13), DE/current-to-best/1 appeared as the best choice, as it found the results similar to the best ones on functions 8 and 9 and the global optima on all the other functions. DE/rand/1 was also attractive, as it found the best solutions on functions 8 and 9. In function 10, DE/rand/1 and DE/best/1 got the worst performance with large error, while all the other candidates acquired the optimum or near optimal solutions. Finally, on functions 11, 12 and 13 all the approaches behaved equally well with optimal or close to optimal results.

Based on the above analysis, we can point out that, for unimodal functions, it is better to use DE/current-to-best/1 or DE/current-to-rand/1. But good results can be achieved with DE/best/2 and DE/rand/2 as well. DE/rand/1 and DE/best/1 are very weak in unimodal functions. In multimodal functions the best results can be achieved with DE/current-to-best/1 and DE/rand/1, except in function 10 for which the performance of DE/rand/1 is not good. DE/current-to-rand/1 and DE/rand/2 can have very poor performance occasionally, such as in functions 8 and 9.

4.3 Comparison of the Computational Cost

In this subsection we compare the numbers of fitness evaluations that were done by the DE variants before the global optima were reached. We did this in the following way: First we recorded the number of DE executions from which the global optimum was reached. These numbers are given in Table 3. Then we

calculated the average number of evaluations for every DE variant on each test function, as listed in Table 4.

Table 3. Number of executions with acquired optimum

FUNCTION	DE/rand/1	DE/best/1	DE/rand/2	DE/best/2	DE/ctr/1	DE/ctb/1
f1	**20**	**20**	**20**	**20**	**20**	**20**
f2	6	18	**20**	19	**20**	**20**
f3	0	0	0	0	0	**18**
f4	0	0	0	0	0	**6**
f5	0	0	0	16	**19**	**19**
f6	**20**	**20**	**20**	**20**	**20**	**20**
f10	0	6	**20**	17	**20**	**20**
f11	**20**	**20**	**20**	**20**	**20**	**20**
f12	**20**	15	**20**	17	**20**	**20**
f13	**20**	16	**20**	17	**20**	**20**

In Table 3 we can observe that DE/current-to-best/1 always got the highest success rate. DE/current-to-rand/1 and DE/rand/2 got success rate similar to that of DE/current-to-best/1 in many cases. No comparison was made on functions 7, 8 and 9 since the DE variants never reached the global optima on these functions. The worst algorithm in this comparison was DE/rand/1 because it never found an optimum in seven functions and on function 2 its success rate was very low.

Table 4. Mean number of evaluations conducted

FUNCTION	DE/rand/1	DE/best/1	DE/rand/2	DE/best/2	DE/ctr/1	DE/ctb/1
f1	211008	103353	96315	57720	64494	**54132**
f2	299013	158205	166530	99303	115071	**94536**
f3	300000	300000	300000	300000	300000	**287898**
f4	300000	300000	300000	300000	300000	**294825**
f5	300000	300000	300000	240870	287796	**206241**
f6	212178	108045	96369	57819	64962	**54189**
f10	300000	260661	153717	132552	102117	**86028**
f11	177993	90879	81240	48219	54213	**45786**
f12	185953	157440	97425	132786	61422	**51192**
f13	208842	147054	104472	95055	67845	**55629**

From Table 4 it is clear that DE/current-to-best/1 was the fastest algorithm on all the functions. Then we attempt to identify the second fastest alternative from DE/current-to-rand/1 and DE/best/2. For this purpose we do comparison on unimodal functions (functions 1 to 7) and multimodal functions (functions 8 to 13) respectively. On unimodal functions, DE/best/2 was better in all of the four functions for comparison, but the difference was not so large. On multimodal

functions, DE/current-to-rand/1 was better in three of the four functions with large differences. Overall the worst algorithm was DE/rand/1 because it took the most evaluations in all the 8 unimodal and multimodal functions (functions 3 and 4 are excluded in this comparison). Based on these results we can point out that DE/current-to-best/1 is a faster DE variant while DE/rand/1 is a slower one.

5 Conclusions

This paper presents an empirical study to compare six different mutation strategies in optimization problems. All these mutation approaches have been tested in a set of benchmark problems in terms of both the quality of solutions and the computational expense, i.e. the number of fitness evaluations required. The results of experiments have led to the recommendation of the DE/current-to-best/1 strategy, which is not only computationally efficient but also superior in guaranteeing the quality of solutions in a diversity of problems.

It is important to be aware of the relative performance of distinct mutation strategies to develop competent DE algorithms. In future we will exploit the information acquired in this paper for construction of optimal or adaptive strategies of mutation to tackle more complex and larger scale optimization tasks. Moreover, we will also apply and test our new computing algorithms in real industrial scenarios.

Acknowledgment. The work is funded by the Swedish Knowledge Foundation (KKS) grant (project no 16317). The authors are also grateful to ABB FACTS, Prevas and VG Power for their co-financing of the project.

References

1. Herrera, F., Lozano, M., Verdegay, J.: Tackling real-coded genetic algorithms: Operators and tools for the behavioral analysis. Artificial Intelligence Review 12, 265–319 (1998)
2. Beyer, H., Schwefel, H.: Evolution strategies: A comprehensive introduction. Natural Computing 1, 3–52 (2002)
3. Lee, C., Yao, X.: Evolutionary programming using mutations based on the levy probability distribution. IEEE Transactions on Evolutionary Computation 8, 1–13 (2004)
4. Xiong, N., Leon, M.: Principles and state-of-the-art of engineering optimization techniques. In: Proc. The Seventh International Conference on Advanced Engineering Computing and Applications in Sciences, ADVCOMP 2013, Porto, Portugal, pp. 36–42 (2013)
5. Storn, R., Price, K.: Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11(4), 341–359 (1997)
6. Price, K., Storn, Lampinen, J.: Differential evolution a practical approach to global optimization. Springer Natural Computing Series (2005)

7. Kumar, P., Pant, M.: Enhanced mutation strategy for differential evolution. In: IEEE Congress on Evolutionary Computation (CEC), pp. 1–6 (2012)
8. Qu, B., Suganthan, P., Liang, J.: Differential evolution with neighborhood mutation for multimodal optimization. IEEE Transactions on Evolutionary Computation 16, 601–614 (2012)
9. Noman, N., Iba, N.: Enhancing differential evolution performance with local search for high dimensional function optimization. In: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, GECCO 2005, pp. 967–974 (2005)
10. Dai, Z., Zhou, A.: A differential evolution with an orthogonal local search. In: Proc. 2013 IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico, pp. 2329–2336 (2013)
11. Poikolainen, I., Neri, F.: Differential evolution with concurrent fitness based local search. In: Proc. 2013 IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico, pp. 384–391 (2013)
12. Leon, M., Xiong, N.: Using random local search helps in avoiding local optimum in differential evolution. In: Proc. Artificial Intelligence and Applications, AIA 2014, Innsbruck, Austria, pp. 413–420 (2014)
13. Xu, H., Wen, J.: Differential evolution algorithm for the optimization of the vehicle routing problem in logistics. In: Proc. 2012 Eighth International Conference on Computational Intelligence and Security (CIS), Guangzhou, China, pp. 48–51 (2012)
14. Gong, W., Cai, Z.: Differential evolution with ranking-based mutation operators. IEEE Transactions on Cybernetics PP, 1–16 (2013)
15. Yao, X., Liu, Y., Lin, G.: Evolutionary programming made faster. Proc. IEEE Transactions on Evolutionary Computation 3, 82–102 (1999)