Analyzing Effects of Ordering Vectors in Mutation Schemes on Performance of Differential Evolution

Sedigheh Mahdavi, Shahryar Rahnamayan, IEEE Senior Member, and Chirag Karia
Department of Electrical, Computer, and Software Engineering University of Ontario Institute of Technology, Oshawa, Canada
Email: Sedigheh.Mahdavi@uoit.ca, Shahryar.Rahnamayan@uoit.ca, chirag.karia@uoit.net

***Abstract*—Differential Evolution (DE) is a simple powerful evolutionary algorithm for solving global continuous optimization problems. The especial characteristic of DE algorithm is calculating a weighted difference vector of two random candidate solutions in the population to generate the new promising candidate solutions. A major operation of the DE algorithm is the mutation which can affect its performance. The main goal of this study is investigating the influence of ordering vectors on various mutation schemes. We design some Monte-Carlo based simulations to analyze several mutation schemes by calculating the probability of closeness of a new trial solutions to a random optimal solution. These simulations indicate that mutation schemes can enhance the performance of the DE algorithm which they consider right ordering of the vectors in their mutation operators. Also, we introduce a new mutation scheme which considers in ordering vectors in the mutation scheme. We benchmark the modified DE algorithm with the ordered mutation scheme (DE/order) on CEC-2014 test functions with three dimensions 30, 50, and 100. Simulation results confirm that DE/order obtains a promising performance on the majority of the test functions on all mentioned dimensions.**

I. Introduction

Global optimization problem has been arisen in many scientific and engineering applications. A global optimization problem can be mathematically formulated as

$$min \quad f(x) \tag{1}$$
$$s.t. \quad x \in \Omega \tag{2}$$

Where Ω is the decision space and x is a decision vector. Many of these global optimization problems cannot be easily solved because they are faced with several challenging features, for example, they can be non-linear, non-convex, multi-modal and non-differentiable. Differential Evolution (DE) was proposed by Storn and Price in 1995 to find the global optimum in challenging optimization problems [1], [2]. DE is a kind of evolutionary algorithm (EA) with an especial mutation operator which uses the difference among the candidate solutions of the population. The main motivation of DE is designing a new mutation strategy which creates a new trial solution by adding a weighted difference vector of two candidate solutions to a third candidate solution as the base vector. One cycle of a DE algorithm executes three basic steps: mutation, crossover, and selection. DE repeats these steps until the stopping criteria are satisfied. A large number of modified DE algorithms have been proposed to analyze and enhance the various steps of DE algorithm [3], [4], such as , modifying crossover and mutation schemes [5], [6], [7], [8], [9], [10], [11], adjusting three control parameters, i.e., population size NP, scale factor F, and crossover rate CR [12], [13], [14], [15], new initialization strategy [16], [17], controlling population's diversity [18], [19], designing self-adaptive strategies [20], [21], etc.

For each parent candidate solution i, the basic mutation operator (DE/rand) in the DE algorithm selects three random candidate solutions x_{i_1}, x_{i_2}, and x_{i_3} from the population to create a mutant candidate solution as the donor solution. The indices x_{i_1}, x_{i_2}, and x_{i_3} are different from the parent candidate solution i. Several mutation schemes of DE have been proposed to improve its performance for solving optimization problems [3], [4]. There are some modifications such as adding several weighted difference vectors to the base vector, adding weighted difference vectors of several candidate solutions (e.g., x_{i_1}+x_{i_2}-x_{i_3}), and using the special candidate solution as the base vector like the best candidate solution of three candidate solutions. In [22], a new version of mutation operator was proposed which sorts two first candidate solutions in the mutation operator according to their fitness in ascending order to set as vectors x_{i_1} and x_{i_2}. Also, the winner mutation strategy [23] was introduced to identify nonlinear system which uses the best candidate of three randomly selected candidate solutions as the base vector in the mutation equation. These methods attempt to investigate which candidate solution can be a potential candidate solution to set as the base vector in the mutation operator but they do not consider the direction of the weighted difference vector. Also, they have not analyzed their proposed schemes in detail.

This paper aims at analyzing the order of candidate solutions to place as vectors of the mutation operator which can affect significantly the performance of the DE algorithm. Some simulations are designed and conducted to investigate which arrangement of candidate solutions can generate promising new trial candidate solutions. Also, we propose a new mutation scheme, called the ordered mutation, which uses the ascending order (for minimization problems) of three randomly selected candidate solutions corresponding to their objective function values to place as the vectors of the mutation scheme. It considers both the direction of the weighted difference vector and setting a potential candidate solution as the base vector. The modified DE algorithm with the ordered mutation scheme (DE/order) is evaluated on CEC-2014 benchmark functions on three dimensions 30, 50, and 100. Simulation results confirm that on the majority of the benchmark functions, the proposed scheme performs better in overall.

The organization of the rest of the paper is as follows. Section II presents a background review. Section III describes the details of modified DE algorithm with the order mutation

978-1-5090-4601-0/17/$31.00

scheme. Section IV explains conducting Monte-Carlo based simulations. Section V presents the experimental results. Finally, the paper is concluded in Section VI.

II. BACKGROUND REVIEW

A. Differential Evolution- A Brief Description

Differential Evolution (DE) was proposed by Price and Storn [2] in 1995. DE operates on a population including NP randomly candidate solutions. There are two main stages, namely, initialization and evolution. DE starts with some randomly generated candidate solutions as an initial population. DE applies two operators, mutation and crossover, to generate new trial solutions during the evolutionary process. The variant versions of DE was proposed according to the used schemes of crossover and mutation which are denoted as DE/x/y/z. In this DE/x/y/z notation, x indicates the candidate solution which is used as a base vector to change, y is the number of difference vectors, and z indicates the type of crossover (i.e., bin or exp). In following, some versions of mutation and crossover schemes are briefly described. The basic mutation, rand scheme, in the classical DE (DE/rand/1) generates the mutant vector as a linear combination of three selected individual candidate solutions from the current population as follows:

$$v_i = x_{i_1} + F.(x_{i_2} - x_{i_3}), \tag{3}$$

where i_1, i_2, i_3 are different random integer numbers within $[1, NP]$ and NP is the population size. The scaling factor F is the real constant factor to control the difference vector. The other three different mutation schemes, suggested by Storn and Price [1], [2] are listed as below:

$$DE/rand/2: \ v_i = x_{i_1} + F.(x_{i_2} - x_{i_3}) + F.(x_{i_4} - x_{i_5}), \tag{4}$$

$$DE/best/1: \ v_i = x_{best} + F.(x_{i_2} - x_{i_3}), \tag{5}$$

$$DE/best/2: \ v_i = x_{best} + F.(x_{i_2} - x_{i_3}) + F.(x_{i_4} - x_{i_5}), \tag{6}$$

Where i_1-i_5 are different random integer numbers within $[1, NP]$ and x_{best} is the current best candidate solution in the population. In [22], a new scheme of mutation operator, DE/2-Opt, was proposed which sorts two first candidate solutions in the mutation operator according to their objective function value in ascending order to place as x_{i_1} and x_{i_2} in the mutation operator as:
'DE/2-Opt/1':

$$v_i = \begin{cases} x_{i_1} + F.(x_{i_2} - x_{i_3}) & \text{if} \quad f(x_{i_1}) < f(x_{i_2}) \\ x_{i_2} + F.(x_{i_1} - x_{i_3}) & \text{if} \quad f(x_{i_2}) < f(x_{i_1}) \end{cases} \tag{7}$$

'DE/2-Opt/2':

$$v_i = \begin{cases} x_{i_1} + F.(x_{i_2} - x_{i_3} + x_{i_4} - x_{i_5}) & \text{if} \quad f(x_{i_1}) < f(x_{i_2}) \\ x_{i_2} + F.(x_{i_1} - x_{i_3} + x_{i_4} - x_{i_5}) & \text{if} \quad f(x_{i_2}) < f(x_{i_1}) \end{cases} \tag{8}$$

Another scheme is the winner mutation (DE/win) [23] which uses the best candidate of three randomly selected candidate solutions for the base vector as follows,
'DE/win/1':

$$v_i = \begin{cases} x_{i_1} + F.(x_{i_2} - x_{i_3}) & \text{if} \quad f(x_{i_1}) < f(x_{i_2}), f(x_{i_3}) \\ x_{i_2} + F.(x_{i_1} - x_{i_3}) & \text{if} \quad f(x_{i_2}) < f(x_{i_1}), f(x_{i_3}) \\ x_{i_3} + F.(x_{i_2} - x_{i_1}) & \text{if} \quad f(x_{i_3}) < f(x_{i_2}), f(x_{i_1}) \end{cases} \tag{9}$$

There are two crossover methods, namely, exponential and binomial. The binomial crossover operator integrates the parameter values of the mutant vector with some selected individual candidate solutions of the current population to generate the final offspring vector. In [1], [2], the binomial crossover is defined to generate a trial vector as follows:

$$u_{z,j} = \begin{cases} v_{z,j} & \text{rand}() \leq CR \text{ or } j\text{=}j_{rand} \\ x_{z,j} & \text{otherwise} \end{cases} \tag{10}$$

where CR is the crossover rate, a constant value within the interval $[0, 1)$ and j_{rand} is a random number in $1, 2, \ldots, D$; D is the problem dimension. In the exponential crossover, first an integer n is randomly selected among the numbers in $[1, D]$ which is the starting point for the exponential crossover. Also, another integer L are taken from the mutant vector which is selected by executing of the following pseudo-code:
L = 0; DO
{
L = L + 1;
} WHILE ((rand(0, 1) = CR) AND (L = D)). Then, the trial vector is calculated as:

$$u_{z,j} = \begin{cases} v_{z,j} & \text{for} \, j = \langle n \rangle_D, \langle n+1 \rangle_D, \ldots, \langle n+L-1 \rangle_D \\ x_{z,j} & \text{otherwise} \end{cases} \tag{11}$$

Where the angular brackets $\langle\rangle_D$ denotes a modulo function with modulus D. After that, DE selects the better one between x_i and u_i according to their fitness values for the next generation (i.e., greedy selection).

III. A MODIFIED DE ALGORITHM WITH THE ORDERED MUTATION SCHEME

Through several research works on the DE algorithms [3], [4], it has been observed that the performance of DE can be influenced by modifying the mutation scheme. Therefore, the mutation scheme and its corresponding direction information are beneficial to guide the search in the DE algorithm. DE selects three different random candidate solutions (x_{i_1}, x_{i_2}, and x_{i_3}) from the current population to generate new candidate trial solutions during the mutation step. In this section, we propose a modified DE algorithm with the order mutation scheme which uses the objective function of three different random candidate solutions to arrange order of vectors in mutation scheme. In DE/order/1, first, three selected random solutions are sorted in ascending order according to their fitness values for placing as vectors (x_{i_1}, x_{i_2}, and x_{i_3}) in the mutation. The sorted candidate solutions can be called as the best, the second best, the worst candidate solutions. Then, the best, the second best, and the worst candidate solutions are considered as solutions x_{i1}, x_{i_2}, and x_{i_3} in the mutation $v_i = x_{i_1} + F.(x_{i_2} - x_{i_3})$.
'DE/order/1':

$$v_i = x_{i_1} + F.(x_{i_2} - x_{i_3}) \tag{12}$$

$$s.t \ f(x_{i_1}) < f(x_{i_2}) < f(x_{i_3}) \tag{13}$$

Where $f(x)$ indicates the objective function for the value of solution x.

The main modification of this mutation is placing the best candidate solution of three candidate solutions as the

base vector and selecting two other candidate solutions as the potential candidate solutions in the computing of the difference vector in the mutation operator in ordered. In DE/best/1, as we mentioned in the section II-A, the current best solution of the population is placed as the base vector to explore more promising regions but it can be trapped in local optimal solution because of using the current best solution to generate all new trial solutions. In the proposed DE/order/1, the best of three selected candidate solution are used as the base vector which was proposed in [23] to modify DE for nonlinear system identification. Placing the best candidate solutions of three selected ones as the base vector instead of the current best solution in the population causes to generate the various solutions located in the different regions of the search space which would not be prone for premature convergence similar to the previous case. Therefore, it can avoid trapping in local optimal solution. We suggest putting the worse candidate solution of three candidate solutions as third vector in the mutation which causes that the new trial candidate solution to get away from the worse candidate solution and move forward toward the second best candidate solution. It means that after selecting three candidate solutions randomly from the population, we use the sorting order of three random selected solutions to place as vectors corresponding to their ascending sorting of their fitness.

Fig. 1 demonstrate an example to indicate the behavior of the proposed mutation scheme. It includes three candidate solutions x_1, x_2, and x_3 and the optimal solution. As we can see that the direction of the difference vector $(x_2 - x_3)$ is toward the candidate solution x_2 which causes to go away from the worst candidate solution. Then, by adding the difference vector $(x_2 - x_3)$ to the best candidate solution x_1 the new point V is generated. To see the difference between two directions of difference vector, it can be seen that by adding the difference vector $(x_3 - x_2)$ to x_1, it reaches the new point V_1 which is far away from the optimal solution and moves toward the worst solution. This mutation scheme is very simple and it does not need any extra computation because the complexity of sorting three candidate solution is constant. Also, it keeps all other steps of DE algorithms untouched.

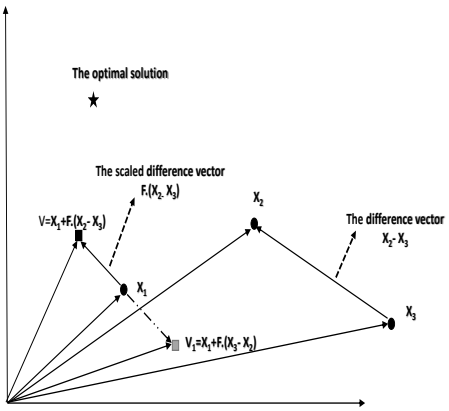
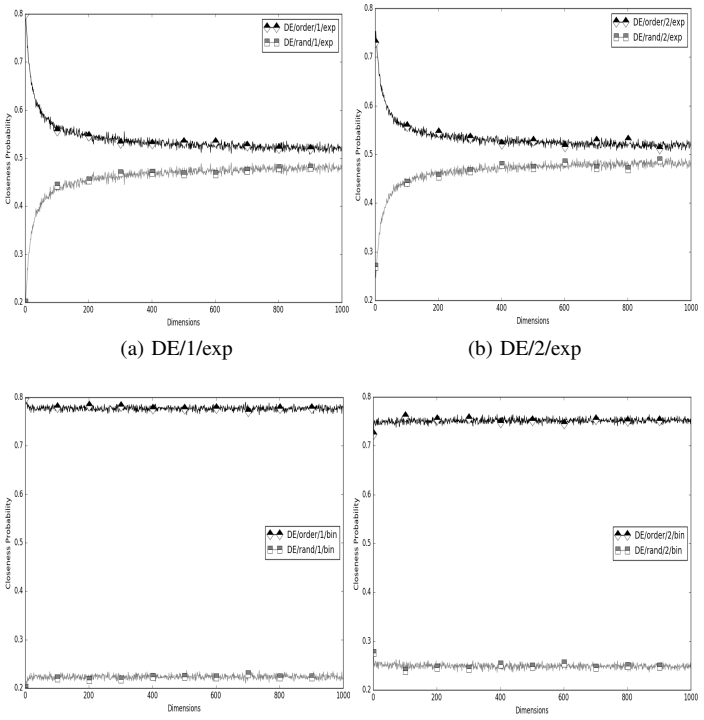


Fig. 1: Illustrating a sample behavior of DE/order/1 in 2-D space.



(a) DE/1/exp (b) DE/2/exp

(c) DE/1/bin (d) DE/2/bin

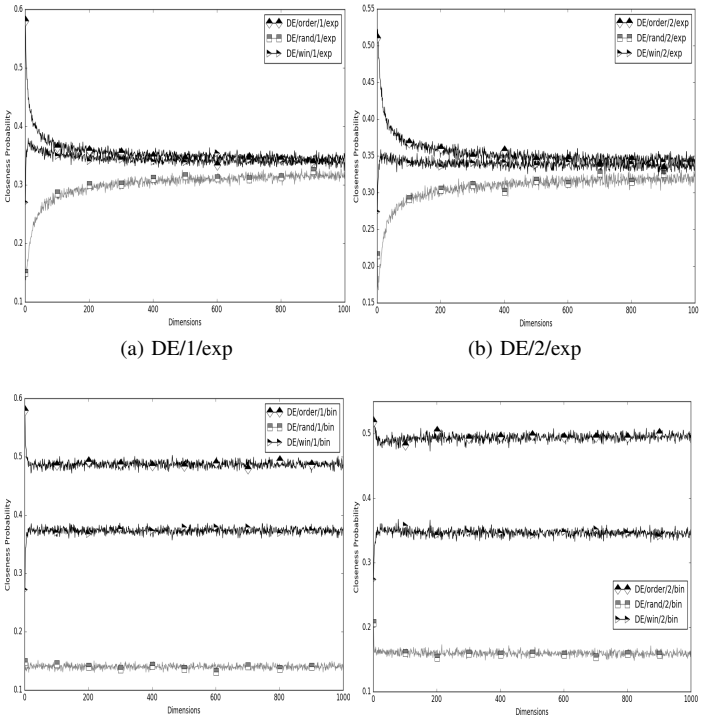
Fig. 2: The probability of closeness for DE/order and DE/rand mutation schemes for the exponential and binomial crossover operators with CR = 0.9 and F = 0.5

A. Analyzing mutation schemes of DE by Monte-Carlo based Simulations

In order to further analyzing the effect of ordering vectors in the mutation schemes, we demonstrate the behaviour of some mutation schemes in the DE algorithm by implementing a series of simulations. Monte-Carlo simulations are applied to calculate the probability of closeness (to a randomly generated solution) of new candidate trial solutions obtained by some mutation schemes. The simulations run for 10^4 times, a solution is generated uniform randomly as an optimal solution and some candidate solutions (i.e., three or more depending on the mutation scheme under study) are generated uniform randomly. Also, in these simulations it is supposed that the Euclidean distances of each candidate solution is consider as its objective function value (closer to solution has better fitness value). Each trial of these simulations has following steps:

1) An optimal solution and several candidate solutions are generated uniform randomly.
2) The Euclidean distances of all candidate solutions to the optimal solution are computed as the objective function values of candidate solutions which are used to sort the candidate solutions in some mutation schemes.
3) One candidate solution is randomly selected as the parent solution.
4) The new trial candidate solutions are calculated by using the different schemes of mutation and crossover operators.

5) The Euclidean distances of new trial candidate solutions to the optimal solution are computed to determine the best mutation scheme which can generate a better new trial candidate solution; closer to the the optimal solution.



(a) DE/1/exp (b) DE/2/exp

(c) DE/1/bin (d) DE/2/bin

Fig. 3: The probability of closeness for DE/order, DE/win, and DE/rand mutation schemes for the exponential and binomial crossover operators with CR = 0.9 and F = 0.5

The closeness (to solution) probability of a mutation scheme is computed as the number of trials which it can generate the better (i.e., closer) trial candidate solution divided by the number of total trials. Note that in these simulations, we generate new trial candidate solutions in each iteration by performing the different mutation schemes in order to recognize which mutation scheme can generate the better trial candidate solutions. These simulations run on various dimensions from 1D to 1000D. The simulations consider four schemes DE/win, DE/rand, DE/2-opt and our proposed mutation scheme, i.e., DE/order. Two parameters CR and F are set to 0.9 and 0.5, respectively. Also, the winner mutation (DE/win) can be extended for using DE with two difference vectors as follows:
'DE/win/2':

$$v_i = \begin{cases} x_{i_1} + F.(x_{i_2} - x_{i_3} + x_{i_4} - x_{i_5}) & \text{if} f(x_{i_1}) < f(x_{i_2}), f(x_{i_3}) \\ x_{i_2} + F.(x_{i_1} - x_{i_3} + x_{i_4} - x_{i_5}) & \text{if} f(x_{i_2}) < f(x_{i_1}), f(x_{i_3}) \\ x_{i_3} + F.(x_{i_2} - x_{i_1} + x_{i_4} - x_{i_5}) & \text{if} f(x_{i_3}) < f(x_{i_1}), f(x_{i_2}) \end{cases} \quad (14)$$

DE/order is also extended for DE with two difference vectors as follows:
'DE/order/2':

$$v_i = x_{i_1} + F.(x_{i_2} - x_{i_3}) + F.(x_{i_4} - x_{i_5}) \quad (15)$$

$$s.t\ f(x_{i_1}) < f(x_{i_2}) < f(x_{i_3}) < f(x_{i_4}) < f(x_{i_5}) \quad (16)$$

Fig. 2 indicates the closeness probability of DE/order and DE/rand mutation schemes with the exponential and binomial crossover operators. As we can see, in the exponential crossover, DE/order has higher probability than DE/rand especially when dimension is less than 200. By increasing dimension, the probability of DE/order is decreased but it is still higher than DE/rand. Also, the probability of DE/rand is increased as the dimension increases. For the binomial crossover, DE/order has higher probability than DE/rand and this behaviour is constant as the dimension increases. Fig. 3 indicates the closeness probability of DE/order, DE/win, and DE/rand mutation schemes with the exponential and binomial crossover operators. As we can see, in the exponential crossover, DE/order and DE/win mutation schemes has higher probability than DE/rand especially when dimension is less than 200. Also, by increasing dimension, the probability of DE/rand is increased while the closeness probability of two other mutation schemes is decreased. For the binomial crossover, DE/order and DE/win mutation schemes have higher probability than DE/rand and this behaviour is constant for higher dimensions.

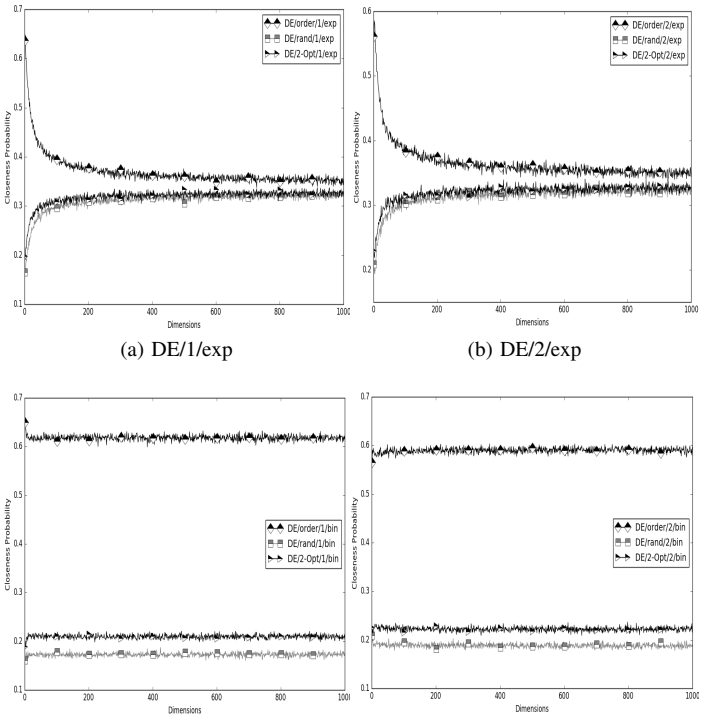
Fig. 4 indicates the closeness probability of DE/order, DE/2-opt, and DE/rand mutation schemes with the exponential and binomial crossover operators. As we can see that in the exponential crossover, DE/order has higher probability than DE/rand and DE/win mutation schemes especially when dimension is less than 200. Also, by increasing dimension, the probability of DE/rand and DE/win mutation schemes are increased while the closeness probability of two other mutation schemes are decreased. For the binomial crossover, DE/order and DE/2-opt mutation schemes have higher probability than DE/rand mutation scheme which there is a significant difference between the closeness probability of DE/order (approximately 0.2) and DE/2-opt (approximately 0.6) mutation schemes.

Fig. 5 indicates the closeness probability of DE/order, DE/2-opt, DE/win and DE/rand mutation schemes with the exponential and binomial crossover operators. It indicates that in the exponential crossover, three mutation schemes DE/order, DE/2-opt, and DE/win has higher probability than DE/rand. As we can see in Fig. 5 that the behaviour of DE/order and DE/win mutation schemes are similar, increasing curve; while DE/rand and DE/2-opt mutation schemes have the decreasing curve. For the binomial crossover, DE/order, DE/2-opt, and DE/win mutation schemes have higher probability than DE/rand mutation scheme while the closeness probability of DE/order and DE/win mutation schemes are values greater than 0.3 and 0.4; respectively. Monte carlo simulations confirm that when mutation schemes consider order candidate solutions to place as vectors in the mutation, they can generate better new trial solutions than the random scheme.

IV. EXPERIMENTAL RESULTS

A. Setup of Experiments

To study the performance of the modified DE algorithm with the order mutation scheme (DE/order), we compare



(a) DE/1/exp (b) DE/2/exp

(c) DE/1/bin (d) DE/2/bin

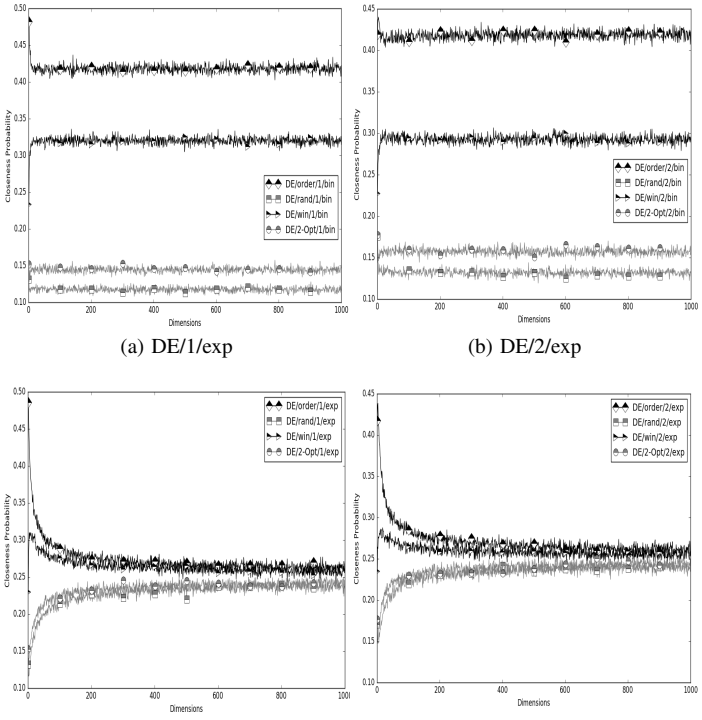
Fig. 4: The probability of closeness for DE/order, DE/2-opt, and DE/rand mutation schemes for the exponential and binomial crossover operators with CR = 0.9 and F = 0.5

DE/order against DE/rand and DE/2-Opt. Two crossover schemes, exponential and binomial, were tested for the DE algorithm. The experiment was performed on 30 benchmark functions with 30, 50, and 100 dimensions. Algorithms were evaluated for 51 independent runs and and the results were recorded. In this study, the maximum number of evaluations was set to $1000 \times D$ and the population size was set to 100. A two-sided Wilcoxon statistical test with a confidence level of 95% is performed between compared algorithms. Symbols \$, $\#$ and $*$ denote the compared algorithms are better than, worse than, or similar to DE/order, respectively. 'w/t/l 'in the last row in tables means that DE/order wins in w functions, ties in t functions, and loses in l functions, compared with the compared algorithms.

B. Numerical Results

The mean and the standard deviation of the obtained error values by three algorithms with dimensions 30, 50, and 100 are summarized in the Tables I-III, respectively. Table I indicates the results of algorithms for D=30. From Table I, it can be seen that with the binomial mutation, DE/order/bin performs better than DE/rand/bin and DE/2-Opt/bin on 21 (f_1-f_4, f_6-f_{10},f_{15},f_{17}-f_{22}, f_{24}-f_{25}, f_{27}-f_{28}, f_{30}) and 18 (f_1-f_4,f_6-f_{10},f_{17}-f_{19}, f_{21} , f_{24}-f_{25}, f_{27}-f_{28}, f_{30}), respectively. Also, DE/2-Opt/bin can achieve the better result on one function f_{29}. The DE/rand/bin and DE/2-Opt/bin perform similar to DE/order/bin on 9 and 10 other functions, respectively. It is obvious from Table I, for the exponential mutation, DE/order/exp achieves better results than DE/rand/exp and DE/2-Opt/exp on 21 (f_1-f_4,f_6-f_{10},f_{15},f_{17},f_{20}-f_{22}, f_{24}-f_{25}, f_{27}-f_{30}) and 19 (f_1-f_4, f_6-f_{10},f_{15},f_{17}-f_{21}, f_{24}-f_{25}, f_{27}-f_{28}), respectively. DE/order/exp cannot achieve better results than DE/2-Opt/exp on only one function (f_{29}). In addition, The DE/rand/exp and DE/2-Opt/exp perform similar to DE/order/exp on 9 and 10 other functions, respectively.

The results of algorithms for D=50 are summarized in Table II. It is obvious from Table II that for the binomial mutation, DE/order/bin algorithm outperforms DE/rand/bin and DE/2-Opt/bin on 17 (f_1-f_4, f_6-f_{10},f_{15},f_{17}, f_{20}-f_{21}, f_{25}, f_{27}-f_{28}, f_{30}) and 16 (f_1-f_2,f_4,f_6,f_{10}-f_{11},f_{17}-f_{18}, $f_{20} - f_{22}$, f_{25}, f_{27}-f_{30}), respectively. DE/order/bin cannot achieve better results than DE/2-Opt/bin on only one function (f_{26}). In addition, The DE/rand/bin and DE/2-Opt/bin perform similar to DE/order/exp on 13 other functions. From Table II, it can be seen that with the exponential mutation, DE/order/exp performs better than DE/rand/exp and DE/2-Opt/exp on 23 (f_1-f_4, f_6-f_{11},f_{15},f_{17}-f_{21}, f_{24}-f_{30}) and 17 (f_1-f_4,f_6,f_{11},f_{15},f_{17}-f_{21}, f_{25}, f_{27}-f_{30}), respectively. Also, DE/2-Opt/exp archives the better result on one function f_{23}. The DE/rand/exp and DE/2-Opt/exp perform similar to DE/order/exp on 7 and 12 other functions, respectively.



(a) DE/1/exp (b) DE/2/exp

(c) DE/1/bin (d) DE/2/bin

Fig. 5: The probability of closeness for DE/order, DE/2-opt, DE/win, and DE/rand mutation schemes for the exponential and binomial crossover operators with CR = 0.9 and F = 0.5

Table III indicates the results of algorithms for D=100. From Table III, it can be seen that with the binomial mutation, DE/order/bin performs better than DE/rand/bin and DE/2-Opt/bin on 18 (f_1-f_4, f_6-f_8,f_{10},f_{15},f_{17},f_{20}-f_{22}, f_{25}, f_{27}-f_{30}) and 12 (f_1-f_2,f_4,f_6-f_7,f_{10},f_{17},f_{20}-f_{21} , f_{25}, f_{28}, f_{30}), respectively. Also, DE/2-Opt/bin can achieve the better result

TABLE I: Results of DE/order,DE/rand, and DE/2-Opt algorithms on the CEC-2014 benchmark functions with D=30.

Functions		DE/rand/bin	DE/2-Opt/bin	DE/order/bin	DE/rand/exp	DE/2-Opt/exp	DE/order/exp
f_1	Mean	2.223e+07$^{\$}$	8.379e+06$^{\$}$	3.904e+06	2.305e+07$^{\$}$	8.968e+06$^{\$}$	3.774e+06
	STD	7.018e+06	2.289e+06	1.626e+06	7.450e+06	3.265e+06	1.371e+06
f_2	Mean	3.070e+07$^{\$}$	9.948e+05$^{\$}$	4.988e+04	2.997e+07$^{\$}$	9.434e+05$^{\$}$	5.670e+04
	STD	1.064e+07	4.245e+05	1.986e+04	9.161e+06	4.355e+05	2.644e+04
f_3	Mean	5.654e+02$^{\$}$	3.155e+02$^{\$}$	3.017e+02	5.020e+02$^{\$}$	3.106e+02$^{\$}$	3.008e+02
	STD	9.294e+01	6.059e+00	1.461e+00	5.584e+01	4.942e+00	4.331e-01
f_4	Mean	5.450e+02$^{\$}$	5.143e+02$^{\$}$	4.892e+02	5.466e+02$^{\$}$	5.112e+02$^{\$}$	4.951e+02
	STD	1.035e+01	1.740e+01	1.636e+01	1.235e+01	1.337e+01	1.967e+01
f_5	Mean	5.210e+02$^{\#}$	5.210e+02$^{\#}$	5.210e+02	5.210e+02$^{\#}$	5.210e+02$^{\#}$	5.210e+02
	STD	6.116e-02	4.701e-02	5.538e-02	4.510e-02	5.259e-02	6.179e-02
f_{06}	Mean	6.330e+02$^{\$}$	6.232e+02$^{\$}$	6.139e+02	6.313e+02$^{\$}$	6.256e+02$^{\$}$	6.135e+02
	STD	4.924e+00	7.807e+00	5.752e+00	5.099e+00	7.572e+00	5.582e+00
f_7	Mean	7.012e+02$^{\$}$	7.007e+02$^{\$}$	7.001e+02	7.012e+02$^{\$}$	7.007e+02$^{\$}$	7.001e+02
	STD	6.094e-02	1.312e-01	6.390e-02	6.997e-02	1.287e-01	7.356e-02
f_8	Mean	1.002e+03$^{\$}$	9.970e+02$^{\$}$	9.888e+02	1.003e+03$^{\$}$	9.959e+02$^{\#}$	9.910e+02
	STD	1.352e+01	1.055e+01	1.123e+01	1.110e+01	1.107e+01	1.202e+01
f_9	Mean	1.124e+03$^{\$}$	1.117e+03$^{\$}$	1.112e+03	1.126e+03$^{\$}$	1.117e+03$^{\$}$	1.110e+03
	STD	1.143e+01	1.048e+01	1.103e+01	1.211e+01	1.219e+01	1.144e+01
f_{10}	Mean	7.165e+03$^{\$}$	7.040e+03$^{\$}$	6.750e+03	7.181e+03$^{\$}$	7.069e+03$^{\$}$	6.799e+03
	STD	3.607e+02	2.994e+02	4.748e+02	3.659e+02	3.308e+02	4.728e+02
f_{11}	Mean	8.668e+03$^{\#}$	8.526e+03$^{\#}$	8.609e+03	8.606e+03$^{\#}$	8.607e+03$^{\#}$	8.506e+03
	STD	2.479e+02	3.185e+02	2.843e+02	3.298e+02	2.871e+02	3.611e+02
f_{12}	Mean	1.203e+03$^{\#}$	1.203e+03$^{\#}$	1.203e+03	1.203e+03$^{\#}$	1.203e+03$^{\#}$	1.203e+03
	STD	3.445e-01	3.012e-01	3.916e-01	4.446e-01	4.062e-01	3.040e-01
f_{13}	Mean	1.301e+03$^{\#}$	1.301e+03$^{\#}$	1.301e+03	1.301e+03$^{\#}$	1.301e+03$^{\#}$	1.301e+03
	STD	7.427e-02	7.272e-02	7.793e-02	7.800e-02	6.527e-02	7.458e-02
f_{14}	Mean	1.400e+03$^{\#}$	1.400e+03$^{\#}$	1.400e+03	1.400e+03$^{\#}$	1.400e+03$^{\#}$	1.400e+03
	STD	8.235e-02	9.617e-02	1.955e-01	9.141e-02	4.650e-02	1.843e-01
f_{15}	Mean	1.522e+03$^{\$}$	1.519e+03$^{\#}$	1.519e+03	1.522e+03$^{\$}$	1.520e+03$^{\$}$	1.519e+03
	STD	1.548e+00	1.176e+00	1.263e+00	1.200e+00	1.137e+00	1.049e+00
f_{16}	Mean	1.613e+03$^{\#}$	1.613e+03$^{\#}$	1.613e+03	1.613e+03$^{\#}$	1.613e+03$^{\#}$	1.613e+03
	STD	1.852e-01	1.942e-01	2.064e-01	1.770e-01	1.488e-01	1.831e-01
f_{17}	Mean	1.983e+04$^{\$}$	5.576e+03$^{\$}$	4.461e+03	4.847e+04$^{\$}$	7.270e+03$^{\$}$	5.121e+03
	STD	9.899e+03	7.349e+02	4.119e+02	2.493e+04	1.261e+03	7.920e+02
f_{18}	Mean	2.616e+03$^{\$}$	1.962e+03$^{\$}$	1.936e+03	2.246e+03$^{\#}$	1.934e+03$^{\$}$	1.921e+03
	STD	2.480e+02	1.821e+01	3.233e+01	1.080e+02	1.768e+01	2.904e+01
f_{19}	Mean	1.912e+03$^{\$}$	1.908e+03$^{\$}$	1.907e+03	1.912e+03$^{\#}$	1.908e+03$^{\$}$	1.907e+03
	STD	1.551e+00	1.265e+00	8.773e-01	1.577e+00	1.335e+00	1.051e+00
f_{20}	Mean	2.095e+03$^{\$}$	2.068e+03$^{\#}$	2.066e+03	2.087e+03$^{\$}$	2.068e+03$^{\$}$	2.061e+03
	STD	1.357e+01	8.630e+00	1.019e+01	9.359e+00	8.934e+00	1.041e+01
f_{21}	Mean	4.177e+03$^{\$}$	3.692e+03$^{\$}$	3.562e+03	4.087e+03$^{\$}$	3.616e+03$^{\$}$	3.532e+03
	STD	2.720e+02	1.726e+02	1.846e+02	2.806e+02	2.231e+02	1.761e+02
f_{22}	Mean	2.830e+03$^{\$}$	2.774e+03$^{\#}$	2.736e+03	2.836e+03$^{\$}$	2.776e+03$^{\#}$	2.693e+03
	STD	1.333e+02	1.439e+02	1.426e+02	1.199e+02	1.122e+02	1.835e+02
f_{23}	Mean	2.616e+03$^{\#}$	2.615e+03$^{\#}$	2.615e+03	2.616e+03$^{\#}$	2.615e+03$^{\#}$	2.615e+03
	STD	1.189e-01	8.777e-03	7.592e-04	1.613e-01	1.219e-02	1.243e-03
f_{24}	Mean	2.637e+03$^{\$}$	2.628e+03$^{\$}$	2.627e+03	2.636e+03$^{\$}$	2.628e+03$^{\$}$	2.627e+03
	STD	3.601e+00	2.236e+00	3.682e+00	3.235e+00	2.250e+00	4.537e+00
f_{25}	Mean	2.712e+03$^{\$}$	2.708e+03$^{\$}$	2.705e+03	2.711e+03$^{\$}$	2.707e+03$^{\$}$	2.705e+03
	STD	1.782e+00	1.529e+00	9.039e-01	1.944e+00	1.797e+00	9.852e-01
f_{26}	Mean	2.701e+03$^{\#}$	2.701e+03$^{\#}$	2.701e+03	2.701e+03$^{\#}$	2.701e+03$^{\#}$	2.701e+03
	STD	8.315e-02	6.167e-02	7.035e-02	6.498e-02	7.705e-02	7.302e-02
f_{27}	Mean	3.444e+03$^{\$}$	3.249e+03$^{\$}$	3.164e+03	3.491e+03$^{\$}$	3.254e+03$^{\$}$	3.151e+03
	STD	1.341e+02	1.065e+02	6.616e+01	1.191e+02	1.121e+02	6.189e+01
f_{28}	Mean	3.819e+03$^{\$}$	3.740e+03$^{\$}$	3.670e+03	3.812e+03$^{\$}$	3.761e+03$^{\$}$	3.654e+03
	STD	3.847e+01	6.486e+01	5.015e+01	4.854e+01	5.936e+01	4.819e+01
f_{29}	Mean	1.773e+05$^{\#}$	4.992e+05*	5.033e+05	5.460e+05$^{\$}$	1.798e+05*	3.385e+05
	STD	1.214e+06	1.996e+06	2.016e+06	2.182e+06	1.249e+06	1.671e+06
f_{30}	Mean	7.778e+03$^{\$}$	6.217e+03$^{\$}$	5.939e+03	7.507e+03$^{\$}$	5.869e+03$^{\#}$	6.500e+03
	STD	2.489e+03	1.321e+03	2.892e+03	4.740e+03	8.506e+02	5.290e+03
	w/t/l	21/9/0	18/11/1	-	20/10/0	19/10/1	-

on 4 functions; f_8,f_{19},f_{24}, and f_{29}. The DE/rand/bin and DE/2-Opt/bin perform similar to DE/order/bin on 12 and 14 other functions, respectively. For the exponential mutation, it is obvious from Table III, DE/order/exp achieves better results than DE/rand/exp and DE/2-Opt/exp on 16 (f_1-f_4, f_6-f_8,f_{10},f_{15},f_{17},f_{20}-f_{21}, f_{25}, f_{27}-f_{28}, f_{30}) and 14 (f_1-f_4, f_6-f_7,f_{10},f_{17},f_{20}-f_{21}, f_{22}-f_{25}, f_{29}-f_{30}), respectively. DE/rand/exp and DE/2-Opt/exp can obtain better results than DE/order/exp on one function f_{29} and 2 functions (f_{19} and f_{27}), respectively. In addition, the DE/rand/exp and DE/2-Opt/exp perform similar to DE/order/exp on 13 and 14 other functions, respectively.

V. Conclusion Remarks and future directions

In this paper, mutation schemes DE/rand, DE/win, DE/2-Opt, and DE/order were analyzed to study the effect of ordering vectors in the mutation scheme by monte carlo simulations. In each iteration of monte carlo simulations, they generate a random solution as the optimal solution and also some other random solutions as the candidate solutions for calculating mutation schemes. Then, the closeness probability of new generated trial candidate solutions for mutation schemes are computed. Monte carlo simulations have shown that all mutation schemes with considering an ordering vectors in the mutation scheme (placing the solutions with the better fitness as base vector or using a full order of candidate solutions according to their fitness for vectors in mutation scheme) can perform better than the random mutation (DE/rand). In addition, we proposed the modified DE algorithm with the order mutation scheme (DE/order)which is evaluated on CEC-2014 benchmark functions with three dimensions; 30, 50, and 100. In future, we are planning to analyze other versions of mutation schemes by monte carlo simulations. In addition, we are interested in analyzing the impact of each mutation scheme on the performance of DE.

References

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

[2] R. Storn and K. Price, *Differential evolution–a simple and efficient adaptive scheme for global optimization over continuous spaces*. ICSI Berkeley, 1995, vol. 3.

[3] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE transactions on evolutionary computation*, vol. 15, no. 1, pp. 4–31, 2011.

[4] S. Das, S. S. Mullick, and P. N. Suganthan, "Recent advances in differential evolution–an updated survey," *Swarm and Evolutionary Computation*, vol. 27, pp. 1–30, 2016.

[5] H.-Y. Fan and J. Lampinen, "A trigonometric mutation operation to differential evolution," *Journal of global optimization*, vol. 27, no. 1, pp. 105–129, 2003.

[6] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 526–553, 2009.

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

[8] J. Zhang and A. C. Sanderson, "Jade: adaptive differential evolution with optional external archive," *IEEE transactions on evolutionary computation*, vol. 13, no. 5, pp. 945–958, 2009.

[9] J. Teo, "Exploring dynamic self-adaptive populations in differential evolution," *Soft Computing*, vol. 10, no. 8, pp. 673–686, 2006.

[10] A. Zamuda and J. Brest, "Self-adaptive control parameters? randomization frequency and propagations in differential evolution," *Swarm and Evolutionary Computation*, vol. 25, pp. 72–99, 2015.

[11] F. Campelo and M. Botelho, "Experimental investigation of recombination operators for differential evolution," in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*. ACM, 2016, pp. 221–228.

[12] M. Weber, V. Tirronen, and F. Neri, "Scale factor inheritance mechanism in distributed differential evolution," *Soft Computing*, vol. 14, no. 11, pp. 1187–1207, 2010.

[13] R. Mallipeddi, "Harmony search based parameter ensemble adaptation for differential evolution," *Journal of Applied Mathematics*, vol. 2013, 2013.

[14] S. Das, A. Konar, and U. K. Chakraborty, "Two improved differential evolution schemes for faster global search," in *Proceedings of the 7th annual conference on Genetic and evolutionary computation*. ACM, 2005, pp. 991–998.

[15] R. Tanabe and A. Fukunaga, "How far are we from an optimal, adaptive de?" in *International Conference on Parallel Problem Solving from Nature*. Springer, 2016, pp. 145–155.

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

[17] A. Esmailzadeh and S. Rahnamayan, "Enhanced differential evolution using center-based sampling," in *2011 IEEE Congress of Evolutionary Computation (CEC)*. IEEE, 2011, pp. 2641–2648.

[18] M. Yang, C. Li, Z. Cai, and J. Guan, "Differential evolution with auto-enhanced population diversity," *IEEE transactions on cybernetics*, vol. 45, no. 2, pp. 302–315, 2015.

[19] J.-H. Zhong, M. Shen, J. Zhang, H. S.-H. Chung, Y.-H. Shi, and Y. Li, "A differential evolution algorithm with dual populations for solving periodic railway timetable scheduling problem," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 4, pp. 512–527, 2013.

[20] S. Biswas, S. Kundu, S. Das, and A. V. Vasilakos, "Teaching and learning best differential evoltuion with self adaptation for real parameter optimization," in *2013 IEEE Congress on Evolutionary Computation*. IEEE, 2013, pp. 1115–1122.

[21] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *2005 IEEE congress on evolutionary computation*, vol. 2. IEEE, 2005, pp. 1785–1791.

[22] C.-W. Chiang, W.-P. Lee, and J.-S. Heh, "A 2-opt based differential evolution for global optimization," *Applied Soft Computing*, vol. 10, no. 4, pp. 1200–1207, 2010.

[23] M.-F. Yeh, H.-C. Lu, T.-H. Chen, and P.-J. Huang, "System identification using differential evolution with winner mutation strategy," in *Machine Learning and Cybernetics (ICMLC), 2014 International Conference on*, vol. 1. IEEE, 2014, pp. 77–81.

TABLE II: Results of DE/order,DE/rand, and DE/2-Opt algorithms on the CEC-2014 benchmark functions with D=50.

Functions		DE/rand/bin	DE/2-Opt/bin	DE/order/bin	DE/rand/exp	DE/2-Opt/exp	DE/order/exp
f_1	Mean	9.823e+07$^{\$}$	4.303e+07$^{\$}$	2.079e+07	8.649e+07$^{\$}$	3.408e+07$^{\$}$	1.740e+07
	STD	2.459e+07	1.271e+07	7.515e+06	1.767e+07	8.780e+06	7.826e+06
f_2	Mean	3.004e+07$^{\$}$	5.012e+05$^{\$}$	1.895e+04	2.797e+07$^{\$}$	4.390e+05$^{\$}$	2.164e+04
	STD	8.724e+06	2.176e+05	8.907e+03	9.538e+06	2.565e+05	1.149e+04
f_3	Mean	5.425e+03$^{\$}$	2.383e+03$^{\#}$	2.135e+03	6.947e+03$^{\$}$	3.932e+03$^{\$}$	3.376e+03
	STD	1.311e+03	1.025e+03	1.273e+03	1.934e+03	1.559e+03	1.998e+03
f_4	Mean	5.235e+02$^{\$}$	5.004e+02$^{\$}$	4.994e+02	5.241e+02$^{\$}$	5.048e+02$^{\$}$	4.982e+02
	STD	1.076e+01	4.186e+00	5.157e+00	1.400e+01	1.679e+01	2.466e+00
f_5	Mean	5.212e+02$^{\#}$	5.212e+02$^{\#}$	5.212e+02	5.212e+02$^{\#}$	5.212e+02$^{\#}$	5.212e+02
	STD	4.356e-02	4.665e-02	4.654e-02	3.025e-02	3.521e-02	2.819e-02
f_6	Mean	6.491e+02$^{\$}$	6.331e+02$^{\$}$	6.198e+02	6.505e+02$^{\$}$	6.307e+02$^{\$}$	6.206e+02
	STD	9.879e+00	1.135e+01	6.984e+00	1.052e+01	1.048e+01	7.656e+00
f_7	Mean	7.012e+02$^{\$}$	7.005e+02$^{\#}$	7.000e+02	7.012e+02$^{\$}$	7.005e+02$^{\#}$	7.000e+02
	STD	8.701e-02	1.396e-01	1.952e-02	7.857e-02	1.323e-01	2.054e-02
f_8	Mean	1.190e+03$^{\$}$	1.178e+03$^{\#}$	1.176e+03	1.185e+03$^{\$}$	1.177e+03$^{\#}$	1.174e+03
	STD	1.611e+01	1.918e+01	1.305e+01	1.372e+01	1.569e+01	1.733e+01
f_9	Mean	1.316e+03$^{\$}$	1.305e+03$^{\#}$	1.300e+03	1.314e+03$^{\$}$	1.299e+03$^{\#}$	1.298e+03
	STD	1.766e+01	1.835e+01	1.821e+01	1.917e+01	1.956e+01	1.304e+01
f_{10}	Mean	1.344e+04$^{\$}$	1.339e+04$^{\$}$	1.307e+04	1.361e+04$^{\$}$	1.328e+04$^{\#}$	1.310e+04
	STD	3.303e+02	4.236e+02	5.282e+02	4.237e+02	5.379e+02	5.273e+02
f_{11}	Mean	1.504e+04$^{\#}$	1.507e+04$^{\$}$	1.491e+04	1.509e+04$^{\$}$	1.512e+04$^{\$}$	1.480e+04
	STD	4.336e+02	4.058e+02	4.632e+02	3.836e+02	3.557e+02	5.362e+02
f_{12}	Mean	1.204e+03$^{\#}$	1.204e+03$^{\#}$	1.204e+03	1.204e+03$^{\#}$	1.204e+03$^{\#}$	1.204e+03
	STD	3.488e-01	3.511e-01	3.424e-01	3.526e-01	3.691e-01	4.002e-01
f_{13}	Mean	1.301e+03$^{\#}$	1.301e+03$^{\#}$	1.301e+03	1.301e+03$^{\#}$	1.301e+03$^{\#}$	1.301e+03
	STD	7.170e-02	7.219e-02	7.445e-02	7.784e-02	7.711e-02	7.765e-02
f_{14}	Mean	1.401e+03$^{\#}$	1.401e+03$^{\#}$	1.401e+03	1.401e+03$^{\#}$	1.400e+03$^{\#}$	1.401e+03
	STD	2.143e-01	2.326e-01	2.295e-01	1.942e-01	2.110e-01	2.431e-01
f_{15}	Mean	1.539e+03$^{\$}$	1.537e+03$^{\#}$	1.536e+03	1.539e+03$^{\$}$	1.537e+03$^{\$}$	1.536e+03
	STD	2.245e+00	1.641e+00	1.844e+00	1.904e+00	1.673e+00	1.712e+00
f_{16}	Mean	1.623e+03$^{\#}$	1.623e+03$^{\#}$	1.623e+03	1.623e+03$^{\#}$	1.623e+03*	1.623e+03
	STD	1.977e-01	1.998e-01	2.359e-01	1.951e-01	1.744e-01	2.064e-01
f_{17}	Mean	7.434e+05$^{\$}$	2.281e+05$^{\$}$	1.332e+05	1.305e+06$^{\$}$	3.607e+05$^{\$}$	2.597e+05
	STD	3.147e+05	1.608e+05	1.614e+05	3.930e+05	2.039e+05	2.808e+05
f_{18}	Mean	1.281e+04$^{\#}$	2.880e+03$^{\$}$	2.565e+03	1.674e+04$^{\$}$	3.099e+03$^{\$}$	2.779e+03
	STD	5.444e+03	5.554e+02	1.059e+03	8.394e+03	5.664e+02	1.101e+03
f_{19}	Mean	1.924e+03$^{\#}$	1.916e+03$^{\#}$	1.916e+03	1.925e+03$^{\$}$	1.919e+03$^{\$}$	1.916e+03
	STD	4.115e+00	2.495e+00	6.926e+00	6.403e+00	6.959e+00	2.735e+00
f_{20}	Mean	2.546e+03$^{\$}$	2.320e+03$^{\$}$	2.190e+03	2.782e+03$^{\$}$	2.379e+03$^{\$}$	2.251e+03
	STD	2.055e+02	3.649e+02	4.583e+01	4.511e+02	2.428e+02	6.422e+01
f_{21}	Mean	4.000e+04$^{\$}$	1.512e+04$^{\$}$	1.018e+04	4.585e+04$^{\$}$	2.244e+04$^{\$}$	1.113e+04
	STD	1.935e+04	1.595e+04	6.537e+03	2.182e+04	2.393e+04	1.056e+04
f_{22}	Mean	3.870e+03$^{\#}$	3.898e+03$^{\$}$	3.826e+03	3.897e+03$^{\#}$	3.866e+03$^{\#}$	3.842e+03
	STD	1.629e+02	1.644e+02	1.538e+02	1.801e+02	1.499e+02	1.723e+02
f_{23}	Mean	2.644e+03$^{\#}$	2.644e+03$^{\#}$	2.644e+03	2.644e+03$^{\#}$	2.644e+03*	2.644e+03
	STD	7.842e-02	2.944e-03	1.094e-03	6.871e-02	3.409e-03	2.624e-04
f_{24}	Mean	2.682e+03$^{\#}$	2.672e+03$^{\#}$	2.673e+03	2.682e+03$^{\$}$	2.672e+03$^{\#}$	2.673e+03
	STD	3.324e+00	1.870e+00	2.555e+00	2.849e+00	2.024e+00	2.687e+00
f_{25}	Mean	2.732e+03$^{\$}$	2.720e+03$^{\$}$	2.711e+03	2.733e+03$^{\$}$	2.719e+03$^{\$}$	2.711e+03
	STD	6.646e+00	3.952e+00	2.389e+00	6.286e+00	3.782e+00	2.380e+00
f_{26}	Mean	2.701e+03$^{\#}$	2.710e+03*	2.736e+03	2.711e+03$^{\$}$	2.705e+03$^{\#}$	2.707e+03
	STD	6.156e-02	4.494e+01	8.089e+01	5.260e+01	3.266e+01	3.598e+01
f_{27}	Mean	4.067e+03$^{\$}$	3.593e+03$^{\$}$	3.377e+03	4.072e+03$^{\$}$	3.572e+03$^{\$}$	3.397e+03
	STD	2.476e+02	1.955e+02	1.121e+02	2.172e+02	2.170e+02	1.150e+02
f_{28}	Mean	4.247e+03$^{\$}$	4.096e+03$^{\$}$	4.004e+03	4.286e+03$^{\$}$	4.120e+03$^{\$}$	4.028e+03
	STD	1.174e+02	1.039e+02	1.443e+02	8.819e+01	1.202e+02	8.337e+01
f_{29}	Mean	2.115e+07$^{\#}$	2.410e+07$^{\$}$	1.832e+07	2.111e+07$^{\$}$	2.092e+07$^{\$}$	1.509e+07
	STD	1.855e+07	1.998e+07	2.061e+07	1.945e+07	2.203e+07	1.987e+07
f_{30}	Mean	1.711e+04$^{\$}$	1.328e+04$^{\$}$	1.219e+04	1.718e+04$^{\$}$	1.303e+04$^{\$}$	1.219e+04
	STD	1.963e+03	9.0589e+02	1.565e+03	1.773e+03	9.698e+02	7.516e+02
	w/t/l	17/13/0	16/14/1	-	23/7/0	17/12/1	-

TABLE III: Results of DE/order,DE/rand, and DE/2-Opt algorithms on the CEC-2014 benchmark functions with D=100.

Functions		DE/rand/bin	DE/2-Opt/bin	DE/order/bin	DE/rand/exp	DE/2-Opt/exp	DE/order/exp
f_1	Mean	5.01e+08$^{\$}$	1.543e+08$^{\$}$	7.365e+07	3.389e+08$^{\$}$	1.122e+08$^{\$}$	6.098e+07
	STD	1.29e+08	4.260e+07	2.227e+07	9.244e+07	3.762e+07	2.053e+07
f_2	Mean	2.18e+07$^{\$}$	4.516e+05$^{\$}$	3.421e+04	3.409e+07$^{\$}$	1.131e+06$^{\$}$	5.628e+04
	STD	6.71e+06	1.571e+05	2.501e+04	1.315e+07	8.312e+05	3.161e+04
f_3	Mean	2.38e+04$^{\$}$	1.597e+04$^{\#}$	1.514e+04	5.526e+04$^{\$}$	3.982e+04$^{\$}$	3.197e+04
	STD	6.20e+03	5.491e+03	6.494e+03	9.289e+03	8.724e+03	7.166e+03
f_4	Mean	7.12e+02$^{\$}$	6.772e+02$^{\$}$	6.486e+02	7.362e+02$^{\$}$	6.928e+02$^{\$}$	6.615e+02
	STD	4.08e+01	4.539e+01	4.386e+01	4.401e+01	4.322e+01	3.743e+01
f_5	Mean	5.21e+02$^{\#}$	5.214e+02$^{\#}$	5.214e+02	5.214e+02$^{\#}$	5.214e+02$^{\#}$	5.214e+02
	STD	2.06e-02	2.454e-02	2.078e-02	2.110e-02	2.813e-02	2.486e-02
f_6	Mean	7.06e+02$^{\$}$	6.550e+02$^{\$}$	6.435e+02	6.982e+02$^{\$}$	6.546e+02$^{\$}$	6.479e+02
	STD	2.22e+01	1.876e+01	8.719e+00	2.463e+01	1.548e+01	7.946e+00
f_7	Mean	7.01e+02$^{\$}$	7.003e+02$^{\$}$	7.000e+02	7.013e+02$^{\$}$	7.006e+02$^{\$}$	7.001e+02
	STD	7.11e-02	1.113e-01	2.320e-02	8.945e-02	1.652e-01	2.935e-02
f_8	Mean	1.68e+03$^{\$}$	1.660e+03*	1.646e+03	1.685e+03$^{\$}$	1.666e+03$^{\#}$	1.665e+03
	STD	2.24e+01	2.657e+01	3.722e+01	2.501e+01	2.977e+01	3.403e+01
f_9	Mean	1.82e+03$^{\#}$	1.808e+03$^{\#}$	1.814e+03	1.829e+03$^{\#}$	1.820e+03$^{\#}$	1.818e+03
	STD	2.48e+01	2.814e+01	2.715e+01	2.653e+01	2.940e+01	2.467e+01
f_{10}	Mean	3.02e+04$^{\$}$	3.003e+04$^{\$}$	2.941e+04	3.023e+04$^{\$}$	3.010e+04$^{\$}$	2.939e+04
	STD	6.60e+02	8.643e+02	7.485e+02	6.208e+02	6.889e+02	9.515e+02
f_{11}	Mean	3.25e+04$^{\#}$	3.240e+04$^{\#}$	3.230e+04	3.233e+04$^{\#}$	3.238e+04$^{\#}$	3.237e+04
	STD	5.60e+02	4.881e+02	5.196e+02	5.097e+02	5.734e+02	4.911e+02
f_{12}	Mean	1.20e+03$^{\#}$	1.204e+03$^{\#}$	1.204e+03	1.204e+03$^{\#}$	1.204e+03$^{\#}$	1.204e+03
	STD	2.78e-01	2.110e-01	3.496e-01	2.759e-01	3.102e-01	2.907e-01
f_{13}	Mean	1.30e+03$^{\#}$	1.301e+03$^{\#}$	1.301e+03	1.301e+03$^{\#}$	1.301e+03$^{\#}$	1.301e+03
	STD	6.98e-02	6.993e-02	6.251e-02	6.426e-02	6.944e-02	7.894e-02
f_{14}	Mean	1.40e+03$^{\#}$	1.400e+03$^{\#}$	1.400e+03	1.400e+03$^{\#}$	1.400e+03$^{\#}$	1.400e+03
	STD	1.48e-01	9.685e-02	1.451e-01	6.692e-02	1.0974e-01	2.054e-01
f_{15}	Mean	1.59e+03$^{\$}$	1.585e+03$^{\#}$	1.584e+03	1.590e+03$^{\$}$	1.588e+03$^{\#}$	1.587e+03
	STD	3.76e+00	3.647e+00	2.680e+00	4.292e+00	3.296e+00	3.543e+00
f_{16}	Mean	1.65e+03$^{\#}$	1.647e+03$^{\#}$	1.647e+03	1.647e+03$^{\#}$	1.647e+03$^{\#}$	1.647e+03
	STD	2.03e-01	2.348e-01	2.151e-01	2.666e-01	2.097e-01	2.125e-01
f_{17}	Mean	2.39e+07$^{\$}$	9.131e+06$^{\$}$	4.644e+06	2.571e+07$^{\$}$	1.055e+07$^{\$}$	5.481e+06
	STD	7.162e+06	2.602e+06	1.719e+06	7.117e+06	3.515e+06	2.122e+06
f_{18}	Mean	5.110e+03$^{\#}$	4.470e+03$^{\#}$	4.400e+03	4.886e+03$^{\#}$	4.237e+03$^{\#}$	5.135e+03
	STD	3.196e+03	2.749e+03	2.327e+03	2.906e+03	2.551e+03	3.258e+03
f_{19}	Mean	2.002e+03$^{\#}$	2.000e+03*	2.001e+03	2.003e+03$^{\#}$	2.000e+03*	2.003e+03
	STD	3.588e+00	7.358e+00	5.780e+00	9.440e+00	3.338e+00	6.039e+00
f_{20}	Mean	4.920e+04$^{\$}$	2.801e+04$^{\$}$	1.544e+04	2.599e+04$^{\$}$	1.742e+04$^{\$}$	1.135e+04
	STD	1.743e+04	1.189e+04	7.942e+03	1.075e+04	8.687e+03	6.528e+03
f_{21}	Mean	4.301e+06$^{\$}$	1.949e+06$^{\$}$	1.229e+06	3.577e+06$^{\$}$	1.636e+06$^{\$}$	1.228e+06
	STD	1.218e+06	9.975e+05	6.828e+05	1.496e+06	7.551e+05	6.386e+05
f_{22}	Mean	6.841e+03$^{\$}$	6.785e+03$^{\#}$	6.754e+03	6.793e+03$^{\#}$	6.770e+03$^{\#}$	6.826e+03
	STD	2.248e+02	3.232e+02	2.445e+02	3.081e+02	2.307e+02	1.993e+02
f_{23}	Mean	2.650e+03$^{\#}$	2.649e+03$^{\#}$	2.648e+03	2.650e+03$^{\#}$	2.649e+03$^{\#}$	2.649e+03
	STD	7.264e-01	4.329e-01	2.214e-01	1.017e+00	6.452e-01	3.356e-01
f_{24}	Mean	2.794e+03$^{\#}$	2.791e+03*	2.797e+03	2.796e+03$^{\#}$	2.795e+03$^{\$}$	2.803e+03
	STD	4.856e+00	5.336e+00	7.258e+00	5.763e+00	6.282e+00	7.540e+00
f_{25}	Mean	2.851e+03$^{\$}$	2.776e+03$^{\$}$	2.738e+03	2.831e+03$^{\$}$	2.764e+03$^{\$}$	2.738e+03
	STD	2.950e+01	1.957e+01	7.118e+00	2.846e+01	1.269e+01	4.955e+00
f_{26}	Mean	2.837e+03$^{\#}$	2.855e+03$^{\#}$	2.842e+03	2.842e+03$^{\#}$	2.840e+03$^{\#}$	2.830e+03
	STD	1.502e+02	1.129e+02	1.154e+02	1.343e+02	1.171e+02	1.026e+02
f_{27}	Mean	4.543e+03$^{\$}$	3.805e+03$^{\#}$	3.811e+03	4.439e+03$^{\$}$	3.89e+03*	3.949e+03
	STD	3.461e+02	1.598e+02	1.492e+02	2.495e+02	1.340e+02	1.306e+02
f_{28}	Mean	5.938e+03$^{\$}$	5.533e+03$^{\$}$	5.388e+03	6.075e+03$^{\$}$	5.553e+03$^{\#}$	5.711e+03
	STD	4.705e+02	4.697e+02	4.638e+02	6.037e+02	4.441e+02	5.872e+02
f_{29}	Mean	3.882e+07$^{\$}$	1.816e+07*	3.233e+07	3.367e+07*	4.887e+07$^{\$}$	4.057e+07
	STD	4.505e+07	3.714e+07	4.629e+07	4.617e+07	5.305e+07	5.406e+07
f_{30}	Mean	2.984e+04$^{\$}$	1.659e+04$^{\$}$	1.290e+04	2.643e+04$^{\$}$	1.733e+04$^{\$}$	1.303e+04
	STD	6.902e+03	3.936e+03	1.816e+03	5.744e+03	6.171e+03	1.892e+03
	w/t/l	18/12/0	12/14/4	-	16/13/1	14/14/2	-